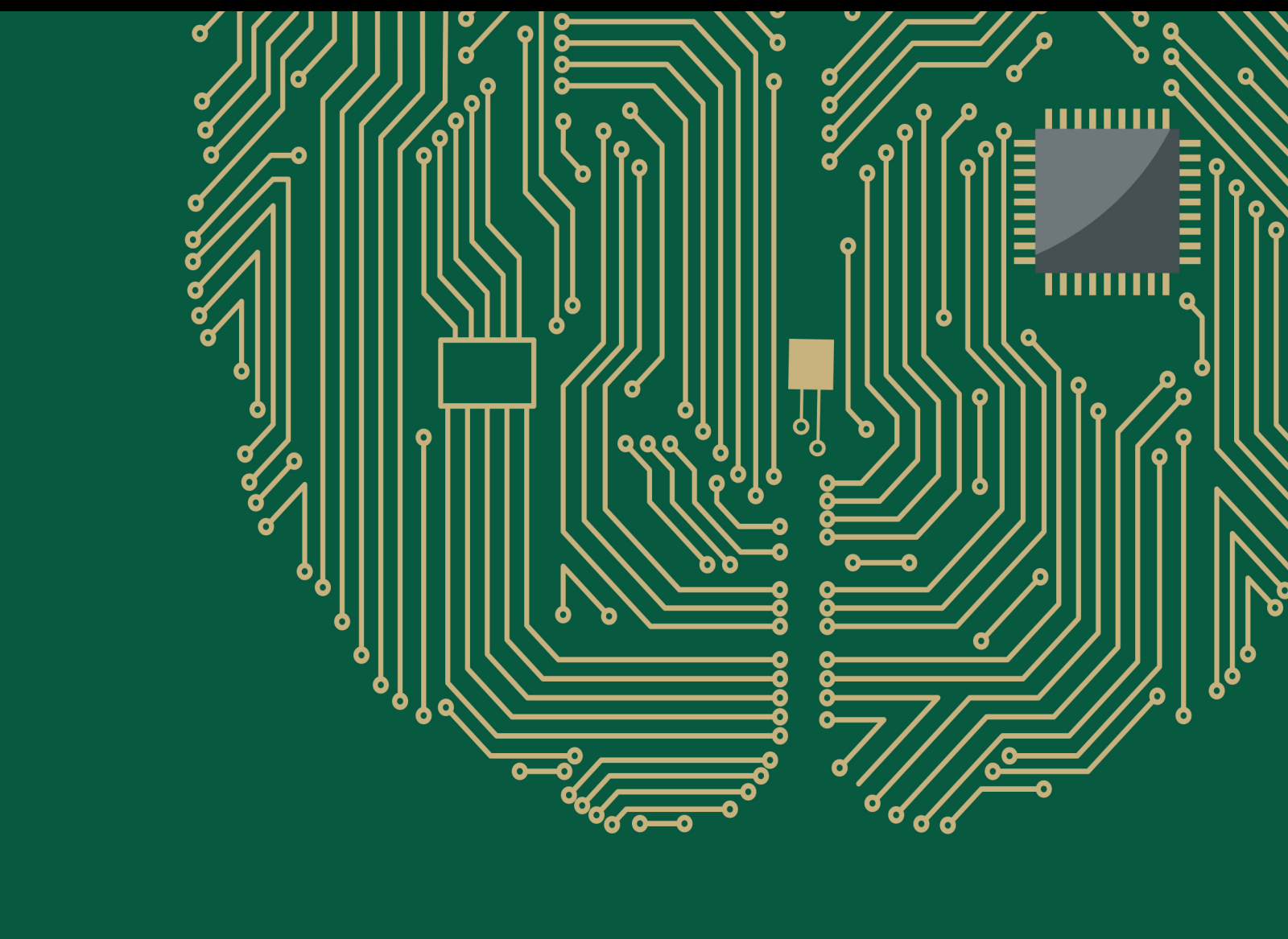
Computational Intelligence and Neuroscience

Nature-Inspired Optimization Algorithms for Neuro-Fuzzy Models in Real World Control and Robotics Applications

Lead Guest Editor: Fevrier Valdez
Guest Editors: Oscar Castillo, Amita Jain, and Dipak K. Jana



Nature-Inspired Optimization Algorithms for Neuro-Fuzzy Models in Real World Control and Robotics Applications

Computational Intelligence and Neuroscience

Nature-Inspired Optimization Algorithms for Neuro-Fuzzy Models in Real World Control and Robotics Applications

Lead Guest Editor: Fevrier Valdez
Guest Editors: Oscar Castillo, Amita Jain, and Dipak K. Jana

Copyright © 2019 Hindawi. All rights reserved.

This is a special issue published in "Computational Intelligence and Neuroscience." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board

Ricardo Aler, Spain
Amparo Alonso-Betanzos, Spain
Pietro Aricò, Italy
Hasan Ayaz, USA
Sylvain Baillet, Canada
Roman Bartak, Czech Republic
Theodore W. Berger, USA
Daniele Bibbo, Italy
Vince D. Calhoun, USA
Francesco Camastra, Italy
Michela Chiappalone, Italy
Andrzej Cichocki, Japan
Jens Christian Claussen, UK
Silvia Conforto, Italy
António D. P. Correia, Portugal
Justin Dauwels, Singapore
Christian W. Dawson, UK
Carmen De Maio, Italy
Sergio Decherchi, Italy
Paolo Del Giudice, Italy
Maria Jose del Jesus, Spain
Arnaud Delorme, France
Thomas DeMarse, USA
Anastasios D. Doulamis, Greece
Steven L. Fernandes, USA
Juan Carlos Fernández, Spain
Piotr Franaszczuk, USA
Leonardo Franco, Spain
Paolo Gastaldo, Italy
Samanwoy Ghosh-Dastidar, USA
Manuel Graña, Spain
Rodolfo E. Haber, Spain
Dominic Heger, Germany
Stephen Helms Tillery, USA
J. A. Hernández-Pérez, Mexico
Luis Javier Herrera, Spain
Etienne Hugues, USA
Ryotaro Kamimura, Japan
Pasi A. Karjalainen, Finland
Elpida Keravnou, Cyprus
Raşit Köker, Turkey
Dean J. Krusienski, USA
Fabio La Foresta, Italy
Antonino Laudani, Italy
Maciej Lawrynczuk, Poland
Mikhail A. Lebedev, USA
Cheng-Jian Lin, Taiwan
Giosuè Lo Bosco, Italy
Ezequiel López-Rubio, Spain
Bruce J. MacLennan, USA
Reinoud Maex, Belgium
Kezhi Mao, Singapore
Sergio Martinoia, Italy
Laura Marzetti, Italy
Elio Masciari, Italy
Paolo Massobrio, Italy
Gerard McKee, Nigeria
Michele Migliore, Italy
Paulo Moura Oliveira, Portugal
Debajyoti Mukhopadhyay, India
Klaus Obermayer, Germany
Karim G. Oweiss, USA
Massimo Panella, Italy
Fivos Panetsos, Spain
David M Powers, Australia
Sandhya Samarasinghe, New Zealand
Saeid Sanei, UK
Friedhelm Schwenker, Germany
Victor R. L. Shen, Taiwan
Fabio Solari, Italy
Jussi Tohka, Finland
Carlos M. Travieso-González, Spain
Lefteri Tsoukalas, USA
Pablo Varona, Spain
Roberto A. Vazquez, Mexico
Meel Velliste, USA
Mario Versaci, Italy
Francois B. Vialatte, France
Thomas Villmann, Germany
Ivan Volosyak, Germany
Cornelio Yáñez-Márquez, Mexico
Michal Zochowski, USA
Rodolfo Zunino, Italy

Contents

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 9128451, 2 pages
https://doi.org/10.1155/2019/9128451

Editorial

Nature-Inspired Optimization Algorithms for Neuro-Fuzzy Models in Real-World Control and Robotics Applications

Fevrier Valdez,[1] **Oscar Castillo**,[1] **Amita Jain**,[2] **and Dipak K. Jana**[3]

[1]*Tijuana Institute of Technology, Tijuana, BC, Mexico*
[2]*Ambedkar Institute of Advanced Communication Technologies and Research, Delhi, India*
[3]*Haldia Institute of Technology, Haldia, India*

Correspondence should be addressed to Fevrier Valdez; fevrier@tectijuana.mx

Received 4 April 2019; Accepted 4 April 2019; Published 15 April 2019

Copyright © 2019 Fevrier Valdez et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nature-inspired optimization algorithms are a recent topic of research, and they are based on using some nature-inspired behaviors to solve optimization problems. Currently, a large number of approaches have been developed in this area, such as particle swarm optimization, bat algorithm, ant colony optimization, bee colony, dolphin algorithm, wolf search, flower pollination algorithm, and cat swarm. However, how to design efficient nature-inspired algorithms and how to use these algorithms for real-world application problems in control and robotics are still important issues. In particular, the design of neuro-fuzzy models, like type 2 fuzzy neural networks, type 1 fuzzy neural models, and intuitionistic fuzzy neural networks, has some current interest. In addition, new emerging neural models have been recently proposed. In all these models, a common problem is how to obtain an optimal structure, which can be handled by nature-inspired optimization algorithms. This special issue aims to bring researchers to report their latest research work on development of new nature-inspired algorithms or innovative applications of existing algorithms in the design of neural models for real-world applications in control and robotics, with an ultimate goal of exploring future research directions. In this special issue, we have five papers selected after a careful reviewing process. The five papers are representative of the current state of the art in this area.

G. López-Vázquez et al. present a grammatical evolution- (GE-) based methodology to automatically design third-generation artificial neural networks (ANNs), also known as spiking neural networks (SNNs), for solving supervised classification problems. The proposal performs the SNN design by exploring the search space of three-layered feedforward topologies with configured synaptic connections (weights and delays) so that no explicit training is carried out. Besides, the designed SNNs have partial connections between input and hidden layers which may contribute to avoid redundancies and reduce the dimensionality of input feature vectors. The proposal was tested on several well-known benchmark datasets from the UCI repository and statistically compared against a similar design methodology for second-generation ANNs and an adapted version of that methodology for SNNs; also, the results of the two methodologies and the proposed one were improved by changing the fitness function in the design process. The proposed methodology shows competitive and consistent results, and the statistical tests support the conclusion that the designs produced by the proposal perform better than those produced by other methodologies.

On the other hand, R. Soto et al. presented a binary cat swarm optimization for solving the manufacturing cell design problem (MCDP). This problem divides an industrial production plant into a certain number of cells. Each cell contains machines with similar types of processes or part families. The goal is to identify a cell organization in such a way that the transportation of the different parts between cells is minimized. The organization of these cells is performed through cat swarm optimization, which is a recent swarm metaheuristic technique based on the behavior of cats. In that technique, cats have two modes of behavior: seeking mode and tracing mode, selected from a mixture ratio. For experimental purposes, a version of the autonomous search algorithm was developed with dynamic mixture ratios. The experimental results for both normal binary cat

swarm optimization (BCSO) and autonomous search BCSO reach all global optimums, both for a set of 90 instances with known optima and for a set of 35 new instances with 13 known optima.

C. Zhang et al. proposed the ionic liquid gel (ILG), a new type of soft actuator material, which is a mixture of 1-butyl-3-methylimidazolium tetrafluoroborate (BMIMBF4), hydroxyethyl methacrylate (HEMA), diethoxyacetophenone (DEAP), and ZrO_2 polymerized into a gel state under ultraviolet (UV) light irradiation. The soft actuator structure consists of a layer of ionic liquid polymer gel sandwiched between two layers of activated carbon capped with gold foil. The volume of the cationic BMIM+ in the ionic liquid BMIMBF4 is much larger than that of the anionic BF_4^-. When voltages are applied to both sides of the actuator, the anions and cations move toward the anode and cathode of the electrode, respectively, under the electric field. The volume of the ILG cathode side therefore expands and the volume of the ILG anode side shrinks, hence bending the entire actuator toward the anode side. The Ogden model was selected as the hyperelastic constitutive model to study the mechanical properties of the ILG by nonlinear analysis. As the ILG is an ideal material for the preparation of a supercapacitor, the equivalent circuit of the ILG can be modeled by the supercapacitor theory to identify the transfer function of the soft actuator. The central pattern generator (CPG) control is widely used in the area of biology, and CPGs based on bioinspired control methods have attracted great attention from researchers worldwide. After the continuum soft actuator is discretized, the CPG-based bioinspired method can be used to control the soft robot drivers. According to the simulation analysis results, the soft actuator can be smooth enough to reach the specified location.

Q. Zhu et al. proposed a novel motion planning method for autonomous ground mobile robot to address dynamic surroundings, nonlinear program, and robust optimization problems. A planner based on recurrent fuzzy neural network (RFNN) is designed to program trajectory and motion of mobile robot to reach target. And, obstacle avoidance is achieved. In RFNN, inference capability of fuzzy logic and learning capability of neural network are combined to improve nonlinear programming performance. Recurrent frame with self-feedback loops in RFNN enhances stability and robustness of the structure. Extended Kalman filter (EKF) is designed to train weights of RFNN considering kinematic constraint of autonomous mobile robot as well as target and obstacle constraints. EKF's characteristics of fast convergence and little limit in training data make it suitable to train the weights in real time. Convergence of the training process is also analyzed in this paper. Optimization technique and update strategy are designed to improve robust optimization of the system in dynamic surroundings. Simulation experiment and hardware experiment are implemented to prove effectiveness of the proposed method. Hardware experiment is carried out on a tracked mobile robot. Omnidirectional vision is used to locate the robot in the surroundings. Forecast improvement of the proposed method is then discussed at the end.

Finally, C. Sepúlveda et al. proposed a population pharmacokinetic (PopPK) model allowing the researchers to predict and analyze the drug behavior in a population of individuals and to quantify the different sources of variability among these individuals. In the development of PopPK models, the most frequently used method is the nonlinear mixed effect model (NLME). However, once the PopPK model has been developed, it is necessary to determine if the selected model is the best one of the developed models during the population pharmacokinetic study, and this sometimes becomes a multiple criteria decision making (MCDM) problem; frequently, researchers use statistical evaluation criteria to choose the final PopPK model. The use of the evaluation criteria mentioned above entails big problems since the selection of the best model becomes susceptible to the human error mainly by misinterpretation of the results. To solve the previous problems, we introduce the development of a software robot that can automate the task of selecting the best PopPK model considering the knowledge of human expertise. The software robot is a fuzzy expert system that provides a method to systematically perform evaluations on a set of candidate PopPK models of commonly used statistical criteria. The presented results strengthen our hypothesis that the software robot can be successfully used to evaluate PopPK models, ensuring the selection of the best PopPK model.

Conflicts of Interest

The editors declare that there are no conflicts of interest of any kind regarding the publication of this issue.

Fevrier Valdez
Oscar Castillo
Amita Jain
Dipak K. Jana

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 4182639, 13 pages
https://doi.org/10.1155/2019/4182639

Research Article

Evolutionary Spiking Neural Networks for Solving Supervised Classification Problems

G. López-Vázquez,[1] **M. Ornelas-Rodriguez**,[1] **A. Espinal**,[2] **J. A. Soria-Alcaraz**,[2] **A. Rojas-Domínguez**,[1] **H. J. Puga-Soberanes**,[1] **J. M. Carpio**,[1] **and H. Rostro-Gonzalez**[3]

[1]*Postgraduate Studies and Research Division, National Technology of Mexico, León Institute of Technology, León, Guanajuato, Mexico*
[2]*Department of Organizational Studies, DCEA-University of Guanajuato, Guanajuato, Guanajuato, Mexico*
[3]*Department of Electronics, DICIS-University of Guanajuato, Salamanca, Guanajuato, Mexico*

Correspondence should be addressed to A. Espinal; aespinal@ugto.mx

Received 9 November 2018; Revised 15 January 2019; Accepted 31 January 2019; Published 28 March 2019

Academic Editor: Oscar Castillo

Copyright © 2019 G. López-Vázquez et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a grammatical evolution (GE)-based methodology to automatically design third generation artificial neural networks (ANNs), also known as spiking neural networks (SNNs), for solving supervised classification problems. The proposal performs the SNN design by exploring the search space of three-layered feedforward topologies with configured synaptic connections (weights and delays) so that no explicit training is carried out. Besides, the designed SNNs have partial connections between input and hidden layers which may contribute to avoid redundancies and reduce the dimensionality of input feature vectors. The proposal was tested on several well-known benchmark datasets from the UCI repository and statistically compared against a similar design methodology for second generation ANNs and an adapted version of that methodology for SNNs; also, the results of the two methodologies and the proposed one were improved by changing the fitness function in the design process. The proposed methodology shows competitive and consistent results, and the statistical tests support the conclusion that the designs produced by the proposal perform better than those produced by other methodologies.

1. Introduction

Artificial neural networks (ANNs) have been successfully used in theoretical and practical fields to solve several kinds of problems (e.g., classification [1, 2], robotic locomotion [3, 4], and function approximation [5, 6]). Basically, ANNs are characterized by *computing units* which are interconnected through *communication links* that serve to send and/or receive *messages of some data type* [7]; these elements define what is known as their architecture or topology. There can be distinguished three generations of ANNs according to their computing units [8], which are capable to solve problems of digital (ANNs from 1st to 3rd generation), analogical (ANNs from 2nd to 3rd generation), and spatiotemporal (ANNs from 3rd generation) nature. The first generation is based on threshold units such as McCulloch–Pitts neurons [9] or perceptrons [10]. The second generation is based on computing units that apply continuous activation functions (e.g., sigmoid or hyperbolic tangent functions); ANNs of this generation can be trained with gradient descent-based algorithms such as the backpropagation learning rule [11]. The third generation is based on spiking neurons (see [12] for a detailed reference) such as integrate and fire model [13] or Hodgkin–Huxley neuron [14]; ANNs of this generation are known as spiking neural networks (SNNs), and these are the kinds of ANNs worked in this paper.

Usually, the implementation of an ANN to solve a specific problem, regardless the generation it belongs to, requires of human experts who define the ANN's topological elements, the learning rule, and its parameters, among other

design criteria. The experts perform such a design process either empirically, following some rule of thumb or by trial and error; this is due because there is a lack of a well-established methodology to set up the ANN design for a given problem. It is well-known that the good performance of ANNs is strongly related to their design and related criteria; thus, design of an ANN may stand for a challenge. Several studies have explored the learnability issues of ANNs related to their design; for example, combinatorial problems that arise related to the design of feedforward ANNs [15] or the problems that ANNs with fixed architectures may face when learning a specific problem [7, 16–19]. Insights have been given to ease or enhance learnability of ANNs, for example, by applying constraints to the task to be learned or to the ANN's architecture [7, 15]. As an example of constraints applied to the ANNs' architecture, partially connected ANNs have shown equal or better performance than their fully connected version; among other interesting benefits, there are reduction of the network complexity and its training and recall times [20]. Another insight is to develop algorithms capable of changing the architecture of an ANN during the learning process [7, 15].

Nowadays, evolutionary artificial neural networks (EANNs) are a special class of ANNs which are the result of using evolutionary algorithms (EAs), or other kinds of metaheuristic methods, for adapting the design of ANNs according to a specific task or problem; this is achieved by optimizing one or several of their design criteria (also the term *Neuroevolution* has been used to refer to this kind of design method). Thus, the EANNs, in some manner, allow us to avoid or overcome the learnability issues related to ANN architectures and to prescind, partially or completely, of human experts (see [21–24] for comprehensive reviews). There are four main approaches of deploying EANNs [25] by means of weight optimization [26–28], topology structure optimization [25, 29–31], weight and topology structure optimization [32–38], and learning rule optimization [39, 40]. Most of the work made on EANNs is focused on deploying ANNs from the first and second generations.

Recently, efforts to use SNNs for solving real problems from engineering and industry are increasing because of interesting characteristics of spiking neurons, such as their greater computational power than that of less plausible neuron models and SNNs can solve problems with fewer computing units than those of ANNs from previous generations [19, 41]. Although there are learning rules to adapt parameters of SNNs, such as SpikeProp [42], the use of metaheurstic algorithms is a common practice to adapt their parameters or define design criteria because they overcome drawbacks of such learning rules [43] and allow us to handle the greater variety of design criteria (parameters of neuron models and synapses, types of synapses, topology's wiring patterns, encoding scheme, etc.) that these kinds of ANNs present; in this work, the combination of SNN and metaheuristic algorithms is referred as evolutionary spiking neural networks (ESNNs). In [44–48], the synaptic weights of a single spiking neuron, e.g., integrate and fire model [13] or Izhikevich model [49], are calibrated by means of algorithms such as differential evolution (DE) [50], particle swarm optimization (PSO) [51], cuckoo search algorithm (CSA) [52], or genetic algorithm (GA) [53] to perform classification tasks; the spiking neuron performs the classification by using the firing rate encoding scheme as the similarity criterion in order to assign the class to which an input pattern belongs. Other works, in [43, 54–57], three-layered feedforward SNNs with synaptic connections were implemented, which are formed by a weight and a delay, to solve supervised classification problems through the use of time-to-first-spike as a classification criterion; in these works, the training has been carried out by means of evolutionary strategy (ES) [58, 59] and PSO algorithms. An extension of previous works is made in [60, 61], where the number of hidden layers and their computing units are defined by grammatical evolution (GE) [62] besides the metaheuristic learning. More complex SNN frameworks have been developed and trained with metaheuristics (such as ES) to perform tasks such as visual pattern recognition, audio-visual pattern recognition, taste recognition, ecological modelling, sign language recognition, object movement recognition, and EEG spatio/spectrotemporal pattern recognition (see [63] for a review of these frameworks). The robotic locomotion is solved through SNNs designed by metaheuristics in [60, 64, 65]; in these works, both the connectivity pattern and synaptic weights of each Belson–Mazet–Soula (BMS) [66] neuron model into SNNs called spiking central pattern generators (SCPGs) are defined through GE or Christiansen grammar evolution (CGE) [67] algorithms; all individual designs are integrated to define the SCPGs that allow the locomotion of legged robots.

The present paper proposes a design methodology for three-layered feedforward ANNs of the third generation for solving supervised classification problems. The design methodology incorporates partial connectivity between input and hidden layers, which contribute to reduce the topological complexity of the ESNNs; in addition, partial connectivity may also contribute to reduce the number of features of the input vector, thus indirectly performing dimensionality reduction. The proposal explores the search space of three-layered feedforward topologies with configured synaptic connections; thus an explicit learning process is not required. This kind of design methodology has been previously proposed for ANNs from first and second generations, and they can be considered as a design of composed functions. To the best of the authors' knowledge, this is the first attempt to perform the design of SNNs that define the number of computing units and their configured connectivity patterns (weights and delays). The rest of the paper is organized as follows: Section 2 explains the proposed methodology and its constituent methods. The experimental configuration of the proposal and other methodologies used for comparison and their results are in Section 3. In Section 4, the results of the proposed methodology are statistically compared to those of other methodologies. Finally, Section 5 contains the conclusion of the paper and future work based on it.

2. Design Methodology and Concepts

This paper proposes a framework to design partially connected spiking neural networks (SNNs) for solving

supervised classification problems. Such proposed framework requires the following elements: a *temporal encoding* scheme to transform original input data into a suitable form for the network; a context-free grammar in Backus–Naur form (BNF grammar) to guide the generation of neural network words and a *mapping process* to transform genotype of individuals into functional network designs; a *fitness function* and a *target* definition to determine the performance of proposed networks, and a *search engine* to optimize the solutions. A general diagram of the methodology can be seen in Figure 1.

2.1. Spiking Neural Networks. The spiking neural networks (*SNNs*) constitute the third generation of *ANNs* because of the inclusion of the firing time component in their computation process [8].

2.1.1. Spike Response Model. The spike response model (SRM) is employed in this framework as basis for the SNN. The SRM fires (i.e., produces a spike) whenever the state of its membrane potential surpasses the firing threshold (θ). In the SRM, its membrane potential is calculated through time as a linear summation of postsynaptic potentials (PSPs) (excitatories and/or inhibitories), which are caused by impinging spikes arriving to a neuron through its presynaptic connections (Figure 2); each PSP is weighted and delayed by its synaptic connection.

The membrane potential x of neuron j at time t is calculated as the weighted (w_{ji}) summation of contributions ($y_i(t)$) from its connecting presynapses (Γ_j), as in the following equation:

$$x_j(t) = \sum_{i \in \Gamma_j} w_{ji} y_i(t). \tag{1}$$

The unweighted contribution $y_i(t)$ is described by equation (2), in which the function $\varepsilon(t - t_i - d_{ji})$ describes a form of the PSPs generated by impinging spikes coming from the presynaptic neuron i at the simulation time (t). The parameters of the presynaptic connection i are: the firing time t_i and synaptic delay d_{ji}:

$$y_i(t) = \varepsilon\left(t - t_i - d_{ji}\right). \tag{2}$$

The spike response function $\varepsilon(t)$ describes the form of PSPs, and it is defined in the following equation, where τ represents the membrane potential time constant that defines the decay time of the postsynaptic potential:

$$\varepsilon(t) = \begin{cases} \dfrac{t}{\tau} e^{1 - t/\tau}, & \text{if } t > 0, \\ 0, & \text{else.} \end{cases} \tag{3}$$

2.2. Temporal Encoding. Due to the nature of the employed neural model, original features from the dataset must be transformed into spikes prior to introducing them into the network. For such purpose, the one-dimensional encoding in the following equation is employed [56]:

$$Y(f) = \left[\frac{(b-a)}{r} * f\right] + \left[\frac{(a * M) - (b * m)}{r}\right], \tag{4}$$

where Y is the spike temporal value, f is the original feature value, $[a, b]$ are the lower and upper temporal interval limits of the encoding, whereas M and m hold the maximum and minimum values that the f variable takes, respectively, and r is the range between M and m. This encoding method preserves the dimension of the samples in the dataset, while providing a temporal representation of the scalar values of the dataset suitable for insertion in the network.

2.3. Grammatical Evolution (GE). Grammatical evolution is an evolutionary algorithm based on the combination of genetic algorithms and context-free grammars [68]. It employs a BNF grammar relating to the problem, a mapping process to obtain the functional form of solutions, and a search engine to drive the search process.

2.3.1. BNF Grammar. The *Backus–Naur form* (Figure 3) is employed to define the topology of the network and its parameters. Any word produced by this grammar includes an arbitrary number of hidden neurons and some specific *pre-* and *post*synapses with their respective parameters. The opening curled bracket symbol ({) indicates the division between hidden neurons, while the opening parenthesis (() marks the different synapses, and the *at* symbol (@) precedes the synapse-specific weight and delay values.

Figure 4 illustrates an example of a word generated by the proposed grammar and its corresponding network topology. By relating the word with its network topology, the word has two "{" symbols (see end of each row), implying that the network has two hidden neurons. In this case, each row has two "(" symbols meaning three synaptic configurations (but it can vary for each hidden neuron), where the first and second synaptic configurations represent connections with neurons from the input layer, and the last configuration marks the synapse with the output layer; each synaptic configuration is formed by a neuron identifier, a synaptic weight, and a delay. In Figure 4, each presynaptic neuron and its synaptic connection with a postsynaptic neuron are portrayed in the same color to clarify the reading of the transformation process from a word to a network topology.

2.3.2. Mapping Process. The mapping process transforms an individual from its genotypic form into its phenotypic form to represent a functional network. The depth-first mapping process—employed in this framework—is the standard in GE; basically, it begins by deriving (i.e., replace it by one of its productions) the left-most nonterminal symbol (initially, <*architecture*> non-terminal symbol) until all nonterminal symbols in depth are derived and then moves to the current left-most nonterminal. The process continues until either nonterminals are depleted, or all elements of the genotype have been used.

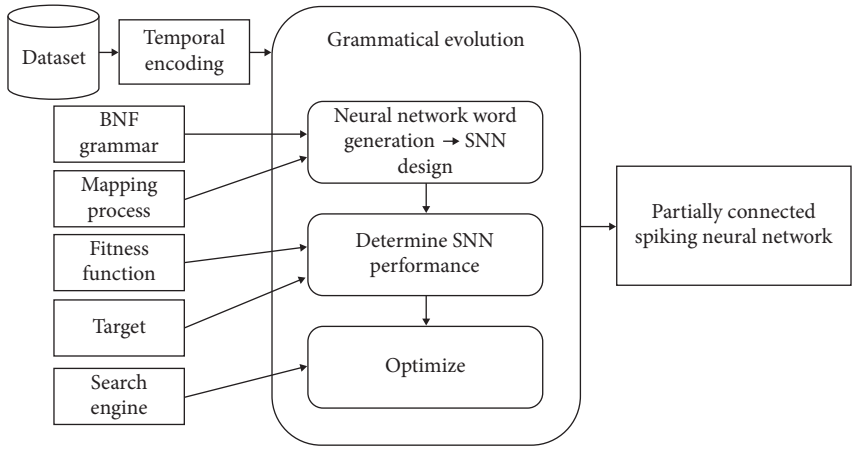


Figure 1: General diagram for the proposed framework.

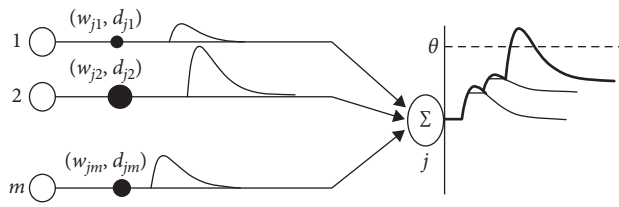


Figure 2: Membrane potential of neuron j: linear summation of PSPs [55].

⟨architecture⟩ ⊨ ⟨hiddenNeurons⟩
⟨hiddenNeurons⟩ ⊨ ⟨hiddenNeuron⟩ | ⟨hiddenNeuron⟩⟨hiddenNeurons⟩
⟨hiddenNeuron⟩ ⊨ ⟨preSynapses⟩⟨posSynapses⟩{
⟨preSynapses⟩ ⊨ ⟨preSynapse⟩ | ⟨preSynapse⟩⟨preSynapses⟩
⟨preSynapse⟩ ⊨ ⟨inputNeurons⟩@⟨synapse⟩(
⟨posSynapse⟩ ⊨ ⟨outputNeurons⟩@⟨synapse⟩
⟨inputNeurons⟩ ⊨ 0 | 1 | 2 | 3 ...
⟨outputNeurons⟩ ⊨ o
⟨synapse⟩ ⊨ ⟨weight⟩,⟨delay⟩
⟨weight⟩ ⊨ ⟨sign⟩⟨digit⟩⟨digit⟩⟨digit⟩.⟨digit⟩⟨nonZeroDigit⟩
⟨delay⟩ ⊨ ⟨delayBound⟩⟨digit⟩.⟨digit⟩⟨nonZeroDigit⟩
⟨sign⟩ ⊨ + | −
⟨delayBound⟩ ⊨ 0 | 1
⟨nonZeroDigit⟩ ⊨ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
⟨digit⟩ ⊨ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 3: Proposed BNF grammar for designing partially connected SNNs.

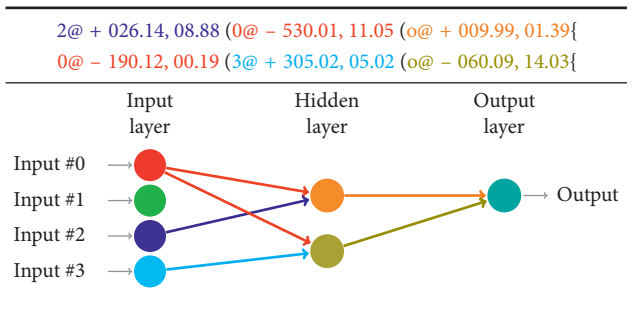


Figure 4: Example of a word generated by the proposed grammar and its corresponding network topology.

2.3.3. Search Engine. Several population-based metaheuristic algorithms can be used as the search engine of grammatical evolution. The well-known genetic algorithm (GA) and differential evolution (DE) are used in this framework [69].

2.4. Fitness Function. Two different fitness functions are considered to provide a measure of the ability of the solutions to solve the problem:

(1) The squared error is as defined in the following equation, where P is the total number of training

patterns, O is the number of neurons in the output layer, $t_o^a(p)$ is the actual firing time, and $t_o^d(p)$ is the desired firing time of neuron o:

$$E_s = \sum_{p}^{P} \sum_{o}^{O} \left(t_o^a(p) - t_o^d(p)\right)^2. \quad (5)$$

(2) The accuracy error of the training subset is as in the following equation, where C is the number of correct predictions and T is the total of predictions:

$$E_a = 1 - \left(\frac{C}{T}\right). \quad (6)$$

Both fitness functions are designed to be minimized.

2.5. Target. In order to obtain a prediction, a particular firing time is assigned to each class in the dataset employed, resulting in a desired time-to-first spike for every sample belonging to a specific class.

3. Experiments and Results

Twelve supervised classification benchmark datasets from the UCI Machine Learning Repository [70] were considered for experimentation: Balance Scale, Blood Transfusion Service Center (Blood), Breast Cancer Wisconsin (Breast Cancer), Japanese Credit Screening (Card), Pima Indians Diabetes (Diabetes), Fertility, Glass Identification (Glass), Ionosphere, Iris Plant, Liver Disorders (Liver), Parkinson, and Wine. Table 1 shows the details of the datasets employed.

Each dataset was randomly divided into two subsets of approximately the same size, accounting for the instances of each class to be evenly distributed between the subsets. One of these subsets is assigned to be the design set, while the other is to be the test set.

Then, the design set is employed to carry out the GE, while the test set is reserved to prove the performance of the best solution provided by the evolutionary process.

Aiming to compare the performance between neural models from different generations in solving pattern recognition tasks, six different configurations were considered, as shown in Table 2, observing the following details:

(i) α configurations employ the parameters defined in [35], focusing on developing second-generation partially connected ANNs

(ii) β configurations aim to be an homology of α configurations but used to produce third-generation partially connected ANNs

(iii) γ configurations are defined as β configurations but employing DE as search engine instead of GA

Parameters between configurations were matched to make a comparison as fair as possible. Furthermore, configurations labeled with subscript 1 look upon the squared error as the fitness function to guide the evolutionary process, while configurations labeled with subscript 2 consider the accuracy error of the design set.

Table 1: Datasets employed for experimentation.

Dataset	Instances	Classes	Features
Balance Scale	625	3	4
Blood	748	2	4
Breast Cancer	683	2	9
Card	653	2	15
Diabetes	768	2	8
Fertility	100	2	9
Glass	214	6	9
Ionosphere	351	2	33
Iris Plant	150	3	4
Liver	345	2	6
Parkinson	195	2	22
Wine	178	3	13

Table 2: Configurations included in experimentation.

Configuration	Fitness function	Search engine
α_1	Squared error	GA
α_2	Accuracy error	GA
β_1	Squared error	GA
β_2	Accuracy error	GA
γ_1	Squared error	DE
γ_2	Accuracy error	DE

In order to guarantee statistical significance, the central limit theorem [71] is satisfied by performing 33 experiments for each configuration. Specific parameters used in this framework for configurations β and γ are provided next.

Temporal Encoding: The one-dimensional encoding scheme observes a temporal range from 0.01 to 9 milliseconds (ms).

SRM: *membrane potential time constant* $\tau = 9$; target: {12 ms, 15 ms, 18 ms, ...} (depending on the number of classes in the dataset); *simulation time* [10 ms, target of the last class plus two]; *threshold* $\theta = 1$ millivolts (mV); *weight range* $\in$ [−999.99, 999.99]; and *delay range* $\in$ [0.01, 19.99] (ms).

GA: binary search space [0, 1], *codon size* = 8; *individual dimension* = 4000 (500 codons); *population size* = 100; *function calls* = 1,000,000; K-tournament ($K = 5$) selection operator; *elitism percentage* = 10%; one-point crossover operator; *mutation:* bit-negator mutation operator (5%).

DE: real search space [0, 255]; *individual dimension* = 500; *function calls* = 1,000,000; *population size* = 100; *crossover rate* = 10%; *mutation*: DE/Rand/1.

Tables 3 and 4 show the results obtained by carrying out the aforementioned methodology. Accuracy value $\in$ [0, 1] grades the average performance of the configurations applied to classify specific datasets, along with its corresponding standard deviation, for all experiments made. Design accuracy relates with the performance of the best network topology obtained by the evolutionary algorithm,

Table 3: Accuracy of design and testing on every configuration for Balance Scale, Blood, Breast Cancer, Card, Diabetes, and Fertility datasets.

Dataset	Configuration	Design accuracy	Test accuracy
Balance Scale	α_1	0.7486 ± 0.0525	0.7219 ± 0.0629
	α_2	0.8354 ± 0.0460	0.8077 ± 0.0582
	β_1	0.7331 ± 0.0718	0.6944 ± 0.0752
	β_2	0.8363 ± 0.0272	0.8078 ± 0.0427
	γ_1	0.8528 ± 0.0197	0.8346 ± 0.0261
	γ_2	**0.8960 ± 0.0062**	**0.8647 ± 0.0134**
Blood	α_1	0.7711 ± 0.0112	0.7622 ± 0.0117
	α_2	**0.8010 ± 0.0135**	**0.7731 ± 0.0168**
	β_1	0.7747 ± 0.0110	0.7607 ± 0.0138
	β_2	0.7863 ± 0.0162	0.7684 ± 0.0120
	γ_1	0.7760 ± 0.0076	0.7618 ± 0.0088
	γ_2	0.7957 ± 0.0145	0.7685 ± 0.0155
Breast Cancer	α_1	0.9494 ± 0.0141	0.9418 ± 0.0238
	α_2	**0.9781 ± 0.0073**	**0.9585 ± 0.0077**
	β_1	0.9474 ± 0.0121	0.9405 ± 0.0151
	β_2	0.9677 ± 0.0073	0.9432 ± 0.0142
	γ_1	0.9574 ± 0.0111	0.9384 ± 0.0140
	γ_2	0.9749 ± 0.0062	0.9478 ± 0.0117
Card	α_1	0.8624 ± 0.0099	**0.8641 ± 0.0140**
	α_2	0.8779 ± 0.0160	0.8591 ± 0.0174
	β_1	0.8561 ± 0.0502	0.8524 ± 0.0527
	β_2	0.8814 ± 0.0137	0.8561 ± 0.0153
	γ_1	0.8740 ± 0.0134	0.8596 ± 0.0197
	γ_2	**0.8879 ± 0.0120**	0.8535 ± 0.0166
Diabetes	α_1	0.7506 ± 0.0231	0.7457 ± 0.0207
	α_2	0.7843 ± 0.0156	0.7476 ± 0.0224
	β_1	0.7570 ± 0.0151	0.7490 ± 0.0215
	β_2	0.7780 ± 0.0143	**0.7477 ± 0.0126**
	γ_1	0.7810 ± 0.0153	0.7370 ± 0.0152
	γ_2	**0.7902 ± 0.0134**	0.7389 ± 0.0205
Fertility	α_1	0.8988 ± 0.0256	0.8218 ± 0.0616
	α_2	0.9297 ± 0.0204	0.8170 ± 0.0551
	β_1	0.9255 ± 0.0243	0.8309 ± 0.0459
	β_2	0.9182 ± 0.0222	0.8279 ± 0.0467
	γ_1	**0.9455 ± 0.0199**	**0.8479 ± 0.0462**
	γ_2	0.9370 ± 0.0131	0.8236 ± 0.0484

Table 4: Accuracy of design and testing on every configuration for Glass, Ionosphere, Iris Plant, Liver, Parkinson, and Wine datasets.

Dataset	Configuration	Design accuracy	Test accuracy
Glass	α_1	0.2549 ± 0.1345	0.2404 ± 0.1433
	α_2	0.6035 ± 0.0448	0.5374 ± 0.0688
	β_1	0.4288 ± 0.0673	0.4002 ± 0.0590
	β_2	0.6641 ± 0.0255	0.5947 ± 0.0436
	γ_1	0.4895 ± 0.0574	0.4351 ± 0.0476
	γ_2	**0.7126 ± 0.0190**	**0.6186 ± 0.0413**
Ionosphere	α_1	0.8549 ± 0.0374	0.8374 ± 0.0295
	α_2	0.9158 ± 0.0217	0.8669 ± 0.0267
	β_1	0.8543 ± 0.0537	0.8137 ± 0.0708
	β_2	0.9190 ± 0.0284	0.8724 ± 0.0241
	γ_1	0.9351 ± 0.0182	0.8907 ± 0.0240
	γ_2	**0.9616 ± 0.0113**	**0.9015 ± 0.0201**
Iris Plant	α_1	0.8857 ± 0.1111	0.8663 ± 0.1269
	α_2	0.9653 ± 0.0161	**0.9386 ± 0.0210**
	β_1	0.9733 ± 0.0157	0.9382 ± 0.0217
	β_2	0.9859 ± 0.0109	0.9362 ± 0.0163
	γ_1	0.9794 ± 0.0123	0.9325 ± 0.0164
	γ_2	**0.9923 ± 0.0074**	0.9358 ± 0.0261
Liver	α_1	0.6820 ± 0.0406	0.6462 ± 0.0536
	α_2	0.7352 ± 0.0245	0.6660 ± 0.0356
	β_1	0.6834 ± 0.0481	0.6304 ± 0.0476
	β_2	0.7461 ± 0.0224	0.6632 ± 0.0394
	γ_1	0.7472 ± 0.0183	**0.6723 ± 0.0302**
	γ_2	**0.7636 ± 0.0196**	0.6612 ± 0.0295
Parkinson	α_1	0.8719 ± 0.0395	0.8281 ± 0.0568
	α_2	0.9080 ± 0.0159	**0.8596 ± 0.0285**
	β_1	0.8563 ± 0.0264	0.8033 ± 0.0519
	β_2	0.8953 ± 0.0205	0.8503 ± 0.0353
	γ_1	0.9025 ± 0.0266	0.8380 ± 0.0387
	γ_2	**0.9200 ± 0.0172**	0.8494 ± 0.0377
Wine	α_1	0.6881 ± 0.1549	0.6415 ± 0.1551
	α_2	0.9063 ± 0.0441	0.8384 ± 0.0606
	β_1	0.7375 ± 0.1126	0.6816 ± 0.1098
	β_2	0.8895 ± 0.0641	0.7855 ± 0.0686
	γ_1	0.9318 ± 0.0285	0.8620 ± 0.0491
	γ_2	**0.9638 ± 0.0164**	**0.8684 ± 0.0458**

whilst test accuracy indicates the performance of such network applied to the test subset; highest values are indicated in boldface.

As well, Tables 5 and 6 show some of the features of the generated topologies, focusing on the average amount of input vector features actually employed by the networks, and its corresponding rate regarding the total size of the original input vector; besides, the average number of hidden units and synapses present in the generated networks. In Supplementary Materials, some examples of SNNs' topologies with best obtained results are shown; each example contains the benchmark dataset, used configuration, accuracies of design and test phases, the generated word, and the network topology.

4. Comparative Statistical Analysis

As detailed in the previous section, data samples from performing thirty-three independent experiments for each configuration on every dataset were obtained. Thereupon, several statistical tests [72] were applied to these data. First of all, a Shapiro–Wilk [73] test was applied to determine the normality of the samples. Such test showed that data can indeed be modelled under normal distributions. Further analysis was divided into three tests applied to configurations using squared error as fitness function, configurations using accuracy error as fitness function, and all configurations.

4.1. Test of Designs Driven by Squared Error Fitness Function. In order to verify statistical significance of the results, analysis of variance (ANOVA [74]) tests were applied to determine if, firstly, implementing different methodologies to develop weighed network topologies impacts on the accuracy of classification and secondly, to identify which of these methodologies offers the best performance. Table 7 shows the results obtained by two-way ANOVA test, observing as independent variables both configurations and datasets.

Table 5: Topology characteristics for every dataset on configurations α_1, β_1, and γ_1.

Configuration	Dataset	Average number of features employed	Rate of used features	Average number of hidden units	Average number of synapses
α_1	Balance Scale	2.97 ± 0.67	0.74	1.58 ± 0.85	9.58 ± 2.89
	Blood	2.30 ± 0.76	0.58	1.85 ± 0.99	7.64 ± 2.24
	Breast Cancer	2.52 ± 0.99	0.28	2.03 ± 1.09	8.79 ± 2.04
	Card	2.52 ± 0.93	0.17	2.09 ± 0.93	8.67 ± 2.22
	Diabetes	2.09 ± 0.90	0.26	1.97 ± 1.11	7.09 ± 1.91
	Fertility	3.03 ± 0.97	0.34	2.30 ± 1.29	9.24 ± 2.87
	Glass	2.21 ± 1.32	0.25	1.48 ± 0.93	11.73 ± 3.77
	Ionosphere	2.24 ± 1.05	0.07	2.21 ± 1.30	7.61 ± 2.81
	Iris Plant	1.30 ± 0.52	0.33	1.70 ± 1.00	8.18 ± 2.35
	Liver	2.48 ± 0.70	0.41	1.88 ± 0.84	6.64 ± 1.79
	Parkinson	2.27 ± 1.38	0.10	1.85 ± 1.50	8.00 ± 4.65
	Wine	2.48 ± 1.23	0.19	1.97 ± 1.75	9.73 ± 4.48
β_1	Balance Scale	3.52 ± 0.56	0.88	4.09 ± 1.58	12.36 ± 4.32
	Blood	3.12 ± 0.69	0.78	4.76 ± 2.55	12.03 ± 5.77
	Breast Cancer	6.12 ± 1.47	0.68	4.03 ± 1.85	14.24 ± 5.46
	Card	5.79 ± 1.95	0.39	4.39 ± 2.09	12.85 ± 5.47
	Diabetes	3.39 ± 0.95	0.42	3.55 ± 1.71	09.36 ± 4.00
	Fertility	5.58 ± 1.74	0.62	4.09 ± 2.22	12.39 ± 6.02
	Glass	5.70 ± 1.62	0.63	3.03 ± 1.82	11.88 ± 6.30
	Ionosphere	5.52 ± 2.24	0.17	3.55 ± 2.05	12.15 ± 6.68
	Iris Plant	3.39 ± 0.89	0.85	4.73 ± 2.60	13.27 ± 6.93
	Liver	3.94 ± 1.07	0.66	4.06 ± 1.91	11.03 ± 5.32
	Parkinson	5.12 ± 1.77	0.23	4.36 ± 2.45	11.48 ± 5.66
	Wine	6.15 ± 1.88	0.47	3.82 ± 1.49	13.03 ± 4.93
γ_1	Balance Scale	4.00 ± 0.00	1.00	4.03 ± 1.64	12.79 ± 3.75
	Blood	3.79 ± 0.48	0.95	4.70 ± 1.71	13.42 ± 4.23
	Breast Cancer	6.70 ± 1.38	0.74	5.39 ± 2.39	16.30 ± 5.93
	Card	8.76 ± 2.85	0.58	5.12 ± 2.04	18.03 ± 8.12
	Diabetes	6.24 ± 1.33	0.78	5.67 ± 3.19	17.76 ± 9.36
	Fertility	7.24 ± 1.18	0.80	4.79 ± 2.42	16.79 ± 6.20
	Glass	6.79 ± 1.32	0.75	4.97 ± 2.18	16.24 ± 5.85
	Ionosphere	11.64 ± 3.31	0.35	4.45 ± 1.86	18.03 ± 6.07
	Iris Plant	3.91 ± 0.29	0.98	5.79 ± 2.79	15.73 ± 6.89
	Liver	5.03 ± 1.03	0.84	4.64 ± 2.36	14.39 ± 6.49
	Parkinson	8.91 ± 3.41	0.40	4.18 ± 2.37	15.06 ± 7.58
	Wine	8.64 ± 1.92	0.66	4.97 ± 1.62	16.97 ± 5.33

ANOVA's null hypothesis (H_0) dictates that observed samples come from one unique normal distribution. As p values (Pr(>F) in Table 7) are smaller than the significance value of 0.05, there is not enough evidence to accept H_0, ergo rejecting that samples are statistically similar. In other words, it can be conclude that configurations come from different distributions. This test provides relevant statistical evidence to support the conclusion that changing the methodology while generating weighted topologies influences the classification accuracy of the networks.

Pairwise t-tests and *Tukey HSD* [75] tests were applied next. As in the ANOVA test, the null hypothesis in both tests assumes that samples come from a single distribution. Table 7 shows t-test p values with a Bonferroni correction. Based on these results, it can be inferred that, with statistical significance, γ_1 configuration can be considered different from β_1 and α_1 configurations, based on a significance level of 0.05. Subsequently, Tukey HSD test results can be found in Table 7, to uphold that γ_1 configuration is significantly different from the other configurations. Once the previous results were found, a higher performance for γ_1 configuration is noticeable in the three left-most configurations shown in, e.g., Fertility (Figure 5), Glass (Figure 6), and Ionosphere (Figure 7), performance box plots.

4.2. Test of Designs Driven by Accuracy Error Fitness Function. Statistical analysis for configurations driven by accuracy error fitness function was performed with the same approach as in the previous subsection; Table 8 shows

Table 6: Topology characteristics for every dataset on configurations α_2, β_2, and γ_2.

Configuration	Dataset	Average number of features employed	Rate of used features	Average number of hidden units	Average number of synapses
α_2	Balance Scale	3.73 ± 0.45	0.93	2.06 ± 1.23	11.48 ± 3.47
	Blood	3.33 ± 0.77	0.83	2.52 ± 1.28	11.64 ± 3.95
	Breast Cancer	3.85 ± 1.10	0.43	2.06 ± 1.04	10.82 ± 3.93
	Card	5.39 ± 2.20	0.36	2.76 ± 1.54	14.42 ± 6.27
	Diabetes	3.21 ± 0.95	0.40	2.33 ± 1.22	10.64 ± 3.56
	Fertility	5.30 ± 1.59	0.59	2.94 ± 2.01	15.39 ± 7.72
	Glass	2.79 ± 1.22	0.31	1.88 ± 0.91	13.91 ± 3.70
	Ionosphere	4.45 ± 1.67	0.13	3.55 ± 1.42	14.18 ± 4.36
	Iris Plant	1.76 ± 1.05	0.44	2.00 ± 1.67	11.27 ± 6.49
	Liver	3.33 ± 0.94	0.56	1.67 ± 0.84	09.18 ± 3.02
	Parkinson	3.67 ± 2.22	0.17	1.91 ± 1.03	10.09 ± 4.43
	Wine	3.15 ± 1.52	0.24	2.67 ± 1.49	12.30 ± 5.26
β_2	Balance Scale	3.88 ± 0.33	0.97	3.39 ± 1.92	09.88 ± 4.16
	Blood	3.21 ± 0.69	0.80	4.21 ± 3.50	11.55 ± 9.49
	Breast Cancer	5.76 ± 1.63	0.64	3.97 ± 2.41	11.94 ± 6.09
	Card	6.94 ± 2.24	0.46	4.55 ± 2.85	14.00 ± 7.21
	Diabetes	4.06 ± 1.23	0.51	3.91 ± 2.11	10.85 ± 5.65
	Fertility	5.30 ± 2.11	0.59	3.39 ± 2.01	10.36 ± 5.64
	Glass	5.18 ± 1.45	0.58	3.73 ± 1.66	11.21 ± 4.44
	Ionosphere	6.48 ± 2.27	0.20	4.30 ± 2.67	12.94 ± 6.33
	Iris Plant	3.24 ± 0.74	0.81	4.09 ± 2.50	10.61 ± 6.30
	Liver	4.27 ± 1.21	0.71	3.52 ± 1.78	10.52 ± 4.39
	Parkinson	5.39 ± 2.44	0.25	3.09 ± 1.99	09.64 ± 5.51
	Wine	5.85 ± 1.73	0.45	4.24 ± 2.22	12.85 ± 5.21
γ_2	Balance Scale	4.00 ± 0.00	1.00	3.15 ± 2.12	10.55 ± 5.78
	Blood	3.67 ± 0.59	0.92	4.58 ± 2.56	12.79 ± 5.86
	Breast Cancer	7.30 ± 1.27	0.81	4.61 ± 1.82	15.85 ± 5.21
	Card	8.58 ± 2.45	0.57	4.00 ± 2.47	15.09 ± 7.10
	Diabetes	6.06 ± 1.41	0.76	4.24 ± 1.78	13.76 ± 4.96
	Fertility	6.73 ± 1.33	0.75	4.06 ± 2.73	14.21 ± 6.65
	Glass	6.70 ± 1.17	0.74	4.91 ± 1.60	15.06 ± 4.26
	Ionosphere	11.64 ± 4.14	0.35	3.88 ± 2.20	16.88 ± 7.10
	Iris Plant	3.61 ± 0.69	0.90	3.82 ± 2.18	10.97 ± 5.42
	Liver	4.91 ± 0.90	0.82	3.21 ± 1.53	10.45 ± 4.11
	Parkinson	7.97 ± 2.47	0.36	3.39 ± 1.50	12.36 ± 4.20
	Wine	8.15 ± 1.96	0.63	4.70 ± 1.93	16.12 ± 5.57

Table 7: Two-way ANOVA F, pairwise t-test, and Tukey HSD test with Bonferroni correction for squared error configurations.

ANOVA	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Configuration	2	0.7109	0.35547	73.217	**<2.2e-16**
Dataset	11	25.3929	2.30845	475.4710	**<2.2e-16**
Residuals	1174	5.6999	0.00486		
t-Tests	α_1		β_1		
β_1	**0.5936**		—		
γ_1	**1.9e-6**		**0.0006**		
Tukey HSD		diff	lwr	upr	p adj
β_1 vs. γ_1		0.014	0.0032	0.0544	**0.00**
α_1 vs. β_1		0.014	0.0030	0.026	0.0078
α_1 vs. γ_1		0.057	0.0460	0.0693	**0.00**

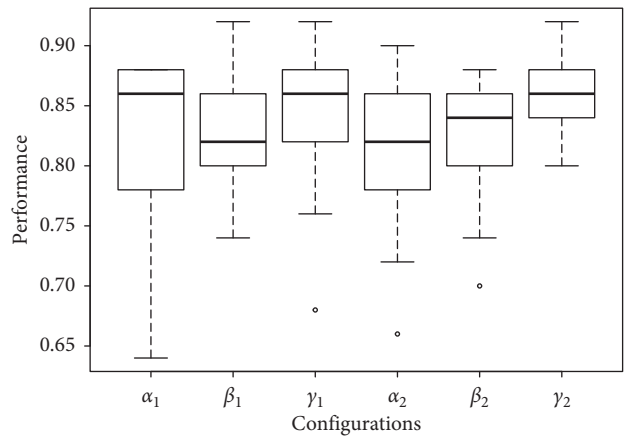


Figure 5: Box plots of the performance of all configurations on the Fertility dataset.

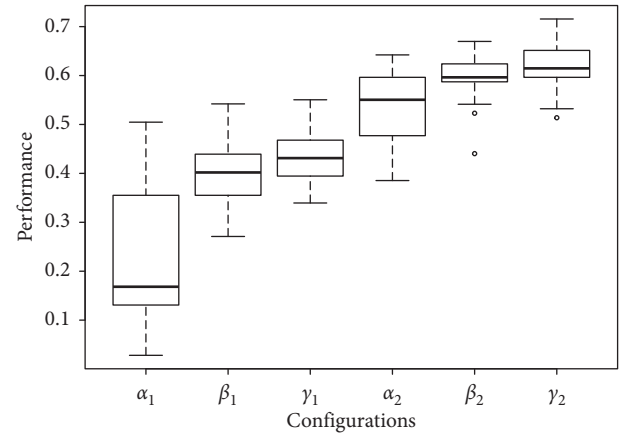


Figure 6: Box plots of the performance of all configurations on the Glass dataset.

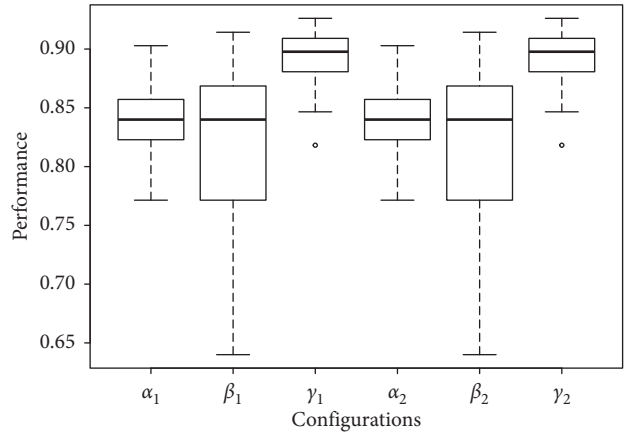


Figure 7: Box plots of the performance of all configurations on the Ionosphere dataset.

ANOVA, t-test, and Tukey HSD tests applied to such configurations. In this case, for designs driven by accuracy error fitness function, the pair-wise t-test show that there is not difference with statistical significance to reject the null hypothesis H_0 for α_2 and γ_2 configurations; however, the α_2 configuration requires a higher computational power to carry out the designing task due its search engine and its respective operators (crossover and mutation). The aforementioned issues are not presented for the γ_2 configuration; besides its results show a similar accuracy results with lower dispersion, this can be noticed in the right-most configurations shown in, e.g., Fertility (Figure 5), Glass (Figure 6), and Ionosphere (Figure 7) performance box plots, and this behavior was consistently observed for all benchmark datasets. The Tukey HSD test shows that there is statistical difference for all configurations; this, along with the observed behavior in the previous box plots, confirms that γ_2 configuration holds as the outperforming algorithm.

4.3. Test of All Configurations. An omnibus test was applied to the entire set of experiments considering both as independent variables, configurations and fitness functions. Two-way ANOVA test was applied to determine if varying both observed variables influences accuracy performance. Table 9 contains such results, providing statistical certainty to reject the null hypothesis H_0; in other words, the accuracy performance is affected by both variables. The p values lower than the significance level of 0.05 indicate that changing the optimization function (squared error and accuracy error) and the configuration does indeed affect the performance accuracy obtained by the generated topology.

Finally, pairwise t-test was applied to discern if, given two configurations, their performances are statistically similar. Considering p values in Table 9 and a significance value of 0.05, it can be inferred with statistically trustworthy that γ_2 configuration generally outperforms other configurations.

5. Conclusions and Future Work

This paper presents a GE-based methodology to design partially connected ANNs for solving supervised classification problems; some interesting characteristics of the methodology are that it provides weighted topologies which allow us to avoid an explicit training and those topologies exhibit partial connectivity between input and hidden layers which may avoid redundancies and reduce the dimensionality of the input feature vectors. The proposed methodology (γ_2) evolved from progressive improvements made to a base methodology (α_1), which uses GE with GA as search engine and squared error as fitness function; improvements were made by changing neuron models which allowed us to generate SNNs (β_1) instead of ANNs from the second ANN generation and by changing the search engine by using DE (γ_1) instead of GA. All the aforementioned configurations were adapted to use another fitness function based on the accuracy error of generated ANNs, so-called α_2, β_2, and γ_2.

In order to validate the achieved improvements, several statistics tests were applied. Each configuration was tested for twelve well-known benchmark datasets of supervised classification problems by performing 33 experiments for

Table 8: Two-way ANOVA F, pairwise t-test, and Tukey HSD test with Bonferroni correction for accuracy error configurations.

ANOVA	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Configuration	2	0.0769	0.0384	26.319	**$6.58e-12$**
Dataset	11	12.3300	1.1210	767.4670	**$<2.2e-16$**
Residuals	1174	1.7160	0.0014		
t-Tests	α_2		β_2		
β_2	**1.0000**		—		
γ_2	**0.1030**		**$1.3e-3$**		
Tukey HSD		diff	lwr	upr	p adj
β_2 vs. γ_2		0.0170	0.0110	0.0240	**0.0000**
α_2 vs. β_2		−0.0010	−0.0070	0.0050	0.8830
α_2 vs. γ_2		0.0100	0.0100	0.0220	**0.0001**

Table 9: Two-way ANOVA F and pairwise t-test with Bonferroni correction tests for all configurations.

ANOVA	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Fitness function	1	1.1220	1.1200	58.9180	**$2.36e-14$**
Configuration	4	0.7880	0.1960	10.3400	**$2.663e-08$**
Residuals	2370	45.1490	0.0190		
t-Tests	α_1	α_2	β_1	β_2	γ_1
α_2	0.0001	—	—	—	—
β_1	0.0100	0.00	—	—	—
β_2	0.0001	0.9900	0.0010	—	—
γ_1	0.0001	0.8900	0.0010	0.9680	—
γ_2	0.0000	0.0040	0.0000	0.0010	$4.8e-5$

each dataset. Three types of statistical analysis were performed, and the first being applied to α_1, β_1, and γ_1 configurations, which use squared error as the fitness function. In such analysis, γ_1 configuration is shown to outperform the other configurations based on the statistical test and graphic box plots. The second analysis focused α_2, β_2, and γ_2 configurations, which use the accuracy error as the fitness function; based on the Tukey HSD test, this analysis yielded a similar conclusion as from the first analysis, but with respect to γ_2 configuration. The last analysis compared all configurations and showed statistical evidence to support that γ_2 is a better configuration with competitive performances and lower dispersions for its designs.

Focusing in topology designs and performance results, evolutionary designs led to the formulation of solution topologies with fewer connections than those in equivalent fully connected topologies, hence reducing the complexity of the networks and achieving good classification performances. The topology simplification provided a good network design (i.e., design accuracy was competent), but it was desirable to get better generalization capability for unseen data in the test phase; some particular cases exhibited lower test accuracies, evidencing an improving opportunity.

Due to the flexibility of the context-free grammars employed in GE, another aspect of neural network topologies can be considered to cope with detected issues while preserving the enhancements accomplished. The design process may consider other traits, e.g., selection of the neural model and/or the search engine, specification of the model parameters, or even aggregation on the number of hidden layers to design SNNs for deep learning topologies. Moreover, additional types of topologies with structures other than layered networks can be explored to be designed, such as those of reservoir computing or central pattern generators. Furthermore, another kind of grammar-based genetic programming algorithms can be used to add semantic to the design process, such as Christiansen grammar evolution [67].

Finally, contemplating the fitness function as another relevant aspect to produce enhanced designs, considerations can also be made to it: to minimize the amount of processing units in the hidden layer or to consider another evaluation measurements to comply with other kinds of problems; features in the fitness function may be treated as weighted mono-objective fitness function or by using algorithms such as the nondominated sorting genetic algorithm (NSGA) [76] with fitness functions with multiple objectives.

Data Availability

The supervised classification dataset benchmarks used to support the findings of this study have been taken from the UCI Machine Learning Repository of the University of California, Irvine (http://archive.ics.uci.edu/ml/datasets.html).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors wish to thank the National Technology of México and University of Guanajuato. A. Espinal wishes to thank SEP-PRODEP for the support provided to the Project

511-6/17-8074 "Diseño y entrenamiento de Redes Neuronales Artificiales mediante Algoritmos Evolutivos." G. López-Vázquez and A. Rojas-Domínguez thank the National Council of Science and Technology of México (CONACYT) for the support provided by means of the Scholarship for Postgraduate Studies (701071) and research grant CÁTEDRAS-2598, respectively. This work was supported by the CONACYT Project FC2016-1961 "Neurociencia Computacional: de la teoría al desarrollo de sistemas neuromórficos".

Supplementary Materials

Examples of the best results obtained for SNNs are shown; each example contains the benchmark dataset, used configuration, accuracies of design and test phases, the generated word, and the network topology. (*Supplementary Materials*)

References

[1] M. Markou and S. Singh, "Novelty detection: a review-part 2:," *Signal processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[2] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[4] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on CPG-inspired control models and system implementation," *IEEE Transactions OnNeural Networks and Learning Systems*, vol. 25, no. 3, pp. 441–456, 2014.

[5] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018.

[6] F. Scarselli and A. Chung Tsoi, "Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.

[7] J. S. Judd, *Neural Network Design and The Complexity of Learning. Neural Network Modeling and Connectionism Series*, MIT Press, Cambridge, MA, USA, 1990.

[8] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[9] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[10] F. Rosenblatt, *The Perceptron, A Perceiving And Recognizing Automaton (Project PARA)*, Cornell Aeronautical Laboratory, Buffalo, NY, USA, 1957.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[12] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, Cambridge, UK, 2002.

[13] L. Lapicque, "Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization," *Journal de Physiologie et de Pathologie Generalej*, vol. 9, pp. 620–635, 1907.

[14] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal Of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[15] E. Amaldi, E. Mayoraz, and D. de Werra, "A review of combinatorial problems arising in feedforward neural network design," *Discrete Applied Mathematics*, vol. 52, no. 2, pp. 111–138, 1994.

[16] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete," *Neural Networks*, vol. 5, no. 1, pp. 117–127, 1992.

[17] B. DasGupta, H. T. Siegelmann, and E. Sontag, "On the intractability of loading neural networks," in *Theoretical Advances in Neural Computation and Learning*, pp. 357–389, Springer, Boston, MA, USA, 1994.

[18] S. Judd, "On the complexity of loading shallow neural networks," *Journal of Complexity*, vol. 4, no. 3, pp. 177–192, 1988.

[19] W. Maass and M. Schmitt, "On the complexity of learning for spiking neurons with temporal coding," *Information and Computation*, vol. 153, no. 1, pp. 26–46, 1999.

[20] D. Elizondo and E. Fiesler, "A survey of partially connected neural networks," *International Journal of Neural Systems*, vol. 8, no. 5-6, pp. 535–558, 1997.

[21] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 251–260, 2013.

[22] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[23] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: a review of two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 97–116, 2017.

[24] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[25] M. Tayefeh, F. Taghiyareh, N. Forouzideh, and L. Caro, "Evolving artificial neural network structure using grammar encoding and colonial competitive algorithm," *Neural Computing and Applications*, vol. 22, no. 1, pp. 1–16, 2013.

[26] A. Espinal, M. Sotelo-Figueroa, J. A. Soria-Alcaraz et al., "Comparison of PSO and DE for training neural networks," in *Proceedings of 2013 12th Mexican International Conference on Artificial Intelligence*, pp. 83–87, Mexico City, Mexico, November 2013.

[27] A. K. Morales, "Non-standard norms in genetically trained neural networks," in *Proceedings of 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pp. 43–51, IEEE, San Antonio, TX, USA, May 2000.

[28] A. K. Morales, "Training neural networks using non-standard norms–preliminary results," in *Proceedings of Mexican International Conference on Artificial Intelligence*, pp. 350–364, Acapulco, Mexico, April 2000.

[29] E. Alba, J. Aldana, and J. M. Troya, "Full automatic ann design: a genetic approach," in *Proceedings of International Workshop on Artificial Neural Networks*, pp. 399–404, Springer, Sitges, Spain, June 1993.

[30] L. F. De Mingo Lopez, N. Gomez Blas, and A. Arteta, "The optimal combination: grammatical swarm, particle swarm optimization and neural networks," *Journal of Computational Science*, vol. 3, no. 1-2, pp. 46–55, 2012.

[31] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex systems*, vol. 4, no. 4, pp. 461–476, 1990.

[32] F. Ahmadizar, K. Soltanian, F. AkhlaghianTab, and I. Tsoulos, "Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 1–13, 2015.

[33] B. A. Garro, H. Sossa, and R. A. Vazquez, "Design of artificial neural networks using a modified particle swarm optimization algorithm," in *Proceedings of The 2009 International Joint Conference On Neural Networks, IJCNN'09*, pp. 2363–2370, IEEE Press, Atlanta, GA, USA, June 2009.

[34] B. A. Garro and R. A. Vázquez, "Designing artificial neural networks using particle swarm optimization algorithms," *Computational intelligence and neuroscience*, vol. 2015, Article ID 369298, 20 pages, 2015.

[35] O. Quiroz-Ramírez, A. Espinal, M. Ornelas-Rodríguez et al., "Partially-connected artificial neural networks developed by grammatical evolution for pattern recognition problems," in *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*, vol. 749, pp. 99–112, 2018.

[36] D. Rivero, J. Dorado, J. Rabuñal, and A. Pazos, "Generation and simplification of artificial neural networks by means of genetic programming," *Neurocomputing*, vol. 73, no. 16–18, pp. 3200–3223, 2010.

[37] W. Sheng, P. Shan, J. Mao, Y. Zheng, S. Chen, and Z. Wang, "An adaptive memetic algorithm with rank-based mutation for artificial neural network architecture optimization," *IEEE Access*, vol. 5, pp. 18895–18908, 2017.

[38] I. Tsoulos, D. Gavrilis, and E. Glavas, "Neural network construction and training using grammatical evolution," *Neurocomputing*, vol. 72, no. 1–3, pp. 269–277, 2008.

[39] J. Fontanari and R. Meir, "Evolving a learning algorithm for the binary perceptron," *Network: Computation in Neural Systems*, vol. 2, no. 4, pp. 353–359, 1991.

[40] H. B. Kim, S. H. Jung, T. G. Kim, and K. H. Park, "Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates," *Neurocomputing*, vol. 11, no. 1, pp. 101–106, 1996.

[41] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International Journal of Neural Systems*, vol. 19, no. 04, pp. 295–308, 2009.

[42] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1–4, pp. 17–37, 2002.

[43] A. Belatreche, *Biologically Inspired Neural Networks: Models, Learning, and Applications*, VDM Verlag, Saarbrücken, Germany, 2010.

[44] A. Cachón and R. A. Vázquez, "Tuning the parameters of an integrate and fire neuron via a genetic algorithm for solving pattern recognition problems," *Neurocomputing*, vol. 148, pp. 187–197, 2015.

[45] R. A. Vazquez, "Pattern recognition using spiking neurons and firing rates," in *Advances in Artificial Intelligence–IBERAMIA 2010. Lecture Notes in Computer Science*, vol. 6433, Springer, Berlin, Germany, 2010.

[46] R. A. Vazquez, "Training spiking neural models using cuckoo search algorithm," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 679–686, New Orleans, LA, USA, June 2011.

[47] R. A. Vazquez and A. Cachon, "Integrate and fire neurons and their application in pattern recognition," in *Proceedings of 7th International Conference on Electrical Engineering Computing Science and Automatic Control*, pp. 424–428, Tuxtla Gutierrez, Mexico, September 2010.

[48] R. A. Vázquez and B. A. Garro, "Training spiking neurons by means of particle swarm optimization," in *Advances in Swarm Intelligence. Lecture Notes in Computer Science*, pp. 242–249, Springer, Berlin, Germany, 2011.

[49] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[50] 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.

[51] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceediongs of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, November-December 1995.

[52] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with computers*, vol. 29, no. 1, pp. 17–35, 2013.

[53] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA, 1975.

[54] J. S. Altamirano, M. Ornelas, A. Espinal et al., "Comparing evolutionary strategy algorithms for training spiking neural networks," in *Advances in Pattern Recognition*, p. 9, 2015.

[55] A. Belatreche, L. P. Maguire, M. McGinnity, and Q. X. Wu, "An evolutionary strategy for supervised training of biologically plausible neural networks," in *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing (CINC)*, pp. 1524–1527, Cary, NC, USA, September 2003.

[56] A. Belatreche, L. P. Maguire, and T. M. McGinnity, "Advances in design and application of spiking neural networks," *Soft Computing*, vol. 11, no. 3, pp. 239–248, 2006.

[57] H. Shen, N. Liu, X. Li, and Q. Wang, "A cooperative method for supervised learning in spiking neural networks," in *Proceedings of 14th International Conference on Computer Supported Cooperative Work in Design*, pp. 22–26, IEEE, Shanghai, China, April 2010.

[58] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der biologischen Evolution. Problemata, 15*, Frommann-Holzboog Verlag, Stuttgart, Germany, 1973.

[59] H. P. Schwefel, *Numerische Optimierung Von Computer-Modellen Mittels der Evolutionsstrategie*, Vol. 1, Birkhäuser, Basel, Switzerland, 1977.

[60] A. Espinal, M. Carpio, M. Ornelas, H. Puga, P. Melin, and M. Sotelo-Figueroa, "Comparing metaheuristic algorithms on the training process of spiking neural networks," in *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, pp. 391–403, Springer, Cham, Switzerland, 2014.

[61] A. Espinal, M. Carpio, M. Ornelas, H. Puga, P. Melín, and M. Sotelo-Figueroa, "Developing architectures of spiking neural networks by using grammatical evolution based on evolutionary strategy," in *Proceedings of Mexican Conference on Pattern Recognition*, pp. 71–80, Springer, Cancun, Mexico, June 2014.

[62] C. Ryan, J. Collins, and M. O'Neill, "Grammatical evolution: evolving programs for an arbitrary language," in *Proceedings of Genetic Programming: First European Workshop, EuroGP'98*, pp. 83–96, Springer Berlin Heidelberg, Paris, France, April 1998.

[63] S. Schliebs, *Optimisation And Modelling Of Spiking Neural Networks: Enhancing Neural Information Processing Systems*

Through The Power Of Evolution, LAP Lambert Academic Publishing, Saarbrücken, Germany, 2010.

[64] A. Espinal, H. Rostro-Gonzalez, M. Carpio et al., "Quadrupedal robot locomotion: a biologically inspired approach and its hardware implementation," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 5615618, 13 pages, 2016.

[65] E. I. Guerra-Hernandez, A. Espinal, P. Batres-Mendoza, C. H. Garcia-Capulin, R. De J. Romero-Troncoso, and H. Rostro-Gonzalez, "A fpga-based neuromorphic locomotion system for multi-legged robots," *IEEE Access*, vol. 5, pp. 8301–8312, 2017.

[66] H. Soula, G. Beslon, and O. Mazet, "Spontaneous dynamics of asymmetric random recurrent spiking neural networks," *Neural Computation*, vol. 18, no. 1, pp. 60–79, 2006.

[67] A. Ortega, M. De La Cruz, and M. Alfonseca, "Christiansen grammar evolution: grammatical evolution with semantics," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 77–90, 2007.

[68] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, 2001.

[69] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, John Wiley & Sons, Hoboken, NJ, USA, 2009.

[70] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017, http://archive.ics.uci.edu/ml.

[71] B. V. Gnedenko and A. N. Kolmogorov, "Limit distributions for sums of independent random variables," in *PredelÊźnye raspredeleniiĺąa dliĺąa summ.English*, Addison-Wesley Pub. Co., Cambridge, MA, USA, 1954.

[72] J. A. Soria Alcaraz, G. Ochoa, M. Carpio, and H. Puga, "Evolvability metrics in adaptive operator selection," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 1327–1334, ACM, Vancouver, Canada, July 2014.

[73] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3-4, pp. 591–611, 1965.

[74] F. J. Anscombe, "The validity of comparative experiments," *Journal of the Royal Statistical Society. Series A (General)*, vol. 111, no. 3, pp. 181–211, 1948.

[75] D. C. Montgomery, *Design and analysis of experiments*, Wiley, Hoboken, NJ, USA, 2013.

[76] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 8256723, 11 pages
https://doi.org/10.1155/2019/8256723

Research Article

Motion Simulation of Ionic Liquid Gel Soft Actuators Based on CPG Control

Chenghong Zhang, Bin He, An Ding, Shoulin Xu, Zhipeng Wang, and Yanmin Zhou

College of Electronics and Information Engineering, Tongji University, No. 4800 Caoan Road, Shanghai 201804, China

Correspondence should be addressed to Bin He; hebin@tongji.edu.cn

Received 24 October 2018; Revised 19 January 2019; Accepted 7 February 2019; Published 26 February 2019

Guest Editor: Fevrier Valdez

Copyright © 2019 Chenghong Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The ionic liquid gel (ILG), a new type of soft actuator material, is a mixture of 1-butyl-3-methylimidazolium tetrafluoroborate ($BMIMBF_4$), hydroxyethyl methacrylate (HEMA), diethoxyacetophenone (DEAP), and ZrO_2 polymerized into a gel state under ultraviolet (UV) light irradiation. The soft actuator structure consists of a layer of ionic liquid polymer gel sandwiched between two layers of activated carbon capped with gold foil. The volume of the cationic $BMIM^+$ in the ionic liquid $BMIMBF_4$ is much larger than that of the anionic BF_4^-. When voltages are applied to both sides of the actuator, the anions and cations move toward the anode and cathode of the electrode, respectively, under the electric field. The volume of the ILG cathode side therefore expands, and the volume of the ILG anode side shrinks, hence bending the entire actuator toward the anode side. The Ogden model was selected as the hyperelastic constitutive model to study the mechanical properties of the ILG by nonlinear analysis. As the ILG is an ideal material for the preparation of a supercapacitor, the equivalent circuit of the ILG can be modeled by the supercapacitor theory to identify the transfer function of the soft actuator. The central pattern generator (CPG) control is widely used in the area of biology, and CPGs based on bioinspired control methods have attracted great attention from researchers worldwide. After the continuum soft actuator is discretized, the CPG-based bioinspired method can be used to control the soft robot drivers. According to the simulation analysis results, the soft actuator can be smooth enough to reach the specified location.

1. Introduction

Due to the large difference in structure between the soft robot and the traditional robot, the material used to manufacture a robot for traditional applications is quite different from the material used to manufacture a bionic robot. In the past few decades, new lightweight, high-performance materials have attracted the attention of soft robot researchers worldwide. The electroactive polymer (EAP) has been proven to be a proper smart material that meets the requirements of soft robot design. The application of EAP materials has become a popular topic this year. Bionic robots and equipment using materials such as soft actuators have been produced [1, 2]. Because the soft robot offers strong adaptability and low pressure impedance, it has wide application prospects in biology, medicine, and agriculture. Electrochemical actuators have been developed quickly over the past few decades due to their desired mechanical properties in intelligent robots [3–5]. Because ionic liquid gels (ILGs) offer chemical stability, thermal stability, and simple ion transport, they are suitable for the production of soft robot actuators [6–8].

Noncovalent interactions allow supramolecular gel materials to have a very high mechanical strength and excellent self-healing ability. It has been demonstrated and well documented that ZrO_2 can improve the electrochemical behavior of ionic liquids and the mechanical strength of ionic polymers in supramolecular nanocomposites [9].

The application of a bioinspired control method based on the central pattern generator (CPG) of neural oscillators to generate rhythmic motion has attracted the attention of some researchers [10–12]. The rhythmic movement of animals is achieved by the interaction between the rhythm signal generated from the CPG and the dynamics of the

musculoskeletal system [13–15]. Nonlinear oscillators have been widely used to model rhythmic motion-generating CPGs for robot control.

In this paper, a new type of ILG soft actuator is demonstrated. The soft actuator is composed of a middle layer of ionic liquid polymer gel sandwiched between two layers of activated carbon capped with gold foil, as illustrated in Figure 1. The ILG of the soft actuator was prepared, and its mechanical properties were described. The bending deformation principle of the soft actuator is discussed. The soft actuator is discretized to meet the conditions of the CPG control method, of which the effectiveness has been proven by numerical simulations. This paper provides a detailed theoretical analysis of ILG soft actuators based on the aspects of design, material, control, etc. This work has introduced a new research direction for the development of soft actuators and will contribute to the development of soft robots.

2. Structure Design of a Soft Robot Actuator

The new type of flexible actuator structure shown in Figure 1 is similar to the traditional EAP actuator. The white area is the ionic liquid polymer gel, which is the structural body of the actuator. The black sides are the activated carbon layers, which adsorb the anions and cations of the ionic liquid, respectively. The outermost yellow area is the gold foil, and each side of the gold foil is cut into 4 segments, each of which is connected to a power source.

The activated carbon layer is used to adsorb the anions and cations in the ionic liquid polymer gel, the gold foil layer is used as the electrode, and a wire is connected on the outside of the gold foil to the power source.

3. Ionic Liquid Gel

3.1. Ionic Liquid Gel Properties. In the experiment, the ILGs were composed of 1-butyl-3-methylimidazolium tetrafluoroborate ($BMIMBF_4$), hydroxyethyl methacrylate (HEMA), ZrO_2 nanoparticles, and 2,2- diethoxyacetophenone (DEAP), with masses of 890 mg, 68.6 mg, 30 mg, and 1.4 mg, respectively. A mixed solution of the ILGs was prepared in the ratio mentioned above, and then the mixed solution was stirred in a magnetic stirrer for 60 minutes or longer to form a suspension. The suspension was then placed under ultrahigh-intensity UV light generated by an ML-3500C Maxima-type cold light source and then polymerized to the gel state. The ML-3500C Maxima lamp has an ultraviolet intensity of 90,000 μWs/cm^2 (15″/380 mm distance) and a wavelength of 365 nm.

The morphological scanning analysis results of freeze-dried samples was obtained by scanning electron microscopy (SEM) and showed that the existence of porous microstructures in ionic gels is common. The internal liquid of the ILG was replaced with distilled water. After the freeze-drying treatment, a Hitachi S4800-type high-resolution field emission SEM was used to scan the section. Figure 2 shows the typical 3D porous structure of the ionic liquid carrier HEMA with a magnification of 5000. The polymerization of HEMA in the solution occurs under UV irradiation, and the

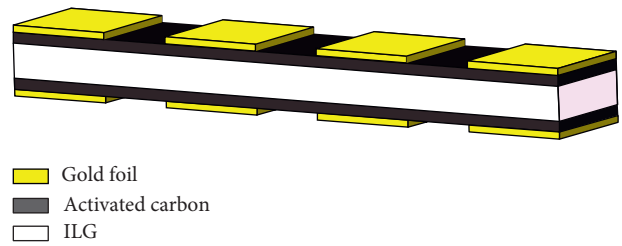


Figure 1: The 5-layer structure of the flexible actuator.

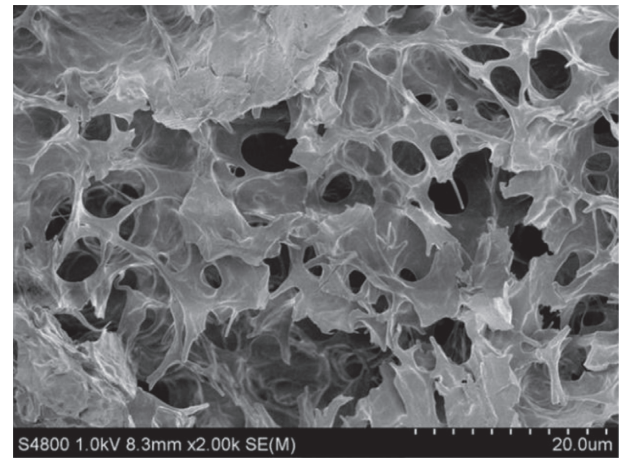


Figure 2: SEM image of a freeze-dried $BMIMBF_4$-based gel after the ionic liquid was replaced with water.

polymer matrix is crosslinked to form a porous network structure [15]. The crosslinked matrix forms a 3D framework, offering good mechanical strength and self-repair properties.

The $BMIMBF_4$-based ionic gel offers a high level of hyperelastic toughness, with tensile deformation reaching as high as 360%, as shown in Figure 3. The average tensile strength (Young's modulus) of the material obtained from the tensile stress-strain curve is 7.6 kPa. The tensile tests show that the tensile properties of the gel increase with increasing ZrO_2 content. An increase in the amount of ZrO_2 will result in a larger number of crosslinking sites and higher conversion rates for HEMA, which can improve the final mechanical properties of the ILGs. Based on the tensile deformation and tensile strength data above, the optimized amount of ZrO_2 is selected as 3 wt.%.

3.2. Driving Principle Analysis. The volumes of cationic $BMIM^+$ and anionic BF_4^- in the ionic liquid $BMIMBF_4$ are very different. The volume of cationic $BMIM^+$ is much larger than that of anionic BF_4^-. If voltages are applied on both sides of the actuator, the anions and cations will move toward the anode and cathode of the electrodes, respectively, under the electric field. The ions penetrate through the contact boundary of the ionic liquid polymer gel layer and the activated carbon layer; afterwards, the ions are strongly adsorbed by the activated carbon powder and accumulate in the activated carbon layer.

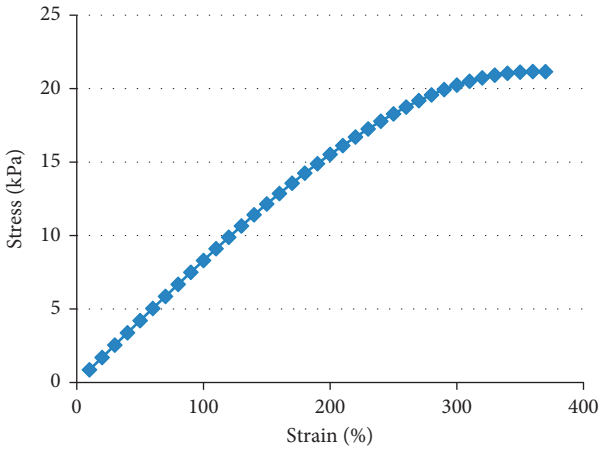


Figure 3: Stress-strain curve of ionic liquid polymer gels.

The external voltage will lead to the accumulation of the opposite charges of the two electrodes. The charges will interact with the immobilized anions in the bulk polymer, as shown in Figure 4. The interaction will attract one electrode and repel the other.

When the ion motion reaches a stable state, due to the difference between the volumes of the anions and the cations, the volume of the ILG cathode side thus expands, and the volume of the ILG anode side shrinks. The entire actuator is therefore bent toward the anode side, as shown in Figure 4.

4. Hyperelastic

4.1. Hyperelastic Stress. To apply the requirements of the mechanical performance model in the following work, the stresses in each direction of the model are supposed to be considered [16, 17] (see Figure 5).

λ_1, λ_2, and λ_3 are the x, y, and z directions of the main (extension) deformation rate, respectively, and are given by

$$\lambda_1 = \frac{x}{x_0},$$
$$\lambda_2 = \frac{y}{y_0}, \quad (1)$$
$$\lambda_3 = \frac{d}{d_0},$$

where x, y, and d are the length, width, and thickness, respectively, and x_0, y_0, and d_0 are the corresponding initial values before deformation.

As the material is incompressible, its volume is kept constant before and after deformation, giving

$$xyd = x_0 y_0 d_0. \quad (2)$$

Therefore,

$$\frac{xyd}{x_0 y_0 d_0} = \lambda_1 \lambda_2 \lambda_3 = 1. \quad (3)$$

The physical properties of the material are mainly subjected to W. Each model of the material is a special form

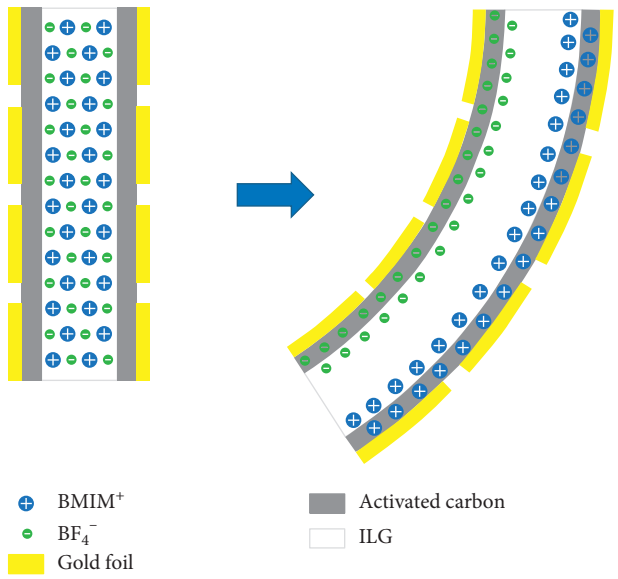


Figure 4: The working principle of the actuator.

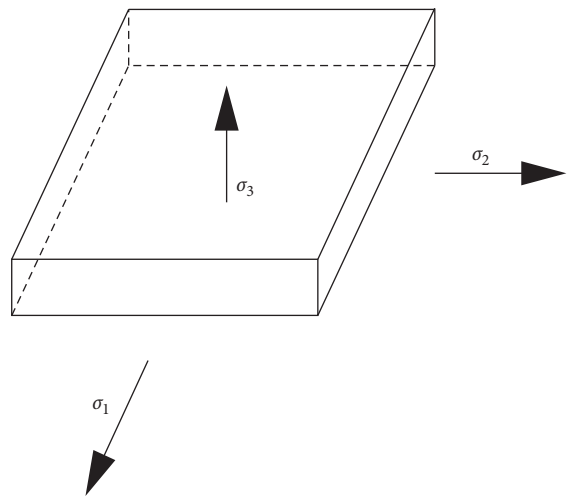


Figure 5: The stresses in each direction.

of W [18, 19]. If the form of W is determined, the Cauchy stress tensor P can be given by the equation below:

$$\sigma = -pI + 2\frac{\partial W}{\partial I_1}B - 2\frac{\partial W}{\partial I_2}B^{-1}, \quad (4)$$

where I is the unit tensor, which is the left Gaussian deformation tensor, and P is the hydrostatic pressure introduced by the incompressibility assumption.

Since I_i is invariable under any changes of B, the following is given:

$$\begin{cases} I_1 = B, \\ I_2 = \frac{1}{2}\left[I_1^2 - t_r\left(B^2\right)\right], \\ I_3 = \det B, \end{cases} \quad (5)$$

where B is the component of the Green strain tensor. The relationship between the invariants and principal elongation is a function of B.

$$\begin{cases} I_1 = t_r[B] = B_{ii} = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \\ I_2 = \frac{1}{2}\left(t_r[B]\right)^2 - \left(t_r[B]^2\right) = \frac{1}{2}\left(B_{ii}B_{ii} - B_{ij}B_{ji}\right) \\ \quad = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 = \frac{1}{\lambda_1^2} + \frac{1}{\lambda_2^2} + \frac{1}{\lambda_3^2}, \\ I_3 = \det B = \lambda_1^2\lambda_2^2\lambda_3^2. \end{cases} \tag{6}$$

The isotropic and incompressible deformation process of the ILG is given by

$$\sqrt{I_3} = \lambda_1\lambda_2\lambda_3 = 1. \tag{7}$$

According to formulas (4) and (6), it can be obtained that

$$\sigma_i = 2\left[\lambda_i^2\frac{\partial W}{\partial I_1} - \frac{1}{\lambda_i^2}\frac{\partial W}{\partial I_2}\right] - p, \tag{8}$$

where I_1, I_2, and I_3 are the relative changes in the length, surface area, and volume of the elastomer, respectively.

4.2. Ogden Model. According to a comprehensive comparison of various hyperelastic constitutive models, the Ogden model is selected in this work. The Ogden model is a preferred energy function in finite element simulation analysis. In this paper, the mechanical properties of ILG-incompressible hyperelastic materials are described by the Ogden model. The mechanical properties of the ionic gel materials were then studied using the Ogden formula to describe the nonlinearity of the ionic gels [20, 21].

The strain energy function of the Ogden model equation is given as follows:

$$W = \sum_{i=1}^{N}\frac{\mu_i}{\alpha_i}\left(\lambda_1^{\alpha_i} + \lambda_2^{\alpha_i} + \lambda_3^{\alpha_i} - 3\right), \quad (1 \le N \le 3), \tag{9}$$

where μ_i and α_i are material constants.

The form of the strain energy potential is given by

$$\begin{aligned} W = {} & \frac{\mu_1}{\alpha_1}\left(\lambda_1^{\alpha_1} + \lambda_2^{\alpha_1} + \lambda_3^{\alpha_1} - 3\right) + \frac{\mu_2}{\alpha_2}\left(\lambda_1^{\alpha_2} + \lambda_2^{\alpha_2} + \lambda_3^{\alpha_2} - 3\right) \\ & + \frac{\mu_3}{\alpha_3}\left(\lambda_1^{\alpha_3} + \lambda_2^{\alpha_3} + \lambda_3^{\alpha_3} - 3\right), \quad (N = 3). \end{aligned} \tag{10}$$

Combining equation (8) with equation (9), it can be obtained that

$$\sigma_i = \lambda_i\frac{\partial W}{\partial \lambda_i} - p. \tag{11}$$

Substituting equation (9) into equation (11) gives

$$\sigma_i = \sum_{k=1}^{3}\mu_k\lambda_i^{\alpha_k} - p. \tag{12}$$

The stresses in each direction are given by

$$\sigma_1 = \mu_1\lambda_1^{\alpha_1} + \mu_2\lambda_1^{\alpha_2} + \mu_3\lambda_1^{\alpha_3} - p, \tag{13}$$

$$\sigma_2 = \mu_1\lambda_2^{\alpha_1} + \mu_2\lambda_2^{\alpha_2} + \mu_3\lambda_2^{\alpha_3} - p, \tag{14}$$

$$\sigma_3 = \mu_1\lambda_3^{\alpha_1} + \mu_2\lambda_3^{\alpha_2} + \mu_3\lambda_3^{\alpha_3} - p. \tag{15}$$

When the ILG is uniformly stretched in the x direction, $\lambda_2 = \lambda_3$, which can be calculated with equation (3), giving

$$\lambda_2 = \lambda_3 = \frac{1}{\sqrt{\lambda_1}}. \tag{16}$$

Because only the axial tensile deformation is considered, the stresses in the other two directions are zero.

$$\sigma_2 = \sigma_3 = 0. \tag{17}$$

Combining equation (14) or (15) with equation (17), it can be obtained that

$$p = \mu_1\lambda_2^{\alpha_1} + \mu_2\lambda_2^{\alpha_2} + \mu_3\lambda_2^{\alpha_3} = \mu_1\lambda_3^{\alpha_1} + \mu_2\lambda_3^{\alpha_2} + \mu_3\lambda_3^{\alpha_3}. \tag{18}$$

Therefore,

$$\begin{aligned} \sigma_1 &= \mu_1\lambda_1^{\alpha_1} + \mu_2\lambda_1^{\alpha_2} + \mu_3\lambda_1^{\alpha_3} - \left(\mu_1\lambda_2^{\alpha_1} + \mu_2\lambda_2^{\alpha_2} + \mu_3\lambda_2^{\alpha_3}\right) \\ &= \mu_1\left(\lambda_1^{\alpha_1} - \lambda_2^{\alpha_1}\right) + \mu_2\left(\lambda_1^{\alpha_2} - \lambda_2^{\alpha_2}\right) + \mu_3\left(\lambda_1^{\alpha_3} - \lambda_2^{\alpha_3}\right) \\ &= \mu_1\left(\lambda_1^{\alpha_1} - \lambda_1^{-(1/2)\alpha_1}\right) + \mu_2\left(\lambda_1^{\alpha_2} - \lambda_1^{-(1/2)\alpha_2}\right) \\ &\quad + \mu_3\left(\lambda_1^{\alpha_3} - \lambda_1^{-(1/2)\alpha_3}\right). \end{aligned} \tag{19}$$

5. Equivalent Circuit Model

The impedance of the ionized EAP actuator is capacitive at low frequencies and resistive at high frequencies. Almost all of the equivalent circuit models are therefore composed of capacitors and resistive elements.

The driving current of the ionic liquid polymer gel actuator can be considered as a combination of the ion current, displacement current, and electron current. The ion current of the ionic polymer-metal composite (IPMC) is formed by the directional movement of the hydrated cations, and the ion current of the ionic liquid polymer gel actuator is formed by the simultaneous movement of the anions and cations in opposite directions [22, 23].

The current response of the ILG actuator is mainly composed of the ion current and the electron current, the dynamic performance is exhibited by the ion current characteristics, and the static performance is exhibited by the electron current characteristics.

The simplified resistance-capacitance (R-C) equivalent circuit model of the ILG flexible actuator is illustrated in Figure 6. R_1 is the direct current (DC) equivalent resistance of the actuator, and this branch represents the electron current branch, reflecting the static response of the static characteristics. R_2 is the alternating current (AC) equivalent resistance of the actuator, C is the equivalent capacitor of the actuator, and this branch represents the ion current branch, reflecting the dynamic characteristics of the drive. R_o is the surface electrode resistance of the actuator. Since the

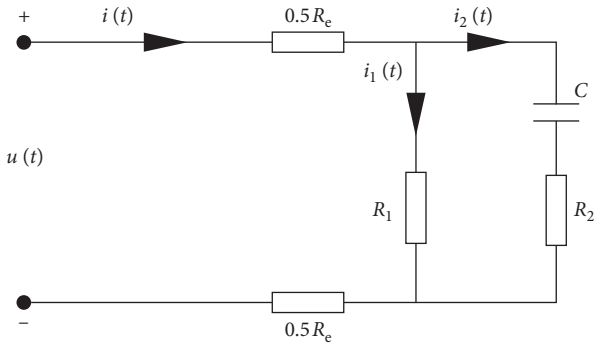


Figure 6: Equivalent circuit model.

electron current response of the ILG actuator is much faster than the ion current response and the initial value is relatively small, the nonlinearity of the electron current branch is thus negligible in the equivalent circuit. The initial energy storage of the dynamic components in the circuit is assumed to be zero, and the initial voltages at both ends of the capacitor are therefore equal to zero.

$$\begin{cases} i_1(t) = \dfrac{u_{\mathrm{RC}}(t)}{R_1}, \\ R_2 i_2(t) + \dfrac{1}{C}\displaystyle\int i_2(t)\,dt = u_{\mathrm{RC}}(t), \\ i(t) = i_1(t) + i_2(t), \\ u_{\mathrm{RC}}(t) + i(t)R_e = u(t), \end{cases} \tag{20}$$

where i_1 is the electron current of the actuator, i_2 is the ion current of the actuator, i is the input current of the actuator, u is the input voltage, and u_{RC} is the actuator actuation voltage.

Under zero-input conditions, the transfer function of the input current and input voltage of the actuator can be obtained from the Laplace transform of equation (20), giving

$$\frac{I(s)}{U(s)} = \frac{(R_1 + R_1)Cs + 1}{[R_1R_2 + R_e(R_1 + R_2)]Cs + R_1 + R_e}. \tag{21}$$

When the input is a step signal with an amplitude of u_0, substituting in equation (21) gives

$$I(s) = \frac{(R_1 + R_1)Cs + 1}{[R_1R_2 + R_e(R_1 + R_2)]Cs + R_1 + R_e}\frac{U_0}{s} = \frac{a_0}{s} + \frac{a_1}{s + \lambda}, \tag{22}$$

$$\begin{cases} a_0 = U_0\dfrac{R_1R_2 + R_1R_e + R_2R_2}{[R_1R_2 + R_e(R_1 + R_2)](R_1 + R_e)}, \\ a_1 = U_0\dfrac{R_1^2}{[R_1R_2 + R_e(R_1 + R_2)](R_1 + R_e)}, \\ \lambda = \dfrac{R_1 + R_e}{[R_1R_2 + R_e(R_1 + R_2)]C}. \end{cases} \tag{23}$$

The current time-domain response of the equivalent circuit model can be obtained from the inverse Laplace transform of equation (22), giving

$$i(t) = a_0 + a_1e^{-\lambda t}. \tag{24}$$

For the equivalent circuit model, the fitting precision function is given by

$$J_1 = \|i_s - i_e\|_2^2 = \sum_{j=1}^{n}\left(i_{sj} - i_{ej}\right)^2, \tag{25}$$

where i_s is the current time-domain response of the equivalent circuit model and i_e is the experimental current data.

The system parameters are identified by the step-response curve and least-squares method. The optimal parameters are $a_0 = 0.0060$, $a_1 = 0.2058$, and $\lambda = 0.1470$.

Therefore,

$$i(t) = 0.006 + 0.2058e^{-0.147t}. \tag{26}$$

6. CPG Control

The CPG is an oscillation unit composed of a series of intermediate neurons, and the entire CPG control network is a complex distributed neural network, which combines a neural oscillator and a multiple-reflection feedback loop system. The CPG network can generate a stable phase interlocking relationship by mutual inhibition of the neurons and can also generate rhythmic motions by self-oscillating excitation of body-related parts. Each cell (motor neuron) is assumed to activate a single actuator in the motor system; the number of cells of the CPG is thus equal to the number of actuators for a structure with n actuators. The synaptic connections among the neurons in the CPG are elastic, and the CPG network can therefore provide a variety of output modes and control the animals to achieve different movement patterns [24–26].

Each oscillator in the CPG chain is aligned in a chain shape, and the rhythmic signal of a body fluctuation can be transmitted from one end to the other; the constant phase lag between every two segments is fixed so that the animal maintains a body-long wavelength at any speed, as shown in Figure 7. When an earthworm creates fluctuating movement, traveling waves propagate from the tail to the head, leading to continuous bending of the muscles of the segment from the tail to the head. To implement this kind of movement mode, a traveling wave propagating from the cellular neural network cell associated with the last (tail) segment to the cell associated with the first (head) segment is supposed to be generated.

Generally, a degree of freedom of movement is subjected to an oscillator, and a plurality of oscillators can form different topologies to control the coordinated motion of the animal.

6.1. Structure of the Actuator Model. A joint arm has a limited number of degrees of freedom, and an ILG soft robot

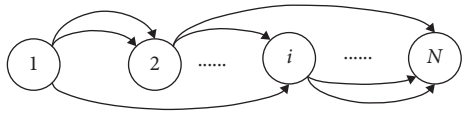


Figure 7: Chained CPG network topology.

Figure 8: The actuator model.

actuator is a continuum with an unlimited number of degrees of freedom. Therefore, the control of soft robots is a great challenge. A discrete substitution of the continuous description of the soft robot actuator is therefore used to simplify the model.

The soft robot actuator is thus modeled using the point quality and the spring of the 2D array. The masses of all the soft robot actuators are concentrated on the discrete points connected by a massless damping spring. In this paper, the soft actuator model is 2D, and all force vectors are in the *x-y* plane, where *y* is the gravitational direction, and the motion of the soft robot is limited in the plane.

(1) The simulated motion of the model is free motion, without interaction with any other objects.

(2) The soft robot actuator is discretized into 15 segments. A comparison of the accuracy of the 15-segment model and that of the 30-segment model shows that the former can provide the desired continuous structure of a soft robot actuator.

(3) All the forces contained in the model are limited in the *x-y* plane. When this model is compared with the 3D model, although the generality is limited, the computational cost is greatly reduced, and the reaching movement of the soft robot is also achievable.

In this paper, there are 15 segments in this model. Figure 8 shows the general structure of the modeled soft robot actuator.

The bending deformation of the ILG is a result of the differences in volume caused by the movement of the cations and anions in opposite directions. The reliability of the soft robot actuator model is based on the assumption that the ILG is incompressible. Due to the constant-volume constraints, the bending deformation of the soft robot actuator will shorten the length of one side while increasing the length of the other side. A simple physical mechanism is used in the 2D model, in which the motion of the soft robot actuator is almost unrestricted in the plane, and the force is transferred from one side of the soft robot actuator to the other side, without the need for a rigid skeleton.

6.2. Dynamic Model of Soft Robot Actuator. The soft robot actuator model built in this work is a 2D model in which all forces are vectors in the *x-y* plane. The movement of the soft robot actuator is thus limited to the plane. In this paper, three types of forces for the actuator are involved. The simplified motion equation can be given by

$$M\ddot{q} = F_{\mathrm{m}} + F_{\mathrm{g}} + F_{\mathrm{c}}, \tag{27}$$

where M is the diagonal mass matrix, q is the position vector, F_{m} is the internal force generated by the ILG, F_{g} is the vertical force resulting from the influences of gravity, and F_{c} is the internal force that maintains constant-volume constraints.

The equation is numerically integrated using an explicit Runge–Kutta equation with an adaptive step size. The initial conditions are the initial positions of all discrete mass points. The initial speed is set to zero.

The internal force of the actuator is simulated with an ideal damping spring and is caused by a change in the applied spring constant. The spring constant is adjusted to allow the user to control the movement of the actuator. The relevant physical theoretical formula is then utilized to calculate the gravity and traction of the soft robot actuator.

As a result of discretization, every linear segment of the actuator in the model exerts a force as below [27], giving

$$f(t) = \left[k_0 + k_{\max} a(t)\right]\left[l(t) - l_{\mathrm{rest}}\right] + \alpha \frac{dl(t)}{dt}, \tag{28}$$

where l_{rest} is the rest length of the soft robot actuator and is selected as the maximum length where both active and passive forces in a real actuator are zero; the linear damping coefficient α has dimensions of Ns/m; k_0 is the passive spring constant of the actuator and $k_{\max}$ is the maximum active spring constant of the actuator, both of which have dimensions of N/m; and $a(t)$ is a dimensionless activation function.

6.3. CPG Network Model. The CPG network interacts with the environment and controls robot joint signals with environmental adaptability. An actuator with N segments can control the motion of the soft robot using N oscillators, from the tail to the head of the drive, consequently causing the ILG actuator to bend from the tail to the head. The Kimura neuron oscillator consists of two mutually suppressed neurons, and each joint of the robot is driven by a neural oscillator [28, 29]. The output of the two neurons is subtracted as the output of the oscillator, and the mathematical model is summarized as the equations below:

$$\begin{cases} T_{\mathrm{r}}\dot{u}_{\{\mathrm{e,f}\}i} + u_{\{\mathrm{e,f}\}i} = w_{\mathrm{fe}} y_{\{\mathrm{e,f}\}i} - \beta v_{\{\mathrm{e,f}\}i} + \sum_{j=1}^{n} w_{ij} y_{\{\mathrm{e,f}\}i} + s_0, \\ T_{\mathrm{a}}\dot{v}_{\{\mathrm{e,f}\}i} + v_{\{\mathrm{e,f}\}i} = y_{\{\mathrm{e,f}\}i}, \\ y_{\{\mathrm{e,f}\}i} = \max\left(0, u_{\{\mathrm{e,f}\}i}\right), \\ y_i = -y_{\{\mathrm{e}\}i} + y_{\{\mathrm{f}\}i}, \end{cases} \tag{29}$$

where i, e, and f represent the flexor and extensor neurons of the i^{th} CPG unit, respectively, $u_{\{e,f\}}$ is the internal state of the neuron, $v_{\{e,f\}}$ is the self-inhibitory state of the neuron, and $y_{\{e,f\}}$ is the output of the neuron. T_r and T_a represent the rise time and adaptation time constant, respectively. w_{fe} represents the mutual inhibition coefficient of neurons, β represents the self-suppression coefficient of neurons, and s_0 represents the tonic input and determines the amplitude of the CPG output.

The CPG network is used to coordinate the multidegree of freedom of the control robot, and the robot motion mode can be adjusted by changing the parameters of the CPG. A mathematical model of each segment of the actuator can be obtained after discretizing the continuous soft robot actuator.

The CPG model of the soft actuator can be described as follows:

$$\begin{cases} T_r\dot{u}_1 + u_1 = -\beta v_1 - w\max(0, u_2) + s_0, \\ T_a\dot{v}_1 + uv_1 = \max(0, u_1), \\ T_r\dot{u}_2 + u_2 = -\beta v_2 - w\max(0, u_1) - w\max(0, u_3) + s_0, \\ T_a\dot{v}_2 + uv_2 = \max(0, u_2), \\ \cdots \\ T_r\dot{u}_{15} + u_{15} = -\beta v_{15} - w\max(0, u_{14}) + s_0, \\ T_a\dot{v}_{15} + uv_{15} = \max(0, u_{15}), \\ y_1 = \max(0, u_1) - \max(0, u_2), \\ y_2 = \max(0, u_2) - \max(0, u_3), \\ \cdots \\ y_{14} = \max(0, u_{14}) - \max(0, u_{15}), \end{cases} \tag{30}$$

where u_1, u_2, and u_3 represent the membrane potentials, v_1, v_2, and v_3 represent the adaptation or fatigue properties in real neurons, and y_1, y_2, and y_3 represent the output signals of the CPG.

Yekutieli et al. [30] demonstrated that the mechanism of bending propagation is an internal force enhancement wave. It is therefore reasonable to add the CPG single-cycle output of the active wave processing to the soft actuator. The interaction between the soft actuator and the CPG is shown by the diagram in Figure 9.

In Figure 9, the left panel is a control block diagram of the CPG, and the right panel is a control block diagram of the soft actuator with the Laplace transform [31].

The basic values of the CPG parameters are set as $T_r = 0.12$ s, $T_a = 0.3$ s, $d = 3$, $w = 3$, and $e = 1$. When the initial input value is set to [0.12 0 0 0 0 0.12], the output and phase diagrams of the CPG are obtained, as plotted in Figures 10(a) and 10(b), respectively. The motion trajectories of the soft actuator for the durations of 0.5 s, 0.7 s, 1.0 s, and 1.5 s are selected, and the motion path of the soft actuator is plotted in Figure 10(c). The internal force of the soft actuator is also obtained, as shown in Figure 10(d). The motion in Figure 10(c) can be used to simulate the soft actuator motion. Since the internal force of each segment of the soft actuator is similar to that of the first section, the internal force of the first section is shown in Figure 10(d) only.

6.4. Simulation Analysis

6.4.1. The Effects of Various e Values on the Soft Actuator Motion. In the simulation, the value of e increases with an increment of 1 in the range of (0, 50). When $e = (0, 4)$, the output and phase curves of the CPG are obtained, as shown in Figures 11(a) and 11(b), respectively, where the shape of the phase diagram of the CPG is a limit cycle. The motion of the soft actuator can also be simulated. The motions of a series of soft actuators are shown in Figure 11(c), and it can be seen that the soft actuators reach the specified location in a smooth way.

6.4.2. The Effects of Various T_r Values on the Soft Actuator Motion. When $e = 1$, the value of T_r increases with an increment of 0.1 in the range of (0, 1). When $T_r \in (0, 0.3)$, the output and phase curves of the CPG are obtained, as shown in Figures 12(a) and 12(b), respectively, where the shape of the phase diagram of the CPG is a limit cycle. The motion of the soft actuator can also be simulated. The motions of a series of soft actuators are shown in Figure 12(c), and it can be seen that the soft actuators reach the specified location in a smooth way.

6.4.3. The Effects of d and w on the Soft Actuator Motion. The effects of various d and w values on the soft actuator when $e = 1$ and $T_r = 0.1$ are investigated. The values of d and w increase with an increment of 1 in the range of (0, 50). When $d = w \in (3, 12)$, the output and phase curves of the CPG are obtained, as shown in Figures 13(a) and 13(b). The movement of the soft actuator can also be simulated, as shown in Figure 13(c), and it can be seen that the soft actuators reach the designated location in a smooth way.

In summary, a larger amplitude causes the soft actuator to become disordered. A smaller frequency destroys the limit cycle, causing failure of the maintenance of the soft actuator movement. The parameters d and w also affect the shape of the CPG phase curve of the soft actuator.

7. Conclusion

This paper demonstrates a new soft actuator design and includes information on the structure of the soft actuator, the material composition, the material mechanical properties analysis, and the CPG control simulation analysis. The structure of the soft actuator is introduced in detail, which introduces a new direction for the development of soft actuators. The material composition of the soft actuator is described, and its mechanical properties are analyzed using a hyperelastic model. The principle of bending deformation of the soft actuator is based on the fact that the volume of the cation side is much larger than the volume of

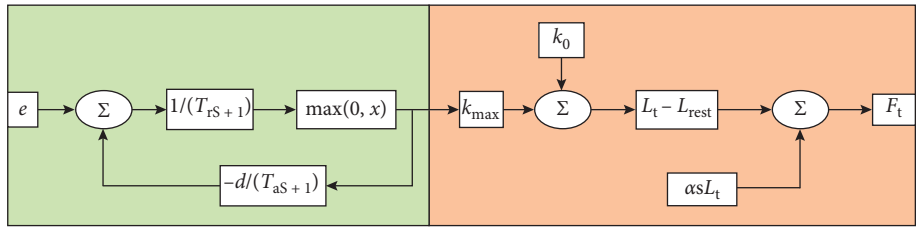


Figure 9: (a) A model of interaction between the soft robot actuator and the CPG (in green). (b) The interaction between the soft robot actuator and the CPG (in amber).

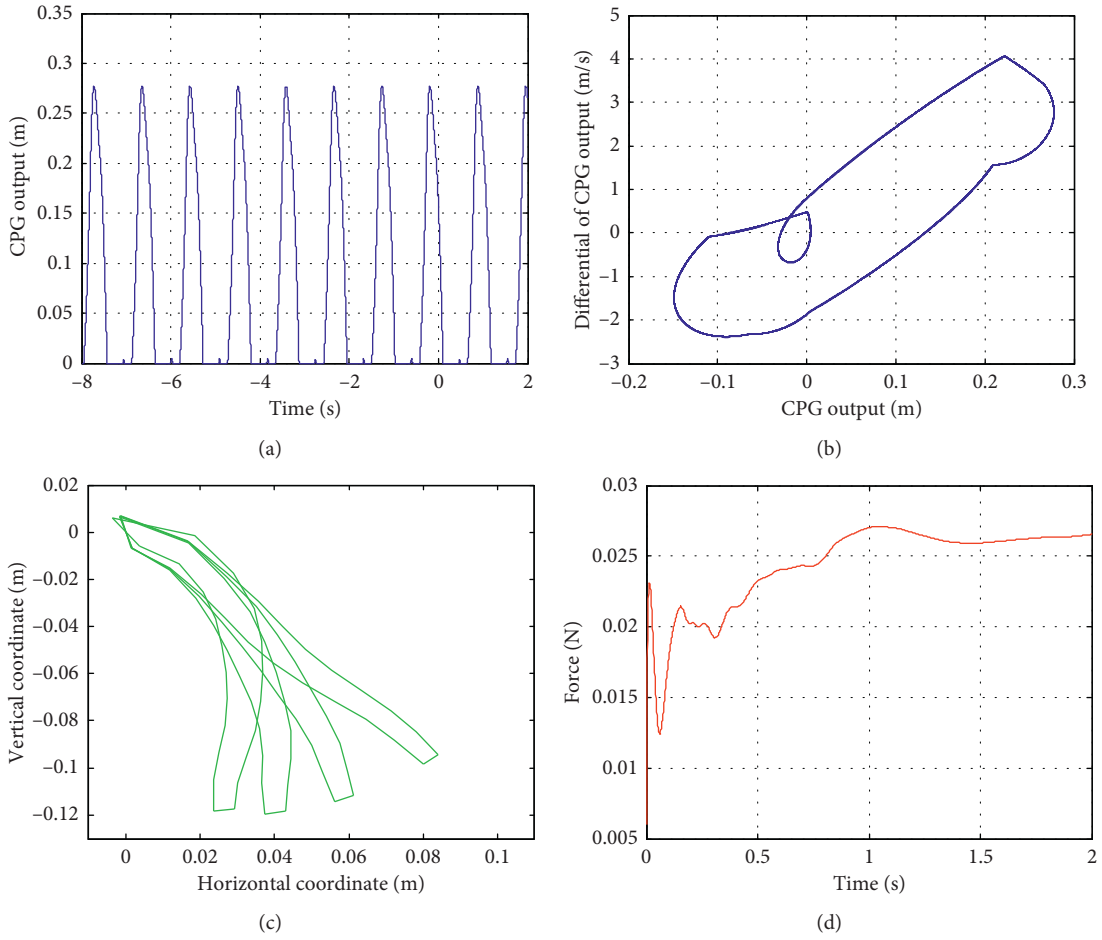


Figure 10: (a) CPG outputs. (b) Phase diagrams of these CPGs. (c) A sequence of the soft robot actuator movements. (d) The internal force of the first segment.

the anion side and that the volume on the cathode side of the actuator expands, causing the volume of the anode side to shrink; thus, the actuator bends toward the anode side. The CPG control is demonstrated in detail, and the motion of the soft actuator under various conditions is simulated. By adjusting the parameters of the CPG to achieve the motion of the soft actuator, the actuator can reach any specified position.

A soft actuator behaves similarly to octopus muscles, without any stiff skeletal support. The biomechanical properties of an octopus arm make it possible to perform tasks without being a skeletal arm. Under the driving force of an applied voltage, continuous bending deformation occurs, which increases the contact area between an actuator and an object. Additionally, this bending deformation reduces damage to objects, such as eggs, when they are being grabbed. The simulation results will help researchers to further understand the motion of soft actuators and will contribute to their development. Due to the chemical stability, thermal stability, and simple ion transport of the ILG, it is an advantageous choice for soft actuator materials. ILGs offer low weight and high distortion and can be controlled by low-voltage signals; thus, ILGs are suitable for the production of soft robot actuators. A soft actuator with a real-

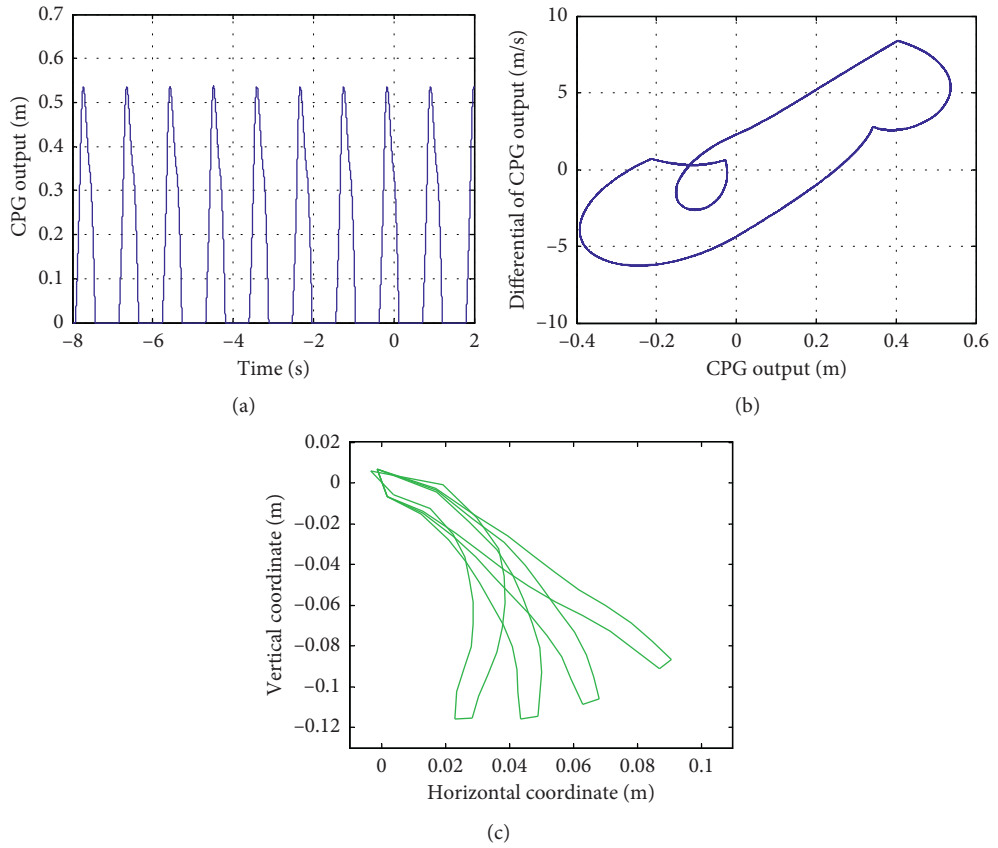


Figure 11: Output and phase diagrams of the CPG and the soft actuator trajectory with $e = 2$.

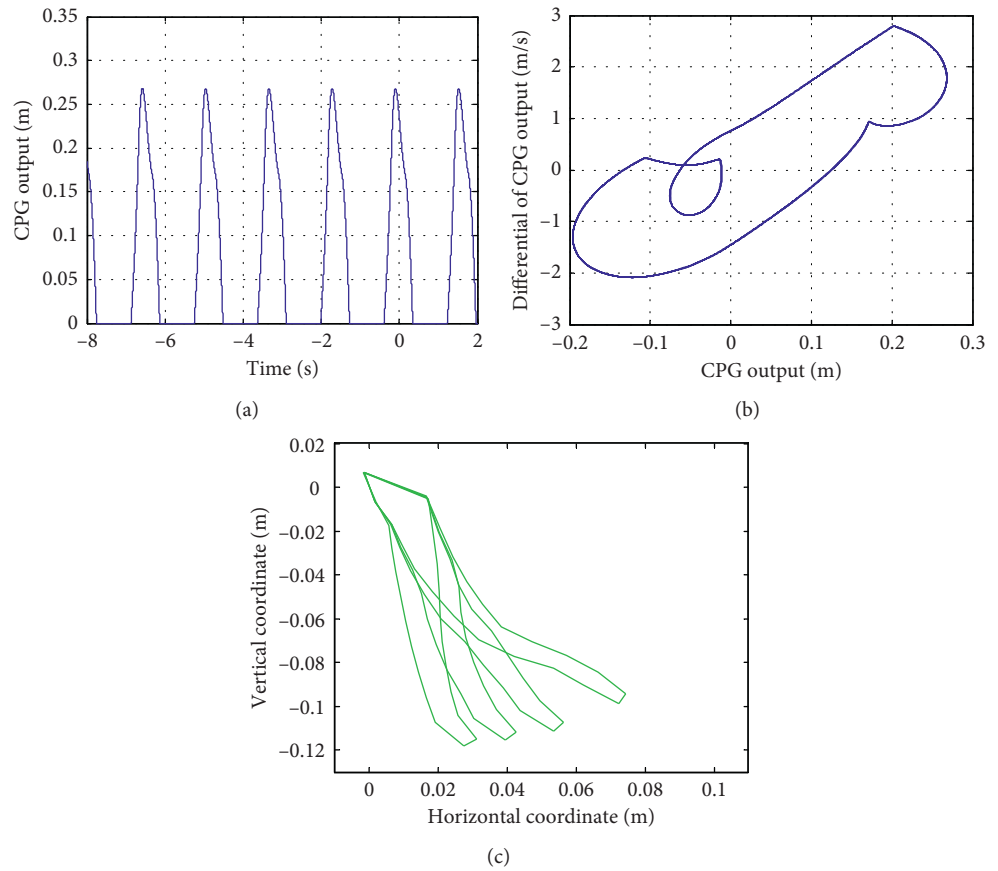


Figure 12: Output and phase diagrams of CPG and the soft actuator trajectory with $T_r = 0.15$.

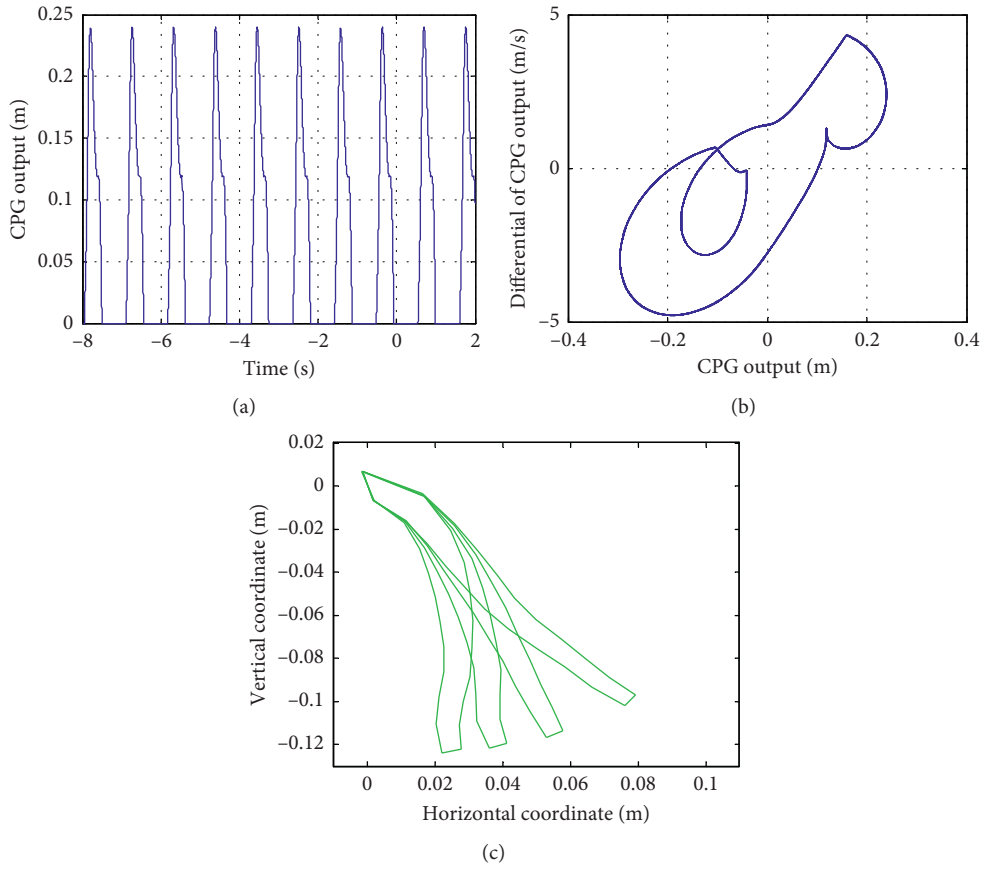


Figure 13: Output and phase diagrams of the CPG and the soft actuator trajectory with $d = w = 6$.

time learning control mechanism will thus produce highly versatile applications, which is also the direction of future related work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 51605334, 51705368, 61825303, and U1713215), Natural Science Foundation of Shanghai (Grant no. 17ZR1441800), Shanghai Sailing Program (Grant no. 17YF1420200), and Ph.D. Student Visiting Scholar Foundation of Tongji University (Grant no. 2018020018).

References

[1] B. J. Le, L. Viau, and A. Vioux, "Ionogels, ionic liquid based hybrid materials," *Chemical Society Reviews*, vol. 40, no. 2, pp. 907–925, 2011.

[2] A. Espinal, H. Rostro-Gonzalez, M. Carpio et al., "Quadrupedal robot locomotion: a biologically inspired approach and its hardware implementation," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 5615618, 13 pages, 2016.

[3] R. Pelrine, R. Kornbluh, Q. Pei, and J. Joseph, "High-speed electrically actuated elastomers with strain greater than 100%," *Science*, vol. 287, no. 5454, pp. 836–839, 2000.

[4] M. L. Hammock, A. Chortos, B. C.-K. Tee, J. B.-H. Tok, and Z. Bao, "25th anniversary article: the evolution of electronic skin (E-Skin): a brief history, design considerations, and recent progress," *Advanced Materials*, vol. 25, no. 42, pp. 5997–6038, 2013.

[5] A. W. Feinberg, A. Feigel, S. S. Shevkoplyas, S. Sheehy, G. M. Whitesides, and K. K. Parker, "Muscular thin films for building actuators and powering devices," *Science*, vol. 317, no. 5843, pp. 1366–1370, 2007.

[6] N. Buchtová, A. Guyomard-Lack, and J. Le Bideau, "Biopolymer based nanocomposite ionogels: high performance, sustainable and solid electrolytes," *Green Chemistry*, vol. 16, no. 3, pp. 1149–1152, 2014.

[7] J.-S. Wang and S. Han, "Feed-Forward neural network soft-sensor modeling of flotation process based on particle swarm optimization and gravitational search algorithm," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 147843, 10 pages, 2015.

[8] Z. Wang, B. Bin He, Q. Wang, and Y. Yin, "Electromechanical bending behavior study of soft photocurable ionogel actuator using a new finite element method," *Smart Materials and Structures*, vol. 25, no. 9, article 095018, 2016.

[9] B. He, C. Zhang, Y. Zhou, and Z. Wang, "A computing method to determine the performance of an ionic liquid gel soft actuator," *Applied Bionics and Biomechanics*, vol. 2018, Article ID 8327867, 11 pages, 2018.

[10] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[11] Y. Miyake, "Interpersonal synchronization of body motion and the walk-mate walking support robot," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 638–644, 2009.

[12] W. Chen, G. Ren, J. Wang, and D. Liu, "An adaptive locomotion controller for a hexapod robot: CPG, kinematics and force feedback," *Science China Information Sciences*, vol. 57, no. 11, pp. 1–18, 2014.

[13] M. Hägglund, L. Borgius, K. J. Dougherty, and O. Kiehn, "Activation of groups of excitatory neurons in the mammalian spinal cord or hindbrain evokes locomotion," *Nature Neuroscience*, vol. 13, no. 2, pp. 246–252, 2010.

[14] J. Li, J. Wang, S. X. Yang, K. Zhou, and H. Tang, "Gait planning and stability control of a quadruped robot," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 9853070, 13 pages, 2016.

[15] X. H. Liu, B. He, Z. P. Wang, H. F. Tan, T. Su, and Q. G. Wang, "Tough nanocomposite ionogel-based actuator exhibits robust performance," *Scientific Reports*, vol. 4, no. 1, 2014.

[16] O. H. Yeoh, "Some forms of the strain energy function for rubber," *Rubber Chemistry and Technology*, vol. 66, no. 5, pp. 754–771, 1993.

[17] B. He, C.-H. Zhang, and A. Ding, "Finite element analysis of ionic liquid gel soft actuator," *Chinese Physics B*, vol. 26, no. 12, article 126102, 2017.

[18] M. Li, Y. Yang, L. Guo, D. Chen, H. Sun, and J. Tong, "Design and analysis of bionic cutting blades using finite element method," *Applied Bionics and Biomechanics*, vol. 2015, Article ID 471347, 7 pages, 2015.

[19] L. Liu, Y. Liu, K. Yu, and J. Leng, "Thermoelectromechanical stability of dielectric elastomers undergoing temperature variation," *Mechanics of Materials*, vol. 72, pp. 33–37, 2014.

[20] K. Sangpradit, H. B. Hongbin Liu, P. Dasgupta, K. Althoefer, and L. D. Seneviratne, "Finite-element modeling of soft tissue rolling indentation," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 12, pp. 3319–3327, 2011.

[21] H. Zhang, "Strain-stress relation in macromolecular microsphere composite hydrogel," *Applied Mathematics and Mechanics*, vol. 37, no. 11, pp. 1539–1550, 2016.

[22] C. Bonomo, L. Fortuna, P. Giannone, and S. Graziani, "A circuit to model the electrical behavior of an ionic polymer-metal composite," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 2, pp. 338–350, 2006.

[23] J. D. Madden, P. G. Madden, and I. W. Hunter, "Polypyrrole actuators: modeling and performance," *Proceedings of SPIE-The International Society for Optical Engineering*, vol. 4329, pp. 72–83, 2001.

[24] R. M. Alexander, *Principles of Animal Locomotionv*, Princeton University Press, Princeton, NJ, USA, 2003.

[25] E. Niebur and P. Erdös, "Theory of the locomotion of nematodes: dynamics of undulatory progression on a surface," *Biophysical Journal*, vol. 60, no. 5, pp. 1132–1146, 1991.

[26] B. He, Q. Lu, and Z. Wang, "Coupling effect analysis between the central nervous system and the CPG network with proprioception," *Robotica*, vol. 33, no. 6, pp. 1281–1294, 2014.

[27] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov et al., "Dynamic model of the octopus arm. I. Biomechanics of the octopus reaching movement," *Journal of Neurophysiology*, vol. 94, no. 2, pp. 1443–1458, 2005.

[28] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *International Journal of Robotics Research*, vol. 22, no. 3, pp. 187–202, 2016.

[29] X. Zhang, *Biological-inspired rhythmic motion & environmental adaptability for quadruped robot*, Ph.D. dissertation, pp. 5–13, Departmentof Mechanical Engineering, Tsinghua University, Beijing, China, 2004.

[30] Y. Yekutieli, R. Sagiv-Zohar, B. Hochner, and T. Flash, "Dynamic model of the octopus arm. II. Control of reaching movements," *Journal of Neurophysiology*, vol. 94, no. 2, pp. 1459–1468, 2005.

[31] K. Matsuoka, "Analysis of a neural oscillator," *Biological Cybernetics*, vol. 104, no. 4-5, pp. 297–304, 2011.

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 4787856, 16 pages
https://doi.org/10.1155/2019/4787856

Research Article

Solving the Manufacturing Cell Design Problem through Binary Cat Swarm Optimization with Dynamic Mixture Ratios

Ricardo Soto,[1] **Broderick Crawford**,[1] **Angelo Aste Toledo**,[1] **Hanns de la Fuente-Mella**,[1] **Carlos Castro**,[2] **Fernando Paredes**,[3] **and Rodrigo Olivares**[4]

[1]*Pontificia Universidad Católica de Valparaíso, Avenida Brasil 2241, Valparaíso 2362807, Chile*
[2]*Universidad Técnica Federico Santa María, Avenida España 1680, Valparaíso 2390123, Chile*
[3]*Universidad Diego Portales, Av. Ejército 441, Santiago 8370109, Chile*
[4]*Universidad de Valparaíso, General Cruz 222, Valparaíso 2603631, Chile*

Correspondence should be addressed to Hanns de la Fuente-Mella; hanns.delafuente@pucv.cl

Received 29 October 2018; Revised 11 January 2019; Accepted 14 January 2019; Published 14 February 2019

Academic Editor: Oscar Castillo

Copyright © 2019 Ricardo Soto et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this research, we present a Binary Cat Swarm Optimization for solving the Manufacturing Cell Design Problem (MCDP). This problem divides an industrial production plant into a certain number of cells. Each cell contains machines with similar types of processes or part families. The goal is to identify a cell organization in such a way that the transportation of the different parts between cells is minimized. The organization of these cells is performed through Cat Swarm Optimization, which is a recent swarm metaheuristic technique based on the behavior of cats. In that technique, cats have two modes of behavior: seeking mode and tracing mode, selected from a mixture ratio. For experimental purposes, a version of the Autonomous Search algorithm was developed with dynamic mixture ratios. The experimental results for both normal Binary Cat Swarm Optimization (BCSO) and Autonomous Search BCSO reach all global optimums, both for a set of 90 instances with known optima, and for a set of 35 new instances with 13 known optima.

1. Introduction

Group technology is a manufacturing philosophy in which similar parts are identified and grouped together to take advantage of their similarities in design and production [1] by organizing similar parts into part families, where each part of the family has similar design and manufacturing characteristics. The basic concept of group technology has been practiced for many years around the world, as part of good engineering and scientific management practices [2, 3], which states that similar things should be manufactured in a similar way [4].

The Manufacturing Cell Design Problem (MCDP) is an application of group technology to organize cells containing a set of machines to process a family of parts [5]. In this context, MCDP involves the creation of an optimal design of production plants, in which the main objective is to minimize the movement and exchange of material between these cells, thus generating greater productivity and reducing production costs.

The Manufacturing Cell Design Problem belongs to the complex NP-hard class of problems, and then exploring good search algorithms is always a challenging task from the optimization and now also from the artificial intelligence world [5]. In particular, in this paper, an efficient metaheuristic implementation is proposed to tackle this problem, demonstrating through several benchmark instances its performance (various global optima are reached), which is also valuable from an artificial intelligence and optimization standpoint. Additionally, this algorithm includes an Autonomous Search Component (dynamic mixture ratio), which is currently an important research trend in the

optimization and metaheuristic sphere. Metaheuristics are intrinsically complex to be configured in order to reach good results, and Autonomous Search comes to facilitate this task by letting the metaheuristic itself to self-tune its internal configuration without the need of a user expert for reaching good results. To the best of our knowledge, the work done on Autonomous Search in metaheuristics is very recent, and no Autonomous Search work for cat swarm exists.

The research work that has been done to solve the problem of cell formation has followed two complementary lines, which can be organized into two groups: approximate methods and exact methods. Approximate methods are mostly focused on finding an optimal solution in a limited time; however, they do not guarantee a global optimum. Exact methods, on the contrary, aim to fully analyze the search space to ensure a global optimum [6]; however, these algorithms are quite time-consuming and can only solve cases of very limited size. For this reason, many research efforts have focused on the development of heuristics, which find near-optimal solutions within a reasonable period of time.

This research focuses on solving the MCDP through a recent metaheuristic in the vein of Swarm Intelligence (SI) [7] called Binary Cat Swarm Optimization (BCSO) [8]. This algorithm was generated from observations of cat behavior in nature, in which cats either hunt or remain alert. BCSO is based on the CSO algorithm, recently proposed by Chu and Tsai [9]. The difference is that in BCSO, the vector position consists of ones and zeros, instead of real numbers (CSO), and the proposed alternate version makes use of a dynamic mixture ratio.

As aforementioned, reaching good results for problems belonging from the NP class is always a challenging and appealing task from the optimization and artificial intelligence world. In this research, our goal was to provide an intelligent algorithm for solving this problem by additionally integrating self-tuning features, which is a very recent research trend in the optimization and metaheuristic sphere.

2. Theoretical Framework

The formation of manufacturing cells has been researched for many years. One of the first investigations focused on resolving this set of problems was Burbidge's work in 1963 [4], which proposed the use of an incidence matrix reorganized into a Block Diagonal Form (BDF) [4]. In recent years, many exact and heuristic algorithms have been proposed in the literature to solve MCDP. Such metaheuristic techniques include genetic Algorithm (GA) [10], inspired by biological evolution and its genetic-molecular basis; the Neural Network (NN) [11] that takes the behavior of neurons and the connections of the human brain; and Constraint Programming (CP) [12] where the relationships between the variables are expressed as constraints. For extensive reviews of previous research and other methods of cell formation, see Selim et al. [1].

Among the metaheuristics used for cell formation, there is also the branch of Swarm Intelligence, which was initially introduced by Beni and Wang in 1989 [13]. Inspired by nature, Swarm Intelligence systems are typically formed by a population of simple agents who interact locally with each other and with their environment and who are able to optimize an overall objective through the search for collaboration in a space [14]. Within this branch, the main techniques are Particle Swarm Optimization (PSO) designed and presented by Eberhart et al. [7, 9] in 1995; Ant Colony Optimization (ACO), which is a family of algorithms derived from Dorigo's 1991 work based on the social behavior of ants [15, 16]; Migrating Birds Optimization (MBO) [17] algorithm based on the alignment of migratory birds during flight; Artificial Fish Swarm Algorithm (AFSA) [18], based on the behavior of fish to find food by themselves or by following other fish; and the discrete Cat Swarm optimization (CSO) Technique presented in 2007 by Chu and Tsai [9], which is based on the behavior of cats. Interestingly, the CSO cat corresponds to a particle in PSO, with a small difference in its algorithms [19, 20]. CSO and PSO were originally developed for continuous value spaces, but there are a number of optimization problems where the values are discrete [21].

3. The Manufacturing Cell Design Problem

The Manufacturing Cell Design Problem (MCDP) divides an industrial production plant into a number of cells. Each cell contains machines with similar process types or part families, determined according to the similarity between parts [4]. A manufacturing cell can be defined as an independent group of functionally different machines, located together, dedicated to the manufacture of a family of similar parts. In addition, a family of parts can be defined as a collection of parts that are similar, either because of their geometric shape and size or because similar processing steps are required to manufacture them [22].

The goal of MCDP is to identify a cell organization in a way that minimizes the transport of different parts between cells, in order to reduce production costs and increase productivity. The idea is to represent the processing requirements of machine parts through an incidence matrix called machine part. This reorganization involves the formulation of two new matrices called machine-cell and part-cell.

A detailed mathematical definition of the formulation of the machine-part clustering problem is defined by the optimization model explained below [6]:

(i) M: number of machines

(ii) P: number of parts

(iii) C: number of cells

(iv) i: machine index ($i = 1, 2, \ldots, M$)

(v) j: part index ($j = 1, 2, \ldots, P$)

(vi) k: cell index ($k = 1, 2, \ldots, C$)

(vii) $M_{\max}$: maximum number of machines per cell

(viii) $A = [a_{ij}]$: machine-to-part binary incidence matrix, where

$$a_{ij} = \begin{cases} 1, & \text{if machine } i \text{ processes a part } j, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

(ix) $B = [b_{ij}]$: machine-to-part binary incidence matrix, where

$$b_{ik} = \begin{cases} 1, & \text{if machine } i \text{ belongs to cell } k, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

(x) $C = [c_{jk}]$: machine-to-part binary incidence matrix, where

$$c_{jk} = \begin{cases} 1, & \text{if part } j \text{ belongs to cell } k, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

4. Binary Cat Swarm Optimization

There are about thirty different species of known felines, e.g., lions, tigers, leopards, common housecat, etc. [23]. Although they have different living environments, cats share similar behavioral patterns [24]. For wild cats, the ability to hunt ensures food supply and survival of the species [25]. To hunt their food, wild cats form groups ranging from 2–15 individuals [26]. Domestic cats also show the same ability to hunt and are curious about moving objects [26–28]. Although cats might seem to be resting most of the time, even when awake [29, 30], they are actually in a constant state of alert; without moving, they may be listening or have their eyes open to look around [31]. BCSO [8] was formulated on the basis of all these behaviors and is an optimization algorithm that mimics the natural behavior of cats [9, 32, 33]. The authors identified two main modes of behavior for simulating cats [3, 34–39]:

(i) Seeking mode: exploration-oriented mode, where cats are attracted by moving objects and have a high hunting capacity. Cats may seem to spend most of their time resting, but in fact, they are constantly alert when moving slowly.

(ii) Tracing mode: exploitation-oriented mode, where cats detect a prey and run after it, spending a lot of energy due to its rapid movements. In this way, the cats follow the best in their group.

In BCSO, these two behaviors are mathematically modeled to solve complex optimization problems. The first decision is to define the number of cats needed for each iteration. Each cat, represented by cat_k, where $k \in \{1, 2, \ldots, C\}$, has its own position consisting of M dimensions composed of ones and zeros (1 and 0). In addition, they have speed for each dimension d, a flag to indicate whether the cat is in the seeking or tracing mode, and finally a fitness value that is calculated based on the MCDP. The BCSO keeps looking for the best solution until iterations are finalized. In BCSO, each cat_x represents a MCDP solution through a machine-cell matrix, where x identifies the cat and d are the position bits of the cat. In addition, the constraint matrix ensures that each row i is covered by at least one column.

Algorithm 1 describes the general BCSO pseudocode where the mixture ratio (MR) is a percentage that determines the number of cats in the seeking mode.

4.1. Seeking Mode. This submodels the state of the cat, which is resting, looking around, and seeking the next position to move towards. The seeking mode has the following essential factors:

(i) PMO: probability of mutation operation, a percentage that defines the mutation probability for the selected dimension.

(ii) CDC: counts of dimensions to change, a percentage that indicates how many dimensions are candidates to change.

(iii) SMP: seeking memory pool, a positive integer used to define the memory size for each cat. SMP indicates the points to be scanned by the cat and can be different for different cats.

The following pseudocode describes the behavior of the cat in the seeking mode. Here, FS_i is the fitness of the ith cat, and $\text{FS}_b = \text{FS}_{\max}$ finds the minimum solution and $\text{FS}_b = \text{FS}_{\min}$ the maximum solution. To solve the MCDP, we use $\text{FS}_b = \text{FS}_{\max}$.

Step 1: create SMP copies of current cat_x.

Step 2: for each copy:

for dimensions that are candidates for change (based on CDC percentage):

get a random number (rand) between 0 and 1

if rand < PMO, then the position changes.

Step 3: evaluate Fitness of all copies.

Step 4: calculate the selection probability by applying a roulette wheel or, by default, choose the best copy according to Fitness.

$$P_i = \left| \frac{\text{FS}_i - \text{FS}_b}{\text{FS}_{\max} - \text{FS}_{\min}} \right|. \quad (4)$$

Step 5: evaluate if the chosen copy is a better solution than the currently selected cat, and replace accordingly.

Figure 1 shows the flow chart of the behavior of the cat in the seeking mode.

4.2. Tracing Mode. This submodel is used to model the state of the cat in hunting or tracing behavior, where the cats are moving towards the best solution obtained so far. Once a cat enters the tracing mode, it moves according to its own velocities for each dimension. Each cat has two velocity vectors, defined as V^1_{kd} and V^0_{kd}, where V^0_{kd} is the probability that the bits of the cat change to zero and V^1_{kd} is the probability they change to one. The velocity vector changes its meaning with the probability of mutation for each dimension d. The tracing mode action is described in the following pseudocode.

```
(1) Create C cats
(2) Randomly initialize cat position values with values between 1 and 0
(3) Initialize velocities and flag of each cat;
(4) while (i < NumberIterations) do
(5)       Evaluate cats according to fitness function
(6)       Store the position of the best cat, which has best fitness
(7) for (x = 1 to C) do
(8)          if (randomNumber < MixtureRatio) then
(9)             Apply seeking mode process to cat_x
(10)         else
(11)            Apply tracing mode process to cat_x
(12)         end if
(13)      Evaluate new solution, update values
(14)         end for
(15) end while
(16) Postprocess results and visualization.
```

Algorithm 1: Binary Cat Swarm Algorithm.

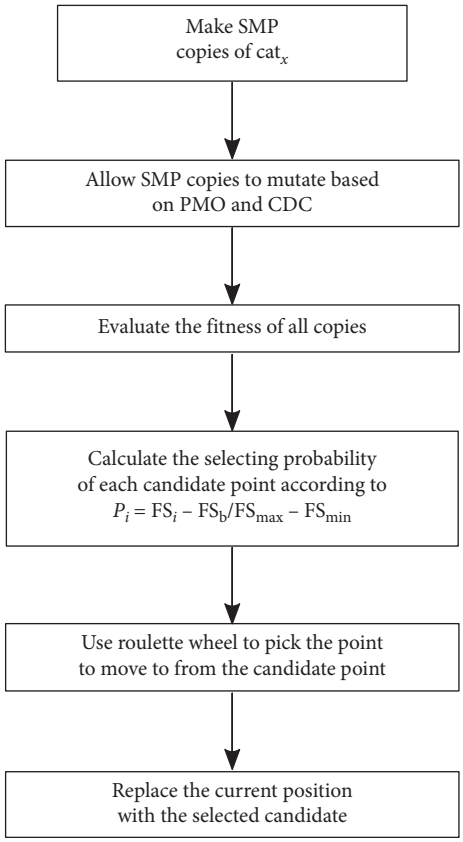


Figure 1: Seeking mode.

Step 1: calculate d^1_{kd} and d^0_{kd} according to the following expression, where $X_{best,d}$ is the dimension d of the best cat, r_1 has random values in the range of [0,1], and c_1 is a user-defined constant.

$$\begin{aligned} &\text{If } X_{best,d} = 1, \text{then } d^1_{kd} = -r_1 c_1, \text{ and } d^0_{kd} = r_1 c_1, \\ &\text{If } X_{best,d} = 0, \text{then } d^1_{kd} = r_1 c_1 y, \ d^0_{kd} = -r_1 c_1. \end{aligned} \tag{5}$$

Step 2: update values for V^1_{kd} and V^0_{kd} according to the expression, where w is the inertia weight and M is the number of columns.

$$\begin{aligned} V^1_{kd} &= \omega V^1_{kd} + d^1_{kd}, \\ V^0_{kd} &= \omega V^0_{kd} + d^0_{kd}, \end{aligned} \quad d = 1, \ldots, M. \tag{6}$$

Step 3: calculate the velocity of cat$_k$, V'_{kd}, according to

$$V'_{kd} = \begin{cases} V^1_{kd}, & \text{If } X_{kd} = 0, \\ V^0_{kd}, & \text{If } X_{kd} = 1. \end{cases} \tag{7}$$

Step 4: calculate the probability of mutation in each dimension, defined by parameter T_{kd} which takes a value in the interval of [0,1]

$$T_{kd} = \frac{1}{1 + e^{-V'_{kd}}}. \tag{8}$$

Step 5: based on the value of T_{kd}, the new value of each dimension of the cat is updated as follows:

$$X_{kd} = \begin{cases} X_{best,d}, & \text{If rand} < t_{kd}, \\ X_{kd}, & \text{If } t_{kd} < \text{rand}, \end{cases} \quad d = 1, \ldots, M. \tag{9}$$

The maximum velocity vector of V'_{kd} must be limited to value V_{max}.

If the value of V'_{kd} surpasses that of V_{max}, V'_{kd} must be selected for the corresponding velocity dimension.

The following is a flow chart for a cat in the tracing mode (Figure 2).

5. Solving the Manufacturing Cell Design Problem (MCDP)

To solve the MCDP, it is essential to use a repair method for solutions that were not feasible. Algorithm 2 describes the pseudocode used to solve the MCDP.

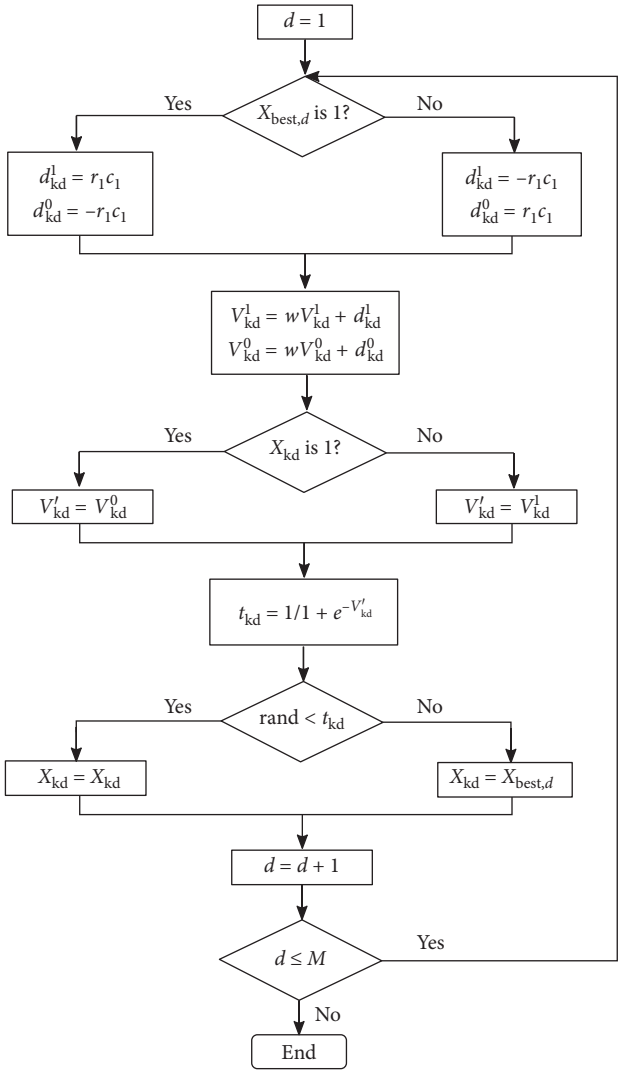


Figure 2: Tracing mode.

6. Repair Method

A solution may not satisfy the constraints, resulting in an unworkable solution. For this reason, the value that violates the constraint is repaired instead of the matrix being removed. In this section, a function is described to transform nonfeasible solutions into feasible solutions.

Thus, Algorithm 3 presents a repair method in which all rows not covered are identified and assigned accordingly. This will cover all restrictions.

7. Autonomous Search

Autonomous Search (AS) is a modern approach that allows the solver to automatically reconfigure its resolution parameters to provide better performance when bad results are detected [40].

In this context, performance is assessed through indicators that collect relevant information during the search. Search parameters are then updated advantageously according to the results obtained by the fitness evaluation.

This approach has been effectively applied to different optimization and satisfaction techniques, such as Constraint Programming [41], SAT [42], mixed integer programming [43, 44], and various other metaheuristic techniques [45–47].

In the present investigation, a version of the BCSO with Autonomous Search has been implemented, where the mixture ratio (MR) variable is used as an autonomous parameter; i.e., the MR value changes while the program is executed to give a more dynamic algorithm that directly influences the mode that the cat will take.

Algorithm 4 is the pseudocode describing the Autonomous Search BCSO.

8. Results

The BCSO implementation process of MCDP has led to results that will be presented in the following section. The metaheuristic was programmed in the JAVA programming language. For the execution of the algorithm, the parameters considered were the following:

(i) Iterations = 5000
(ii) Number of cats = 30
(iii) MR = 0.75 (75% seeking; 25% tracing)
(iv) SMP = 15
(v) CDC = 0.2
(vi) PMO = 0.76
(vii) $w = 1$
(viii) $c1 = 1$
(ix) $r1 \in [0, 1]$

9. Boctor Instances

Tests with the implemented solution were carried out based on 90 instances of 16×30 matrices, obtained from 10 problems found in the paper of Boctor [48], hereafter called Boctor Instances. These problems included the use of 2 or 3 cells. In the case of 2 cells, the maximum number of machines ($M_{\max}$) in each took values between 8 and 12. In the case of 3, $M_{\max}$ varied between 6 and 9 machines per cell. In both cases, the value of $M_{\max}$ remained constant throughout the execution of the algorithm.

The values obtained by submitting each problem to the Classic BCSO and BCSO with Autonomous Search are summarized in Tables 1–9, where "*O*" denotes the global optimum given in [48]; "BCSO," the best value obtained by the BCSO here proposed; "*A*," the average number of optima obtained; "*I*," the average number of iterations in which the optimum is reached; "Ms," the time (in milliseconds) used to reach the optimum; and "RPD," the Relative Percent Difference, calculated as follows:

$$\text{RPD} = \frac{Z - Z_{\text{opt}}}{Z_{\text{opt}}} * 100, \tag{10}$$

where Z_{opt} is the best known optimal value and Z is the best optimal value achieved by BCSO.

```
(1) Create C cats, each cat is a machine-cell matrix
(2) Initialize the machine-cell matrices with random values (1 or 0)
(3) Initialize all other parameters for each cat
(4)    while (i < NumberIterations) do
(5)    Evaluate MCDP fitness of the cats
(6)    Store position of Best Matrix cat_x with highest fitness value
(7)    for (x = 1 to C) do
(8)       if (randomNumber < MixtureRatio) then
(9)          Apply seeking mode process to cat_x
(10)         Repair each modified matrix
(11)      else
(12)         Apply tracing mode process to cat_x
(13)         Repair each modified matrix
(14)      end if
(15)      Evaluate new solution and update values
(16)   end for
(17) end while
(18) Postprocess results and visualization
```

Algorithm 2: Solving MCDP.

```
(1)    for (i to Machines) do
(2)       for (j to Cells) do
(3)          Count the number of cells the same machine is assigned to
(4)       end for
(5)       if (Assignments! = 1) then
(6)          Calculate least cost column
(7)          Assign the machine to the calculated least cost cell
(8)       end if
(9)    end for
(10)   for (i to Machines) do
(11)   for (j to Cells) do
(12)      Count the number of machines in the same cell
(13)   end for
(14)   if (Number of grouped machines is greater than M_max) then
(15)      Find cell with fewer machines assigned
(16)      Reassign the machine to found cell
(17)   end if
(18) end for
```

Algorithm 3: Repairing solutions.

The above results were run 40 times for each of the 90 Boctor Instances. It is important to point out that 100% of these were optimized, proving that BCSO can work with any MCDP instance. The performance of the BCSO metaheuristic in its Autonomous Search version was slightly better, demonstrated by some of the optima averages reached in the experimental results.

10. Other Author Instances

To analyze the effectiveness of the implemented algorithm in a wider range of problems, new instances from different authors were investigated. Matrix sizes ranged from 5 to 40 machines and from 7 to 100 parts. Table 10 shows the instances used:

In order to improve the quality of the exhibited behavior by the autonomous version of the Binary Cat Swarm Optimization, we performed a detailed comparison by using these new instances, because they are hardest. This comparison includes two well-known metaheuristics: the first one is inspired by the behavior of the Egyptian vulture (EVOA) [71], and the second one mimics the flashing behavior of fireflies [72]. Table 11 reports the result comparison between our proposal and the methods published in [73].

If it observes the showed results for instances CF01 to CF11, we can conclude that BCSO presents a similar performance to EVOA. In both cases, the optimal values are reached. Moreover, we note the worst and mean values are equal. This behavior can be attributed to the similarity of the

```
(1)  while (i < NumberIterations) do
(2)  if (FitnessIteration = = FitnessIterationPrevious) then
(3)    RepetitionsFitness++
(4)  else
(5)    RepetitionsFitness = 0
(6)  end if
(7)  if (RepetitionsFitness > 30) then
(8)    Change MixtureRatio to 1/(MixtureRatio * 50)
(9)    if (MixtureRatio < 10%) then
(10)     Change MixtureRatio to MixtureRatio * 15
(11)     if (MixtureRatio < 50%) then
(12)       Reinitialize 5 cat (machine-cell matrices) with random values (1 or 0)
(13)       for (x = 1 to C) do
(14)         Change PMO to 0.9
(15)         Change CDC to 0.3
(16)       end for
(17)     else
(18)       for (x = 1 to C) do
(19)         Restore PMO value to 0.76
(20)         Restore CDC value to 0.2
(21)       end for
(22)       RepetitionsFitness = 0
(23)     end if
(24)   end if
(25)   Order cat arrangements
(26) end if
(27) end while
```

Algorithm 4: Autonomous search.

Table 1: Experimental results with cell = 2 and $M_{max} = 8$.

		Results for Boctor Instances with $C = 2$ $M_{max} = 8$									
P	O	Classic BCSO					Autonomous Search BCSO				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0.00	5	30697.5	11	11	0.00	5	15977.6
2	7	7	7	0.00	7	27797.7	7	7	0.00	7	14487.6
3	4	4	4.05	0.00	79	27927.8	4	4	0.00	133	14175.7
4	14	14	14	0.00	5	29998.9	14	14	0.00	6	15149.6
5	9	9	9	0.00	93	28348.8	9	9	0.00	102	14358.0
6	5	5	5	0.00	5	28233.3	5	5	0.00	8	14165.0
7	7	7	7	0.00	5	28338.9	7	7	0.00	7	14562.7
8	13	13	13	0.00	6	28860.8	13	13	0.00	7	14895.6
9	8	8	8	0.00	6	28206.7	8	8	0.00	8	14583.8
10	8	8	8	0.00	19	28547.1	8	8	0.00	20	14750.2
$\overline{X}$	8.6	8.6	8.605	0.00	23	28695.7	8.6	8.6	0.00	30.3	14710.6

operations between both algorithms. Now, if it evaluates MFAO with respect to BCSO, we again can report a similar conclusion. Nevertheless, in CF05 and CF07, BCSO achieves two optimal values that they are not reached with MFAO.

From CF12 onwards, BCSO begins to exhibit an outstanding performance. For instance, in CF12, BCSO is the only one that finds the best solution (optimum value) reaching RPD 0%. Its closer competitor (MBFA) obtains RPD 28.57%. However, the biggest significant difference can be seen from CF15. In this instance, BCSO exhibits higher efficiency than EVOA and it overcomes the reached value by MBFA. Now, if taken any instances between CF16 and CF 35 (more than 57% of instances), the good yield of the BCSO exceeds the two compared approaches term of the best-found values, average-found values, and worst-found values also. Therefore, we can state that BCSO is more than a competitive technique. It is a real alternative for solving the Manufacturing Cell Design Problem.

Now, the values obtained by submitting each problem to Classic BCSO and BCSO with Autonomous Search are summarized in Table 12, where the global optimum is given in [74].

Table 2: Experimental results with cell = 2 and $M_{max} = 9$.

Results for Boctor Instances with $C = 2$ $M_{max} = 9$											
P	O	Classic BCSO					Autonomous Search BCSO				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0	5	19412	11	11	0	5	15109.4
2	6	6	6	0	5	17803.6	6	6	0	12	14284.1
3	4	4	4	0	5	16817	4	4	0	6	14140.3
4	13	13	13	0	5	18236.3	13	13	0	5	15027.4
5	6	6	6	0	81	16863.1	6	6	0	57	14228.9
6	3	3	3	0	10	16552.5	3	3	0	24	13997.8
7	4	4	4	0	5	17681.6	4	4	0	5	14432.9
8	10	10	10	0	7	18277.8	10	10	0	9	14734.4
9	8	8	8	0	5	17690.7	8	8	0	5	14473.7
10	5	5	5	0	7	18035.8	5	5	0	7	14645.4
$\overline{X}$	7	7	7	0	13.5	17737	7	7	0	13.5	14507.4

Table 3: Experimental results with cell = 2 and $M_{max} = 10$.

Results for Boctor Instances with $C = 2$ $M_{max} = 10$											
P	O	Classic BCSO					Autonomous Search BCSO				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0	5	18084.4	11	11	0	5	15047
2	4	4	4	0	9	16894.2	4	4	0	11	14336.8
3	4	4	4	0	5	16221.7	4	4	0	5	13991
4	13	13	13	0	6	17602.6	13	13	0	6	15051.8
5	6	6	6	0	7	16459.6	6	6	0	36	14620.4
6	3	3	3	0	8	16217.7	3	3	0	38	14853.1
7	4	4	4	0	5	16828.9	4	4	0	6	15199.7
8	8	8	8	0	8	17648.1	8	8	0	8	15858.6
9	8	8	8	0	6	16669.3	8	8	0	5	15226.4
10	5	5	5	0	6	16713.1	5	5	0	7	15591.5
$\overline{X}$	6.6	6.6	6.6	0	6.5	16933.96	6.6	6.6	0	12.7	14977.63

Table 4: Experimental results with cell = 2 and $M_{max} = 11$.

Results for Boctor Instances with $C = 2$ $M_{max} = 11$											
P	O	Classic BCSO					Autonomous Search BCSO				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0	5	17173.2	11	11	0	5	16818.2
2	3	3	3	0	7	16000.2	3	3	0	9	15159.5
3	3	3	3	0	8	15755.9	3	3	0	29	14537.3
4	13	13	13	0	6	17011.4	13	13	0	6	15634.2
5	5	5	5	0	9	16680.3	5	5	0	18	14958.3
6	3	3	3	0	6	16433.2	3	3	0	7	28722.4
7	4	4	4	0	5	16714.5	4	4	0	6	15837.8
8	5	5	5	0	6	17223.9	5	5	0	6	17155.2
9	5	5	5	0	10	16733.8	5	5	0	22	16827.5
10	5	5	5	0	6	16698.9	5	5	0	7	17077.1
$\overline{X}$	5.7	5.7	5.7	0	6.8	16642.53	5.7	5.7	0	12	17272.75

The above results were obtained after 40 executions for each of the 35 new instances. It should be noted that it was possible to reach optima in 100% of instances for both algorithms, proving that BCSO can work with almost any instance. The performance of the BSCO metaheuristic in its Autonomous Search version was slightly better, demonstrated in some of the optima achieved, improving by 3% with respect to the original.

11. Results for Boctor Instances Using BCSO and BCSO with Autonomous Search

Figure 3 shows the results of the experiments conducted for the Boctor Instances presented above. Thanks to the operation mode of the BCSO, a fast optimum convergence is obtained at $C = 2$; however, when $C = 3$, the BCSO does not converge as quickly that said, the optimum is reached

Table 5: Experimental results with cell = 2 and $M_{max} = 12$.

Results for Boctor Instances with $C = 2$ $M_{max} = 12$											
P	O	Classic BCSO					Autonomous Search BCSO				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0	5	18226.7	11	11	0	5	17694.2
2	3	3	3	0	6	16901.6	3	3	0	6	16481
3	1	1	1	0	8	16377.7	1	1	0	21	15203.2
4	13	13	13	0	5	17816.3	13	13	0	6	15927.9
5	4	4	4	0	6	16824.9	4	4	0	20	15004.5
6	2	2	2	0	35	16411.3	2	2	0	158	14512
7	4	4	4	0	6	16939	4	4	0	6	15269.6
8	5	5	5	0	7	17716.7	5	5	0	7	15861.7
9	5	5	5	0	8	17175.5	5	5	0	18	15091
10	5	5	5	0	6	20025.5	5	5	0	7	15258.2
$\overline{X}$	5.3	5.3	5.3	0	9.2	17441.52	5.3	5.3	0	25.4	15630.33

Table 6: Experimental results with cell = 3 and $M_{max} = 6$.

Results for Boctor Instances with $C = 3$ $M_{max} = 6$											
P	O	Classic BCSO					Autonomous Search				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	27	27	27	0	40	23130.3	27	27	0	85	18900.6
2	7	7	7	0	12	21710.5	7	7	0	11	17891.1
3	9	9	9	0	38	21010.5	9	9	0	52	437225
4	27	27	27	0	10	22764.7	27	27	0	11	18664.8
5	11	11	11	0	11	21380.6	11	11	0	11	17956.1
6	6	6	6	0	13	20749.7	6	6	0	15	17323.4
7	11	11	11	0	60	21698.3	11	11	0	91	17904.1
8	14	14	14	0	14	22665.2	14	14	0	12	18360.5
9	12	12	12	0	12	21730.1	12	12	0	22	17725.5
10	10	10	10	0	27	22397.3	10	10	0	32	18011.2
$\overline{X}$	13	13.4	13	0	24	21923.7	13.4	13.4	0	34	59996.3

Table 7: Experimental results with cell = 3 and $M_{max} = 7$.

Results for Boctor Instances with $C = 3$ $M_{max} = 7$											
P	O	Classic BCSO					Autonomous Search				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	18	18	18	0	42	22915.4	18	18	0	49	18995.1
2	6	6	6	0	14	21540.1	6	6	0	16	18216.9
3	4	4	4	0	27	20955.9	4	4	0	20	17177.4
4	18	18	18	0	21	23486.3	18	18	0	22	18684
5	8	8	8	0	15	19261	8	8	0	15	16942.8
6	4	4	4	0	28	18009	4	4	0	24	16508.8
7	5	5	5	0	35	18720	5	5	0	243	16994.5
8	11	11	11	0	14	300126	11	11	0	15	17666.8
9	12	12	12	0	16	21566.1	12	12	0	14	17191.1
10	8	8	8	0	16	21380.5	8	8	0	16	17549.1
$\overline{X}$	9	9.4	9	0	23	48796.1	9.4	9.4	0	43	17592.7

in most cases before 100 executions, which demonstrates the effectiveness of the proposed approach.

Figure 4 shows the results of problem 3, $C = 2$ and $M_{max} = 8$, over iterations. Both versions converge quickly: while the Autonomous Search BCSO reaches the optimum early (iteration 10), the normal BCSO is stuck at optimum of fitness 5 at iteration 4.

The following graph (Figure 3) shows the results of problem 7, with $C = 3$, $M_{max} = 8$, reaching the overall optimum in both cases at similar iterations: normal

Table 8: Experimental results with cell = 3 and $M_{max} = 8$.

Results for Boctor Instances with $C = 3$ $M_{max} = 8$											
P	O	Classic BCSO					Autonomous Search				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0	16	21580	11	11	0	15	18090.8
2	6	6	6	0	20	20246.3	6	6	0	20	17017.4
3	4	4	4	0	17	19927.2	4	4	0	42	16878.1
4	14	14	14	0	19	21022.2	14	14	0	24	18642.2
5	8	8	8	0	36	19499.3	8	8	0	215	17156.4
6	4	4	4	0	31	19735.9	4	4	0	144	16542.7
7	5	5	5	0	30	20064.9	5	5	0	39	17352
8	11	11	11	0	19	21113.4	11	11	0	76	18459.6
9	8	8	8	0	36	20879.4	8	8	0	56	17950
10	8	8	8	0	17	19959.3	8	8	0	17	18353.6
$\overline{X}$	8	7.9	8	0	24	20402.8	7.9	7.9	0	65	17644.3

Table 9: Experimental results with cell = 3 and $M_{max} = 9$.

Results for Boctor Instances with $C = 3$ $M_{max} = 9$											
P	O	Classic BCSO					Autonomous Search				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	11	11	11	0	13	21872.7	11	11	0	16	18462.7
2	6	6	6	0	20	20489.4	6	6	0	16	17624.9
3	4	4	4	0	14	20044	4	4	0	15	16748.8
4	13	13	13	0	15	22408	13	13	0	17	17698.7
5	6	6	6	0	69	22768.2	6	6	0	168	17120.3
6	3	3	3	0	69	19580.2	3	3	0	139	17167.1
7	4	4	4	0	24	20863.5	4	4	0	29	17602.5
8	10	10	10	0	66	23977.9	10	10	0	184	18202.8
9	8	8	8	0	15	26618.9	8	8	0	21	17849.1
10	5	5	5	0	17	20694.3	5	5	0	26	18483.6
$\overline{X}$	7	7	7	0	32	21931.7	7	7	0	63	17696.1

Table 10: New instances from other authors.

Problem	Author	Machines	Parts	Cells	M_{max}
CFP01	King and Nakornchai [49]	5	7	2	3
CFP02	Waghodekar and Sahu [50]	5	7	2	4
CFP03	Seifoddini [51]	5	18	2	3
CFP04	Kusiak and Cho [52]	6	8	2	3
CFP05	Kusiak and Chow [53]	7	11	5	2
CFP06	Boctor [48]	7	11	4	2
CFP07	Seifoddini and Wolfe [54]	8	11	4	3
CFP08	Chandrasekharan and Rajagopalan [55]	8	20	3	4
CFP09	Chandrasekharan and Rajagopalan [56]	8	20	2	5
CFP10	Mosier and Taube [57]	10	10	5	4
CFP11	Chan and Milner [58]	10	15	3	4
CFP12	Askin and Subramanian [59]	14	24	7	3
CFP13	Stanfel [60]	14	24	7	3
CFP14	McCormick et al. [61]	16	24	8	5
CFP15	Srinivasan et al. [62]	16	30	6	6
CFP16	King [63]	16	43	8	4
CFP17	Carrie [64]	18	24	9	4
CFP18	Mosier and Taube [65]	20	20	6	7
CFP19	Kumar et al. [66]	23	20	7	6
CFP20	Carrie [64]	20	35	5	5
CFP21	Boe and Cheng [67]	20	35	5	5
CFP22	Chandrasekharan and Rajagopalan [68]	24	40	12	5

Table 10: Continued.

Problem	Author	Machines	Parts	Cells	M_{max}
CFP23	Chandrasekharan and Rajagopalan [68]	24	40	7	5
CFP24	Chandrasekharan and Rajagopalan [68]	24	40	7	5
CFP25	Chandrasekharan and Rajagopalan [68]	24	40	11	5
CFP26	Chandrasekharan and Rajagopalan [68]	24	40	12	3
CFP27	Chandrasekharan and Rajagopalan [68]	24	40	12	3
CFP28	McCormick et al. [61]	27	27	6	11
CFP29	Carrie [64]	28	46	10	4
CFP30	Kumar and Vannelli [69]	30	41	14	4
CFP31	Stanfel [60]	30	50	13	3
CFP32	Stanfel [60]	30	50	14	4
CFP33	King-Nakornchai [49]	36	90	17	6
CFP34	McCormick et al. [61]	37	53	3	15
CFP35	Chandrasekharan and Rajagopalan [70]	40	100	10	6

Table 11: Comparison between classic BCSO.

ID	M	P	C	M_{max}	Optimum values	EVOA				MBFA				CSOA			
						Best	Worst	Mean	RPD (%)	Best	Worst	Mean	RPD (%)	Best	Worst	Mean	RPD (%)
CF01	5	7	2	3	0	0	0	0	0.00	0	0	0	0.00	0	0	0	0.00
CF02	5	7	2	4	3	3	3	3	0.00	3	3	3	0.00	3	3	3	0.00
CF03	5	18	2	3	5	5	5	5	0.00	5	5	5	0.00	5	5	5	0.00
CF04	6	8	2	3	2	2	2	2	0.00	2	2	2	0.00	2	2	2	0.00
CF05	7	11	5	2	8	8	8	8	0.00	9	9	9	12.50	8	8	8	0.00
CF06	7	11	4	2	4	4	4	4	0.00	4	4	4	0.00	4	4	4	0.00
CF07	8	12	4	3	7	7	7	7	0.00	8	8	8	14.29	7	7	7	0.00
CF08	8	20	3	4	7	7	7	7	0.00	7	7	7	0.00	7	7	7	0.00
CF09	8	20	2	5	25	25	25	25	0.00	27	27	27	8.00	25	25	25	0.00
CF10	10	10	5	4	0	0	2	1.2	0.00	3	3	3	0.00	0	0	0	0.00
CF11	10	15	3	4	0	0	4	0.8	0.00	0	0	0	0.00	0	0	0	0.00
CF12	14	24	7	3	7	11	16	13.3	57.14	9	11	10.1	28.57	7	7	7	0.00
CF13	14	24	7	3	8	12	17	14.3	50.00	8	9	8.4	0.00	8	8	8	0.00
CF14	16	24	8	5	Unknown	30	35	32.9	—	36	41	39.6	—	24	24	24	—
CF15	16	30	6	6	Unknown	31	39	35.7	—	18	25	21.1	—	17	17	17	—
CF16	16	43	8	4	Unknown	42	47	44.6	—	39	46	43.8	—	29	30	29.05	—
CF17	18	24	9	4	Unknown	32	36	34.2	—	32	35	33.2	—	26	27	26.53	—
CF18	20	20	6	7	Unknown	46	53	49.9	—	52	59	56.2	—	41	42	41.18	—
CF19	20	23	7	6	Unknown	51	56	53.4	—	49	55	51.6	—	38	38	38	—
CF20	20	35	5	5	Unknown	28	42	36	—	7	16	12.3	—	2	2	2	—
CF21	20	35	5	5	Unknown	57	65	60.3	—	43	45	43.5	—	35		35	—
CF22	24	40	7	5	Unknown	30	43	37.5	—	0	23	15.5	—	0	5	4.9	—
CF23	24	40	7	5	Unknown	39	48	44.2	—	13	19	15	—	10	15	13.53	—
CF24	24	40	7	5	Unknown	44	53	49.7	—	25	30	27.6	—	18	22	20.98	—
CF25	24	40	11	5	Unknown	60	64	61.6	—	49	57	56.1	—	40	44	43.6	—
CF26	24	40	12	3	Unknown	68	71	70	—	64	67	65.6	—	59	63	62.15	—
CF27	24	40	12	3	Unknown	69	72	70.6	—	67	72	68.8	—	61	66	64.05	—
CF28	27	27	6	11	Unknown	84	100	94.1	—	76	97	92.1	—	54	54	54	—
CF29	28	46	10	4	Unknown	102	119	112.8	—	106	112	109.1	—	91	98	96.1	—
CF30	30	41	14	4	Unknown	57	63	59.7	—	43	65	58.3	—	37	43	42.6	—
CF31	30	50	13	3	Unknown	70	79	75.3	—	54	63	60.4	—	52	59	57.9	—
CF32	30	50	14	4	Unknown	86	90	87.6	—	76	81	77.6	—	66	75	72.15	—
CF33	36	90	17	6	Unknown	136	153	144.8	—	116	125	122.6	—	93	95	94.93	—
CF34	37	53	3	15	Unknown	352	383	369.2	—	325	335	329.5	—	256	256	256	—
CF35	40	100	10	6	Unknown	181	207	195.6	—	114	130	119.2	—	83	121	110.58	—

Table 12: Experimental results for new instances.

		Results with 35 new instances									
P	O	Classic BCSO					Autonomous Search BCSO				
		BCSO	A	RPD	I	Ms	BCSO	A	RPD	I	Ms
1	0	0	0	0	1	3583.2	0	0	0	1	3433.4
2	3	3	3	0	1	3619.7	3	3	0	1	3558.5
3	5	5	5	0	1	6273	5	5	0	1	6155.3
4	2	2	2	0	1	4157.7	2	2	0	1	3807.2
5	8	8	8	0	1	7897.8	8	8	0	1	7575.8
6	4	4	4	0	2	6919	4	4	0	2	6389.2
7	7	7	7	0	4	8158.6	7	7	0	5	7529.3
8	7	7	7	0	4	10048.4	7	7	0	4	9240.1
9	25	25	25	0	2	9373.3	25	25	0	2	8705.8
10	0	0	0	0	8	8982.7	0	0	0	10	8254.2
11	0	0	0	0	4	8893.5	0	0	0	4	8164.4
12	7	7	7	0	128	21747.6	7	7	0	194	19928.8
13	8	8	8	0	67	21835	8	8	0	87	20146.5
14	Unknown	24	24		103	27558	24	24		147	25383.5
15	Unknown	17	17		260	27283.5	17	17		296	25025.1
16	Unknown	29	29.05		922	40535.1	29	29.08		1268	37265.8
17	Unknown	26	26.53		717	31776.2	26	26.73		876	30591.1
18	Unknown	41	41.18		1241	26712.9	41	41.5		1174	27322.4
19	Unknown	38	38		577	31345.7	38	38.3		590	32086.5
20	Unknown	2	2		300	31251.8	2	2		358	29678.2
21	Unknown	35	35		318	34413.4	35	35.08		443	33892.7
22	Unknown	0	4.9		1909	42425.3	0	2.48		2017	103160.1
23	Unknown	10	13.53		1649	45014.9	10	12.43		1988	42954.3
24	Unknown	18	20.98		1958	45974.2	18	20.03		2080	43359.1
25	Unknown	40	43.6		2186	62062.9	40	44.08		2096	56834.7
26	Unknown	59	62.15		815	66630.4	57	60.73		2352	61659.3
27	Unknown	61	64.05		1204	66655.3	61	63.55		2202	62349.9
28	Unknown	54	54		466	43228.5	54	54.05		477	41759.4
29	Unknown	91	96.1		1434	76860.5	90	95.2		2578	67203.9
30	Unknown	37	42.6		1270	84810.9	34	40.9		2520	75800.3
31	Unknown	52	57.9		641	93391.4	49	54.3		2453	81183.7
32	Unknown	66	72.15		1670	99440.7	67	71.8		2431	87624.9
33	Unknown	93	94.93		2423	165907.8	93	95.48		1999	143833
34	Unknown	256	256		1345	70985.4	256	256		1005	70893.8
35	Unknown	83	110.58		964	153404.2	55	82.2		3579	150723
$\overline{X}$	5.85	34.51	36.63	0	703	42547.4	33.49	35.51	0	1007	41242.1

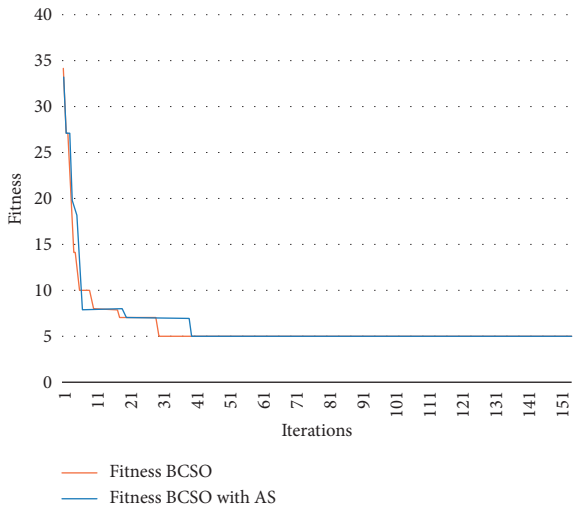


Figure 3: Graph showing the results of problem 7 for BCSO and BCSO AS with $C = 3$.

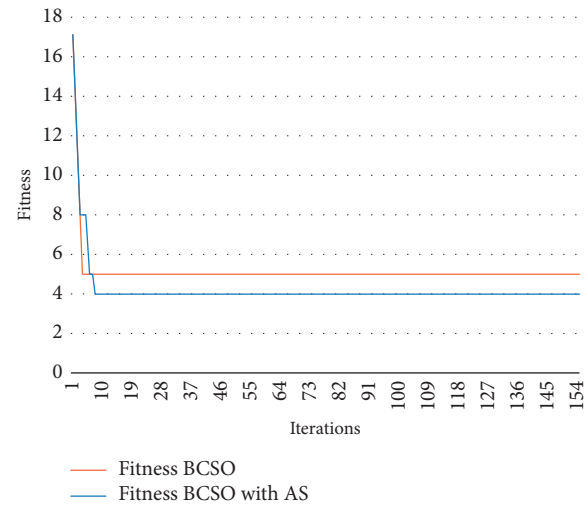


Figure 4: Graph showing the results of problem 3 for BCSO and BCSO AS with $C = 2$.

BCSO, iteration 30; and Autonomous Search BCSO, iteration 40.

12. Results for New Instances Using BCSO and BCSO with Autonomous Search

Figure 5 shows the results of the experiments performed for new instances, in which it can be seen that the Autonomous Search algorithm helps the solution not to get trapped at some local optimum; however, not all results with Autonomous Search present an advantage over the original version.

Figure 5 represents the results of problem 26, with $M = 24$, $P = 40$, $C = 12$, and $M_{\max} = 3$, in which it can be seen that Autonomous Search BCSO does not have a great difference over the normal BCSO; however, Autonomous Search BCSO is able to explore new solutions, which makes it achieve better results.

The graph in Figure 6 represents the results of problem 30, with $M = 30$, $P = 41$, $C = 14$, and $M_{\mathrm{MAX}} = 4$, in which it can be seen that the Autonomous Search BCSO solutions continue to change without being trapped in a local optimum, whereas normal BCSO is trapped near iteration 4000.

The graph in Figure 7 represents the results of problem 35, with $M = 40$, $P = 100$, $C = 10$, and $M_{\max} = 6$, in which Autonomous Search BCSO solutions are changing, exploring new solutions, expanding their search space early on, before iteration 3000; normal BCSO is trapped in a local optimum near iteration 1000.

13. Conclusions

In the present investigation, a new algorithm inspired by cat behavior, called Cat Swarm Optimization, was presented in solving the Manufacturing Cell Design Problem, used for placement of machinery in a manufacturing plant.

The proposed BCSO was implemented and tested using 90 Boctor Instances plus 35 new instances, for a total of 125 instances: The BCSO managed to obtain 100% of known optima in the 90 Boctor Instances, achieving rapid convergence and reduced execution times. In the case of the 35 new instances, it was possible to obtain 100% of the 13 known optima. It should be noted that these results were obtained after a long testing process, where the different parameters of the algorithm were calibrated based on experimentation. For that reason, Autonomous Search was implemented as an optimization method to influence variables in real time, which resulted in dynamic MR that slightly improved results obtained: 3% compared to the original, with 100% of the known optima, both for the 90 Boctor Instances and the 35 new instances.

As can be seen from the results, this metaheuristic behaves well in all observed cases. This research demonstrates that BCSO is a valid alternative for solving the MCDP. The algorithm works well, regardless of the scale of the problem. However, solutions obtained could be improved by using different parameters for each set of instances.

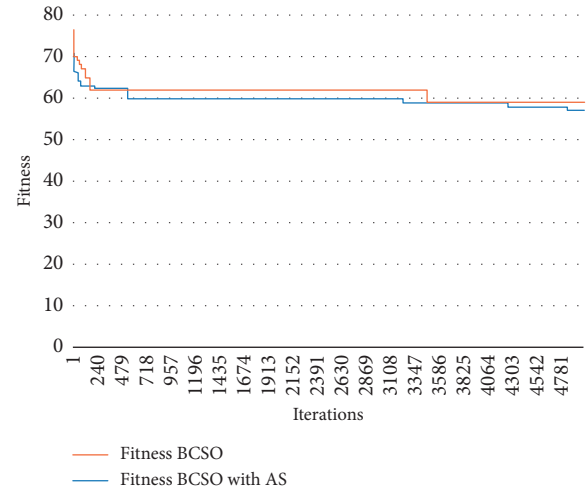


Figure 5: Graph showing the results of problem 26 for BCSO and BCSO AS.

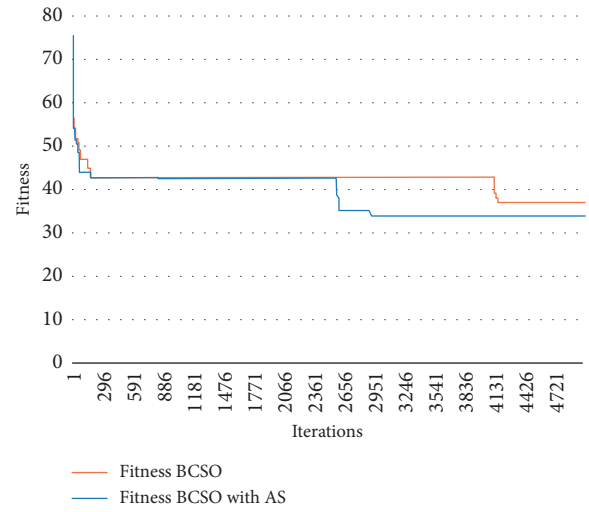


Figure 6: Graph showing the results of problem 30 for BCSO and BCSO AS.

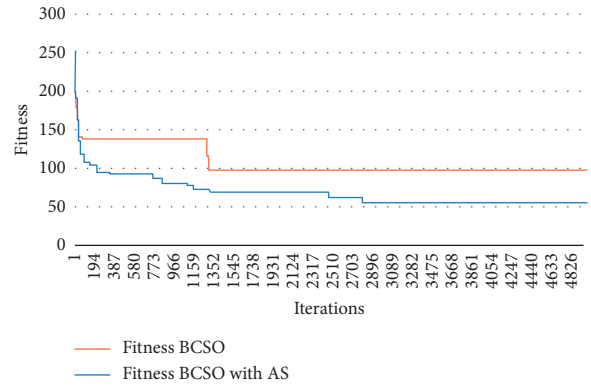


Figure 7: Graph showing the results of problem 35 for BCSO and BCSO AS.

The BCSO performance was significantly increased after selecting a good repair technique. However, relying on a repair method leads us not to recommend the use of this algorithm for other types of problems because it is far less efficient than other techniques for more complex problems.

For future research, a more extensible configuration could be developed to cover a wider set of problems. It would also be interesting to implement this technique in conjunction with other recent metaheuristics where limited work on Autonomous Search exists such as cuckoo search, firefly optimization, or bat algorithms [75]. Finally, hybridization with learning techniques is another interesting research line to pursue, where feedback gathered for the self-tune phase could be processed with machine learning in order to better track the complete solving process.

Data Availability

The authors declare that the data used to support the findings of this study are available from the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Broderick Crawford was supported by the Grant CONICYT/FONDECYT/REGULAR/1171243, and Ricardo Soto was supported by the Grant CONICYT/FONDECYT/REGULAR/1160455.

References

[1] H. M. Selim, R. G. Askin, and A. J. Vakharia, "Cell formation in group technology: review, evaluation and directions for future research," *Computers & Industrial Engineering*, vol. 34, no. 1, pp. 3–20, 1998.

[2] I. Ham, K. Hitomi, and T. Yoshida, *Group Technology: Applications to Production Management*, Springer Science & Business Media, Berlin, Germany, 2012.

[3] H. Zhenggang, Z. Guo, and J. Wang, "Integrated scheduling of production and distribution operations in a global MTO supply chain," *Enterprise Information Systems*, vol. 2018, pp. 1–25, 2018.

[4] P. D. Medina, E. A. Cruz, and M. Pinzón, "Generación de celdas de manufactura usando el algoritmo de ordenamiento binario (aob)," *Scientia Et Technica*, vol. 1, no. 44, pp. 106–110, 2010.

[5] J. L. Burbidge, *The Introduction of Group Technology*, Halsted Press, New York, NY, USA, 1975.

[6] R. Soto, H. Kjellerstrand, O. Durán, B. Crawford, E. Monfroy, and F. Paredes, "Cell formation in group technology using constraint programming and boolean satisfiability," *Expert Systems with Applications*, vol. 39, no. 13, pp. 11423–11427, 2012.

[7] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, Hoboken, NJ, USA, 2006.

[8] Y. Shara, M. A. Khanesar, and M. Teshnehlab, "Discrete binary cat swarm optimization algorithm," in *Proceedings of Computer, Control & Communication (IC4), 2013 3rd International Conference*, pp. 1–6, IEEE, Karachi, Pakistan, September 2013.

[9] S.-C. Chu and P.-W. Tsai, "Computational intelligence based on the behavior of cats," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 1, pp. 163–173, 2007.

[10] A. Kusiak, "The part families problem in flexible manufacturing systems," *Annals of Operations Research*, vol. 3, no. 6, pp. 277–300, 1985.

[11] M. Shargal, S. Shekhar, and S. A. Irani, "Evaluation of search algorithms and clustering efficiency measures for machine-part matrix clustering," *IIE transactions*, vol. 27, no. 1, pp. 43–59, 1995.

[12] H. Seifoddini and C.-P. Hsu, "Comparative study of similarity coefficients and clustering algorithms in cellular manufacturing," *Journal of Manufacturing Systems*, vol. 13, no. 2, pp. 119–127, 1994.

[13] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?*, pp. 703–712, Springer, Berlin, Germany, 1993.

[14] J. F. Kennedy, J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, Burlington, MA, USA, 2001.

[15] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[16] F. Olivas, F. Valdez, O. Castillo, C. I. Gonzalez, G. Martinez, and P. Melin, "Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems," *Applied Soft Computing*, vol. 53, pp. 74–87, 2017.

[17] R. Soto, B. Crawford, B. Almonacid, and F. Paredes, "A migrating birds optimization algorithm for machine-part cell formation problems," in *Proceedings of Mexican International Conference on Artificial Intelligence*, pp. 270–281, Springer, Cuernavaca, MX, USA, October 2015.

[18] G. Srinivasan, "A clustering algorithm for machine cell formation in group technology using minimum spanning trees," *International Journal of Production Research*, vol. 32, no. 9, pp. 2149–2158, 2007.

[19] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of 1997 Computational Cybernetics and Simulation, 1997 IEEE International Conference*, pp. 4104–4108, IEEE, Orlando, FL, USA, October 1997.

[20] J. So and W. Jenkins, "Comparison of cat swarm optimization with particle swarm optimization for IIR system identification," in *Proceedings of Signals, Systems and Computers, 2013 Asilomar Conference*, pp. 903–910, IEEE, Pacific Grove, CA, USA, November 2013.

[21] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "A novel binary particle swarm optimization," in *Proceedings of Control & Automation, 2007. MED'07. Mediterranean Conference*, pp. 1–6, IEEE, Marrakech, Morocco, June 2007.

[22] S. A. Irani, *Handbook of Cellular Manufacturing Systems*, John Wiley & Sons, Hoboken, NJ, USA, 1999.

[23] V. Aspinall, *Complete Textbook of Veterinary Nursing*, Butterworth Heinemann, Oxford, UK, 2006.

[24] B. Pallaud, "Hypotheses on mechanisms underlying observational learning in animals," *Behavioural Processes*, vol. 9, no. 4, pp. 381–394, 1984.

[25] J. Dards, "Feral cat behaviour and ecology," *Bulletin of the Feline Advisory Bureau*, vol. 15, 1976.

[26] A. Yamane, T. Doi, and Y. Ono, "Mating behaviors, courtship rank and mating success of male feral cat (*felis catus*)," *Journal of Ethology*, vol. 14, no. 1, pp. 35–44, 1996.

[27] S. Crowell-Davis, "Cat behaviour: social organization, communication and development," in *The Welfare Of Cats*, I. Rochlitz, Ed., pp. 1–22, Springer, Berlin, Germany, 2005.
[28] W. Sung, "Effect of gender on initiation of proximity in free ranging domestic cats (*Felis catus*)," M.Sc. thesis, University of Georgia, Athens, GA, USA, 1998.
[29] R. E. Adamec, "The interaction of hunger and preying in the domestic cat (*Felis catus*): an adaptive hierarchy?," *Behavioral Biology*, vol. 18, no. 2, pp. 263–272, 1976.
[30] H. Adler, "Some factors of observation learning in cats," *Journal of Genetic Psychology*, vol. 86, no. 1, pp. 159–177, 1995.
[31] B. Santosa and M. K. Ningrum, "Cat swarm optimization for clustering," in *Proceedings of 2009 International Conference of Soft Computing and Pattern Recognition*, pp. 54–59, Malacca, Malaysia, December 2009.
[32] G. Panda, P. M. Pradhan, and B. Majhi, "IIR system identification using cat swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12671–12683, 2011.
[33] P.-W. Tsai, J.-S. Pan, S.-M. Chen, and B.-Y. Liao, "Enhanced parallel cat swarm optimization based on the taguchi method," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6309–6319, 2012.
[34] G.-G. Wang, A. H. Gandomi, X. Zhao, and H. C. E. Chu, "Hybridizing harmony search algorithm with cuckoo search for global numerical optimization," *Soft Computing*, vol. 20, no. 1, pp. 273–285, 2014.
[35] M. Yazdani and F. Jolai, "Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm," *Journal of Computational Design and Engineering*, vol. 3, no. 1, pp. 24–36, 2016.
[36] X. Zhang, K. Tang, S. Li, K. Xia, and D. Zhao, "Design of slow-wave structure based on multi-objective quantum particle swarm optimization algorithm with inertia weight," *Chinese Journal of Vacuum Science & Technology*, vol. 30, no. 6, pp. 651–656, 2010.
[37] D. Zhao, K. Xia, H. Liu, and X. Shi, "A pitch distribution in slow-wave structure of STWT using Cauchy mutated cat swarm optimization with gravitational search operator," *Journal of the Chinese Institute of Engineers*, vol. 41, no. 4, pp. 297–307, 2018.
[38] M. Zhao, "A novel compact cat swarm optimization based on differential method," *Enterprise Information Systems*, vol. 2018, pp. 1–25, 2018.
[39] C. Peraza, F. Valdez, and P. Melin, "Optimization of intelligent controllers using a type-1 and interval type-2 harmony search algorithm," *Algorithms*, vol. 10, no. 3, p. 82, 2017.
[40] Y. Hamadi, E. Monfroy, and F. Saubion, "What is autonomous search?," *Hybrid Optimization*, pp. 357–391, Springer, Berlin, Germany, 2011.
[41] B. Crawford, R. Soto, E. Monfroy, W. Palma, C. Castro, and F. Paredes, "Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1690–1695, 2013.
[42] F. Hutter, Y. Hamadi, H. H. Hoos, and K. Leyton-Brown, "Performance prediction and automated tuning of randomized and parametric algorithms," in *Proceedings of International Conference on Principles and Practice of Constraint Programming*, pp. 213–228, Springer, Nantes, France, September 2006.
[43] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Automated configuration of mixed integer programming solvers," in *Proceedings of International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pp. 186–202, Springer, Bologna, Italy, June 2010.
[44] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of International Conference on Learning and Intelligent Optimization*, pp. 507–523, Springer, Rome, Italy, Januray 2011.
[45] J. Maturana, F. Lardeux, and F. Saubion, "Autonomous operator management for evolutionary algorithms," *Journal of Heuristics*, vol. 16, no. 6, pp. 881–909, 2010.
[46] J. Maturana and F. Saubion, "On the design of adaptive control strategies for evolutionary algorithms," in *Proceedings of International Conference on Artificial Evolution (Evolution Artificielle)*, pp. 303–315, Springer, Tours, France, October 2007.
[47] J. Maturana and F. Saubion, "A compass to guide genetic algorithms," in *Proceedings of International Conference on Parallel Problem Solving from Nature*, pp. 256–265, Springer, Birmingham, UK, September 2008.
[48] F. F. Boctor, "A Jinear formulation of the machine-part cell formation problem," *International Journal of Production Research*, vol. 29, no. 2, pp. 343–356, 1991.
[49] J. R. King and V. Nakornchai, "Machine-component group formation in group technology: review and extension," *International Journal of Production Research*, vol. 20, no. 2, pp. 117–133, 2007.
[50] P. H. Waghodekar and S. Sahu, "Machine-component cell formation in group technology: Mace," *International Journal of Production Research*, vol. 22, no. 6, pp. 937–948, 2007.
[51] H. Seifoddini, "A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications," *International Journal of Production Research*, vol. 27, no. 7, pp. 1161–1165, 2007.
[52] A. Kusiak and M. Cho, "Similarity coefficient algorithms for solving the group technology problem," *International Journal of Production Research*, vol. 30, no. 11, pp. 2633–2646, 2007.
[53] A. Kusiak and W. S. Chow, "Efficient solving of the group technology problem," *Journal of Manufacturing Systems*, vol. 6, no. 2, pp. 117–124, 1987.
[54] H. Seifoddini and P. M. Wolfe, "Application of the similarity coefficient method in group technology," *IIE transactions*, vol. 18, no. 3, pp. 271–277, 1986.
[55] M. P. Chandrasekharan and R. Rajagopalan, "MODROC: an extension of rank order clustering for group technology," *International Journal of Production Research*, vol. 24, no. 5, pp. 1221–1233, 1986.
[56] M. P. Chandrasekharan and R. Rajagopalan, "An ideal seed non-hierarchical clustering algorithm for cellular manufacturing," *International Journal of Production Research*, vol. 24, no. 2, pp. 451–463, 1986.
[57] C. Mosier and L. Taube, "The facets of group technology and their impacts on implementation-A state-of-the-art survey," *Omega*, vol. 13, no. 5, pp. 381–391, 1985.
[58] H. M. Chan and D. A. Milner, "Direct clustering algorithm for group formation in cellular manufacture," *Journal of Manufacturing systems*, vol. 1, no. 1, pp. 65–75, 1982.
[59] R. G. Asktn and S. P. Subramantan, "A cost-based heuristic for group technology configuration," *International Journal of Production Research*, vol. 25, no. 1, pp. 101–113, 2007.
[60] L. E. Stanfel, "Machine clustering for economic production," *Engineering costs and production economics*, vol. 9, no. 1–3, pp. 73–81, 1985.
[61] W. T. McCormick Jr., P. J. Schweitzer, and T. W. White, "Problem decomposition and data reorganization by a

clustering technique," *Operations Research*, vol. 20, no. 5, pp. 993–1009, 1972.

[62] G. Srinvasan, T. Narendran, and B. Mahadevan, "An assignment model for the part families problem in group technology," *International Journal of Production Research*, vol. 28, no. 1, pp. 145–152, 1990.

[63] J. R. King, "Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm," *International Journal of Production Research*, vol. 18, no. 2, pp. 213–232, 2007.

[64] A. S. Carrie, "Numerical taxonomy applied to group technology and plant layout," *International Journal of Production Research*, vol. 11, no. 4, pp. 399–416, 1973.

[65] C. Mosier and L. Taube, "Weighted similarity measure heuristics for the group technology machine clustering problem," *Omega*, vol. 13, no. 6, pp. 577–579, 1985.

[66] K. R. Kumar, A. Kusiak, and A. Vannelli, "Grouping of parts and components in flexible manufacturing systems," *European Journal of Operational Research*, vol. 24, no. 3, pp. 387–397, 1986.

[67] W. J. Boe and C. H. Cheng, "A close neighbour algorithm for designing cellular manufacturing systems," *International Journal of Production Research*, vol. 29, no. 10, pp. 2097–2116, 1991.

[68] M. P. Chandrasekharan and R. Rajagopalan, "GROUPABIL1TY: an analysis of the properties of binary data matrices for group technology," *International Journal of Production Research*, vol. 27, no. 6, pp. 1035–1052, 2007.

[69] K. R. Kumar and A. Vannelli, "Strategic subcontracting for efficient disaggregated manufacturing," BEBR faculty working paper; no. 1252, University of Illinois, Champaign, IL, USA, 1986.

[70] M. P. Chandrasekharan and R. Rajagopalan, "ZODIAC-an algorithm for concurrent formation of part-families and machine-cells," *International Journal of Production Research*, vol. 25, no. 6, pp. 835–850, 2007.

[71] C. Sur, S. Sharma, and A. Shukla, "Egyptian vulture optimization algorithm–A new nature inspired meta-heuristics for knapsack problem," in *Proceedings of 9th International Conference on Computing and Information Technology (IC2IT2013)*, pp. 227–237, Bangkok, Thailand, May 2013.

[72] A. Ritthipakdee, A. Thammano, N. Premasathian, and D. Jitkongchuen, "Firefly mating algorithm for continuous optimization problems," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 8034573, 10 pages, 2017.

[73] B. Almonacid, F. Aspée, R. Soto, B. Crawford, and J. Lama, "Solving the manufacturing cell design problem using the modified binary firefly algorithm and the egyptian vulture optimisation algorithm," *IET Software*, vol. 11, no. 3, pp. 105–115, 2017.

[74] B. Almonacid, F. Aspée, R. Soto, B. Crawford, and J. Lama, *Solving Manufacturing Cell Design Problem Using Modified Binary Firefly Algorithm and Egyptian Vulture Optimization Algorithm*, IET Software, Wales, UK, 2016.

[75] R. Soto, B. Crawford, B. Almonacid, and F. Paredes, "Efficient parallel sorting for migrating birds optimization when solving machine-part cell formation problems," *Scientific Programming*, vol. 2016, Article ID 9402503, 39 pages, 2016.

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 1934575, 16 pages
https://doi.org/10.1155/2019/1934575

Research Article

Motion Planning of Autonomous Mobile Robot Using Recurrent Fuzzy Neural Network Trained by Extended Kalman Filter

Qidan Zhu, Yu Han, Peng Liu, Yao Xiao, Peng Lu, and Chengtao Cai

College of Automation, Harbin Engineering University, Harbin 15001, China

Correspondence should be addressed to Yu Han; hanyu@hrbeu.edu.cn

Received 10 October 2018; Revised 7 December 2018; Accepted 2 January 2019; Published 29 January 2019

Guest Editor: Fevrier Valdez

Copyright © 2019 Qidan Zhu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a novel motion planning method for an autonomous ground mobile robot to address dynamic surroundings, nonlinear program, and robust optimization problems. A planner based on the recurrent fuzzy neural network (RFNN) is designed to program trajectory and motion of mobile robots to reach target. And, obstacle avoidance is achieved. In RFNN, inference capability of fuzzy logic and learning capability of neural network are combined to improve nonlinear programming performance. A recurrent frame with self-feedback loops in RFNN enhances stability and robustness of the structure. The extended Kalman filter (EKF) is designed to train weights of RFNN considering the kinematic constraint of autonomous mobile robots as well as target and obstacle constraints. EKF's characteristics of fast convergence and little limit in training data make it suitable to train the weights in real time. Convergence of the training process is also analyzed in this paper. Optimization technique and update strategy are designed to improve the robust optimization of a system in dynamic surroundings. Simulation experiment and hardware experiment are implemented to prove the effectiveness of the proposed method. Hardware experiment is carried out on a tracked mobile robot. An omnidirectional vision is used to locate the robot in the surroundings. Forecast improvement of the proposed method is then discussed at the end.

1. Introduction

In recent decades, unmanned ground mobile robots have been widely applied in various areas of both indoor and outdoor environments such as industry, mine, museum, port, or some dangerous places for their excellent maneuverability [1–3]. Research about navigation which can fully reflect artificial intelligence and automatic ability of unmanned ground mobile robots has been an attractive topic for a long time [4]. In order to achieve navigation, an effective motion planner should be designed [5]. Among the existing solutions, planning techniques were classified in two groups: nonheuristic methods and heuristic methods [6]. The most important nonheuristic methods consist of the potential field method (PFM) [7, 8], sampling-based planner (SBP), and interpolating curve method (ICM). PFM and SBP do not produce optimal paths and tend to be locked in some local minima [9]. ICM generates trajectories by constructing and inserting a new set of states considering reference points, i.e., a given set of way points, which cannot deal with dynamic surroundings well [10]. In order to solve the problem mentioned above, heuristic approaches are proposed. The most popular heuristic methods contain hybrid-heuristics A*, neural network (NN), fuzzy logic (FL), genetic algorithm (GA), particle swarm optimization (PSO), etc. Contrasted to nonheuristic methods, heuristic methods are more intelligent and advanced to deal with complex problems [11]. However, the serious disadvantage is the necessary learning phase. Much online or offline computation is needed. So, a more efficient method should be proposed.

Fuzzy logic is well suited for programming mobile robot's motion for its accurate calculation capability and inference capability under uncertainty [12]. Many researchers have implemented this method to address the navigation problem of unmanned mobile robots. Wang and Liu [13] proposed a real-time fuzzy logic-based navigation strategy in unknown environments. The

proposed approach employs a grid-based map that can record environment information and experience. However, the method focuses on building a map that is computationally expensive. The structure of fuzzy logic is so simple that it cannot deal with complex problems. Neural network is widely used due to its strong nonlinear approximation capability and self-learning capability [14]. Many researches have been done on the feedforward multilayer perception neural network [15]. However, feedforward NN methods require multilayer structures with a lot of neurons to represent dynamical responses. This leads to divergence and is time-consuming [16]. The weights of them are updated without considering the internal information and are sensitive to the training data. So, recurrent neural network attracts more attention for its superior dynamic capability. Recently, fuzzy logic and recurrent neural network structure are combined to form a new structure, i.e., recurrent fuzzy neural network (RFNN). Many approaches have been proposed by using RFNN and have shown superior performances. In [17], Juang et al. proposed a recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing. The structure forms a local internal feedback loop by feeding the firing strength of each rule back to itself. All rules are trained online via structure and parameter learning. Lin et al. [18] proposed an interactively recurrent fuzzy neural network for prediction and identification of dynamic systems. Their method is the same with Juang's method but employs a functional link neural network (FLNN) to the consequent part of fuzzy rules. The mapping ability is promoted. Although the concept of RFNN is investigated in detail, it has not been used in practical navigation well. For example, in [19], optimization of the result is not considered.

Back propagation (BP), evolutionary algorithm, and extended Kalman filter (EKF) are the three most popular training methods of supervised learning algorithms. In [20], the BP method is used to train the fuzzy neural network to achieve task planning and action selection of mobile robots. But, it needs data base and is trained offline. To apply RFNN to real-time nonlinear programs, an effective training method should be adopted. The extended Kalman filter is famous for its training efficiency and accuracy [21]. Rubio and Yu [22] applied EKF to train state-space recurrent neural networks, and identification of the nonlinear system is realized. And, the Lyapunov method is used to prove the stability of system. Wang and Huang [23] developed an effective RNN training approach based on EKF by using a controllable training convergence on the basis of Rubio. By adapting two artificial training noise parameters, i.e., the covariance of measurement noise and covariance of process noise, performance of EKF is improved. But, the proposed method is used in RNN instead of RFNN. The EKF algorithm possesses good online learning ability. Therefore, it is suitable for training RFNN to program the autonomous mobile robot's motion and achieving navigation.

Depending on the analysis above, the main contribution of this paper is that a real-time program strategy in unknown dynamic surroundings is proposed, i.e., without any previous offline computation. And, the optimal motion is generated in a free-space, i.e., without previous map information. A simple but effective RFNN structure is designed. A modified extended Kalman filter method is used to train RFNN in real time. An autonomous mobile robot is driven to reach the target and avoid obstacles. In the EKF training algorithm, target and obstacle constraints in practical situation are considered. Robustness of the proposed method against disturbances is discussed. Then, a numeric nonlinear optimization method and an update strategy are designed to guarantee robust optimization of the prediction. Besides the simulation experiment, our method is also evaluated on a real tracked mobile robot. An omnidirectional vision is used to locate the robot by using artificial landmarks on the basis of our previous work.

The rest of this paper is structured as follows. Section 2 illustrates the modelling of the autonomous mobile robot surrounded by the target and obstacles. Section 3 describes the planner in detail, including RFNN structure, EKF online learning algorithm, and convergence analysis. Section 4 constructs the cost function and update strategy to guarantee optimization. Section 5 is simulation and hardware experiment results. Finally, Section 6 concludes the proposed scheme.

2. Model of Autonomous Mobile Robot

Some programming methods ignore kinematic constraints [24]; thus, the stability of the system in practical situation cannot be guaranteed. An additional algorithm needs to be designed to smooth the trajectories. This leads to more computation cost. And, control effect of driving actual mobile robots to track the trajectory is not good. In order to get good dynamics performance in the tracking process, a natty kinematics of mobile robot is considered in motion planning.

The autonomous mobile robot favored in this paper is a kind of tracked mobile platform. There are two caterpillar tracks independently driven by actuators for the mobile robot's motion, and they are placed symmetrically on both sides of the mobile robot. The kinematics can be illustrated as shown in Figure 1. C is the geometry center of a mobile platform. $2b$ is the length between two tracks. $\{O, X, Y\}$ is the global coordinate frame, and $\{C, X_C, Y_C\}$ is the local coordinate frame.

Assume that the unmanned mobile robot is made up of a rigid frame equipped with nondeformable caterpillar tracks. There is no slip between tracks and actuator gears. And, they are moving on a horizontal plane only in the direction normal to the axis of driving.

The posture is represented by three variables as

$$\mathbf{p} = [x, y, \theta]^T, \tag{1}$$

where x and y are the coordinates of the center point C in $\{O, X, Y\}$ and θ is the angle of its heading direction X_C taking counterclockwise from the X-axis. The motion state is

$$\mathbf{q} = [v, w]^T, \tag{2}$$

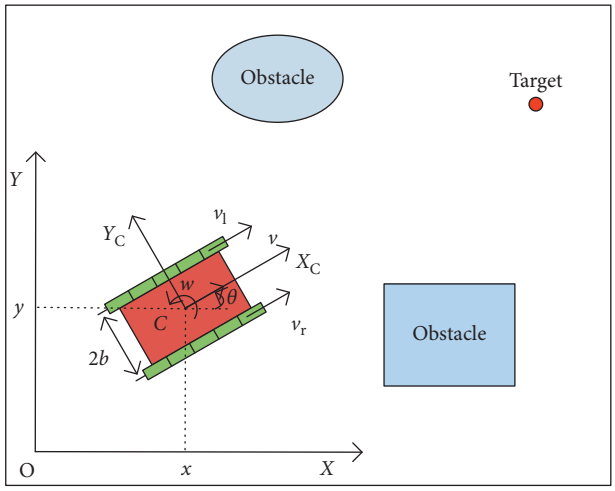


Figure 1: Model of the autonomous mobile robot in dynamic surroundings.

where v is the linear velocity and w is the angular velocity of the mobile robot. Then, $\mathbf{p}$ and $\mathbf{q}$ can be associated by the following equitation:

$$\mathbf{p} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{q}. \tag{3}$$

Thr autonomous mobile robot is motivated by a pair of independent caterpillar tracks, so

$$v = \frac{(v_r + v_l)}{2}, \tag{4}$$

$$w = \frac{(v_r - v_l)}{2b}, \tag{5}$$

$$R = \frac{b(v_r + v_l)}{(v_r - v_l)}, \tag{6}$$

where v_r is the right linear velocity, v_l is the left linear velocity, and R is the rotation radius of the mobile robot.

A simple discrete-time kinematic model is used in this paper to illustrate the moving process. The difference equation can be illustrated as

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \Delta t \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix} \mathbf{q}_k. \tag{7}$$

The positions of the target and obstacles in the global coordinate frame need to be transformed to the local frame. The transformation matrix is

$$\mathbf{T} = \begin{bmatrix} \cos\theta & \sin\theta & -x\cos\theta - y\sin\theta \\ -\sin\theta & \cos\theta & x\sin\theta - y\cos\theta \\ 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

3. Motion Planner Based on RFNN

A nonlinear program strategy is shown in Figure 2. It is made up of four parts such as coordinate transformation, RFNN structure, unmanned mobile robot model, and online learning algorithm. The coordinate transformation part establishes the environment map. Target and obstacles' information is then collected. The RFNN structure generates desired velocities of the mobile robot. It is trained by the extended Kalman filter that makes up the online learning algorithm. Detailed information of each part is shown in Figure 2.

3.1. RFNN Structure. A simple recurrent fuzzy neural network is designed in this section for the purpose of improving computation efficiency. Considering that the navigation system is multiinput multioutput, RFNN is made up of five layers as shown in Figure 3. The structure is first used in our previous work [25]. In this paper, detailed information of the structure and training progress is introduced. Convergence is analyzed. Furthermore, the original structure is improved in this paper to achieve nonlinear motion planning.

3.1.1. Layer 1 (Input Layer). Only the current state is traded as the input in this layer. $\mathbf{s} = (s_1, s_2, s_3, s_4) = (d_g, d_o, \theta_g, \theta_o)$ is chosen as the input, so there are 4 nodes transmitting the input variables to the next layer directly. d_g is the distance between mobile robot and goal. d_o is the distance between mobile robot and the nearest obstacle. θ_g is the angle between mobile robot's front direction and goal. θ_o is the angle between mobile robot's front direction and the nearest obstacle.

3.1.2. Layer 2 (Membership Layer). Each node in this layer performs a membership function. In this paper, the input of the membership layer is s_i $(i = 1, 2, 3, 4)$. The membership function is defined with Gaussian MF as follows:

$$\text{net}_j\left(s_i^j\right) = -\frac{\left(s_i^j - c_i^j\right)^2}{\left(\sigma_i^j\right)^2}, \tag{9}$$

$$u_i^j\left[\text{net}_j\left(s_i^j\right)\right] = \exp\left[\text{net}_j\left(s_i^j\right)\right], \tag{10}$$

where $\exp[\cdot]$ is the exponential function and c_i^j and σ_i^j (j is the number of the linguistic variables with respect to each input) are the mean and standard deviation of the Gaussian function, respectively. The values of them are initially set via expert experience before the strategy begins.

According to the actual occasion, d_g (d_o) can be divided into far and near depending on the distance between mobile robot and goal (the nearest obstacle). θ_g (θ_o) can be divided into left and right depending on goal's position (the nearest obstacle's position) related to the mobile robot's front direction. So, there are two linguistic variables of each input.

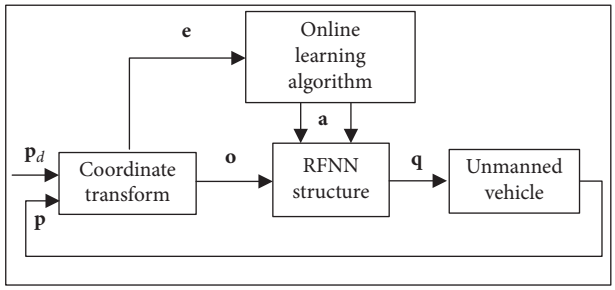


Figure 2: Nonlinear program strategy based on RFNN.

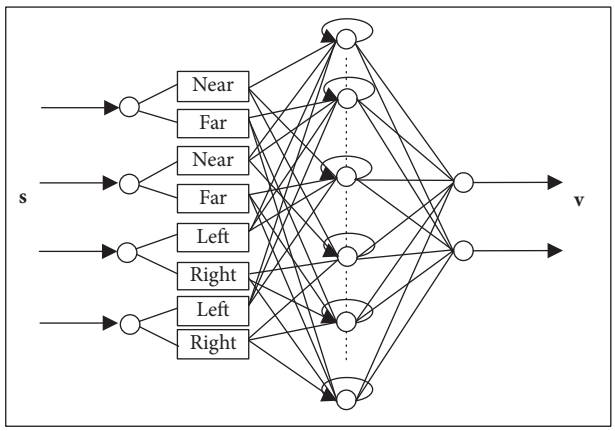


Figure 3: Structure of RFNN.

3.1.3. Layer 3 (Fuzzy Rule Layer). Each node in this layer corresponds to one fuzzy rule. As shown in Figure 3, each input has two membership values. Hence, there are 16 fuzzy rules.

The firing strength of each rule at the current step is determined by the outputs of layer 2 through an AND operator. The result of each rule is calculated as follows:

$$f_n = \prod u_i^j, \quad (i = 1, 2, 3, 4,\ j = 1, 2,\ n = 1, 2, \ldots, 16). \tag{11}$$

Moreover, a local internal feedback with a time delay is added to each node of this layer forming a recurrent frame. The mathematical form is described as

$$\psi_{nk} = (1 - \lambda_n) f_n + \lambda_n \psi_{n(k-1)}, \tag{12}$$

where λ_n is the constant representing weight of a self-feedback loop and $\psi_{n(k-1)}$ indicates the output of layer 3 in the previous time step.

3.1.4. Layer 4 (Consequent Layer). This layer describes a linear combination of functions in the consequent part, and each node is called the consequent node. According to the definition of TSK fuzzy rules, weight $w_o = (w_{o1}, w_{o2}, \ldots, w_{on})$ can be obtained:

$$w_{on} = a_{on}^1 s_1 + a_{on}^2 s_2 + \cdots + a_{on}^i s_i. \tag{13}$$

For the purpose of simplicity calculation, it is assumed that $a_{on} = a_{on}^1 = \cdots = a_{on}^i$. So, $w_{on} = a_{on}(s_1 + s_2 + s_3 + s_4)$, and $o = 1, 2$.

The output of this layer is

$$\varphi_o = \sum_n w_{on} \psi_n. \tag{14}$$

3.1.5. Layer 5 (Output Layer). There are two nodes in this layer representing linear velocities of right and left caterpillar tracks, respectively. An activation function is set at each node:

$$v_o = \frac{\varphi_o}{\left(1 + \alpha \left|\varphi_o\right|\right)}, \tag{15}$$

where α is a constant.

3.2. Online Training Algorithm Based on EKF. The EKF training algorithm can be summarized as parallel EKF and parameter-based EKF [26]. In this paper, the parameter-based EKF in [23] is modified to learn weights of RFNN. Considering the practical condition in motion planning of an autonomous mobile robot, a Jacobian matrix is designed.

At time step k, the EKF function has the following form:

$$\begin{aligned} \mathbf{a}_{k+1} &= \mathbf{a}_k - \mathbf{K}_k \mathbf{e}_k, \\ \mathbf{e}_k &= \mathbf{o}_k - \mathbf{o}_{dk}, \\ \mathbf{K}_k &= \mathbf{P}_k \mathbf{O}_k \left(\mathbf{R}_k + \mathbf{O}_k^T \mathbf{P}_k \mathbf{O}_k\right)^{-1}, \\ \mathbf{P}_{k+1} &= \mathbf{Q}_k + \left(\mathbf{I} - \mathbf{K}_k \mathbf{O}_k^T\right) \mathbf{P}_k, \end{aligned} \tag{16}$$

where $\mathbf{a}_k = (a_{11}, \ldots, a_{1,16}, a_{21}, \ldots, a_{2,16})^T$ is the estimation of weights, $\mathbf{K}$ is the Kalman gain matrix, $\mathbf{e}$ is the estimation error, $\mathbf{o}_d$ is the desired value of $\mathbf{o}$ which is the observation vector, $\mathbf{O}$ is the orderly derivative matrix, $\mathbf{R}$ is the covariance matrix of the measurement error, $\mathbf{Q}$ is the covariance matrix of the process noise, $\mathbf{P}$ is the covariance matrix of the estimation error, and $\mathbf{I}$ is the identity matrix.

In order to achieve navigation, both distance and angle information need to be considered. So, the observation vector can be represented as $\mathbf{o} = (d_g, d_o, \theta_g, \theta_o)^T$.

Then, $\mathbf{O}$ can be calculated as

$$\mathbf{O} = \frac{\partial \mathbf{o}^T}{\partial \mathbf{a}}. \tag{17}$$

As the only one-step recurrence is considered here, we take d_g as an example to calculate $\mathbf{O}$. Then, it is the same to d_o, θ_g, and θ_o:

$$d_{gk} = \sqrt{(x_k - x_d)^2 + (y_k - y_d)^2} = \sqrt{x_{dk}'^2 + y_{dk}'^2}, \tag{18}$$

where $(x_d', y_d', 1)^T = T(x_d, y_d, 1)^T$ is the target position in the local frame.

Then,

$$\frac{\partial d_{gk}}{\partial \mathbf{a}} = \frac{\left[(x_k - x_d)\partial x_k / \partial \mathbf{a} + (y_k - y_d)\partial y_k / \partial \mathbf{a}\right]}{d_{gk}}, \tag{19}$$

and according to (7),

$$\frac{\partial x_k}{\partial \mathbf{a}} = \frac{\partial x_{k-1}}{\partial \mathbf{a}} + \cos\theta_{k-1}\frac{\partial v_{k-1}}{\partial \mathbf{a}} - v_{k-1}\sin\theta_{k-1}\frac{\partial \theta_{k-1}}{\partial \mathbf{a}},$$
$$\frac{\partial y_k}{\partial \mathbf{a}} = \frac{\partial y_{k-1}}{\partial \mathbf{a}} + \sin\theta_{k-1}\frac{\partial w_{k-1}}{\partial \mathbf{a}} + w_{k-1}\cos\theta_{k-1}\frac{\partial \theta_{k-1}}{\partial \mathbf{a}}, \quad (20)$$
$$\frac{\partial \theta_k}{\partial \mathbf{a}} = \frac{\partial \theta_{k-1}}{\partial \mathbf{a}} + \frac{\partial w_{k-1}}{\partial \mathbf{a}},$$

and according to (4), (5), and (24),

$$\frac{\partial v_{k-1}}{\partial \mathbf{a}} = \frac{\psi_{n(k-1)}\sum s_{k-1}}{2\left(1+\alpha\varphi_{o(k-1)}\right)^2},$$
$$\frac{\partial w_{k-1}}{\partial \mathbf{a}} = \pm\frac{\psi_{n(k-1)}\sum s_{k-1}}{2b\left(1+\alpha\varphi_{o(k-1)}\right)^2}, \quad (21)$$

where $o = 1, 2$ and $n = 1, 2, \ldots, 16$ corresponding to $\mathbf{a} = (a_{11}, \ldots, a_{1,16}, a_{21}, \ldots, a_{2,16})^T$. If $o = 1$, the results are positive. Otherwise, the results are negative.

If the distance between obstacles is long enough for the mobile robot to pass through without collision, d_o and θ_o will not be considered in EKF. However, if the distance is shorter than safe distance, the EKF algorithm should train the weights of RFNN to avoid collision. d_o and θ_o are then considered at this moment. The flow diagram is shown in Figure 4.

3.3. Convergence Analysis. In this section, we will prove that the EKF proposed in the above section is effective to RFNN in Section 3.1. And the designed Jacobian matrix $\mathbf{O}$ is reasonable and feasible.

As shown in Figure 2, observation vector $\mathbf{o}$ is a function of $\mathbf{v}$, i.e., $\mathbf{o} = \mathcal{O}(\mathbf{v})$. By calculating the first-order derivative of $\mathbf{a}$ in (21), fuzzy logic inference and weights learning process are clearly reflected in $\mathbf{O}$. For the parameter-based EKF, only weights are viewed as states to be estimated [23]. $\mathbf{o}$ is first expanded at optimal weights $\mathbf{a}_d$ as

$$\mathbf{o}_k = \mathcal{O}(\mathbf{a}_d) + (\mathbf{a}_k - \mathbf{a}_d)\frac{\partial \mathbf{o}}{\partial \mathbf{a}} + \boldsymbol{\xi}_k, \quad (22)$$

where $\boldsymbol{\xi}_k$ is the first-order approximation residue. The error of weights can be defined as $\mathbf{e}_{ak} = \mathbf{a}_k - \mathbf{a}_d$.

Then, the Lyapunov function is written as

$$E_k = \mathbf{e}_{ak}^T\mathbf{P}_k^T\mathbf{e}_{ak},$$
$$\Delta E_k = E_{k+1} - E_k. \quad (23)$$

Then,

$$\Delta E_k = \mathbf{e}_{a(k+1)}^T\mathbf{P}_{k+1}^{-1}\mathbf{e}_{a(k+1)} - \mathbf{e}_{ak}^T\mathbf{P}_k^{-1}\mathbf{e}_{ak} < \left[\mathbf{e}_{a(k+1)} - \mathbf{e}_{ak}\right]^T\mathbf{P}_k^{-1}\mathbf{e}_{ak} - \mathbf{e}_{a(k+1)}^T\left[\mathbf{P}_{k+1} - \mathbf{Q}_k\right]^{-1}\mathbf{P}_k\mathbf{O}_k\mathbf{B}_k^{-1}\boldsymbol{\xi}_k. \quad (24)$$

According to Jolly et al. [20], (24) becomes

$$\Delta E_k < \frac{-\left\|\mathbf{e}_k\right\|^2}{\mathbf{o}_k/m + r_k} + \frac{3\left\|\boldsymbol{\xi}_k\right\|^2}{r_k}, \quad (25)$$

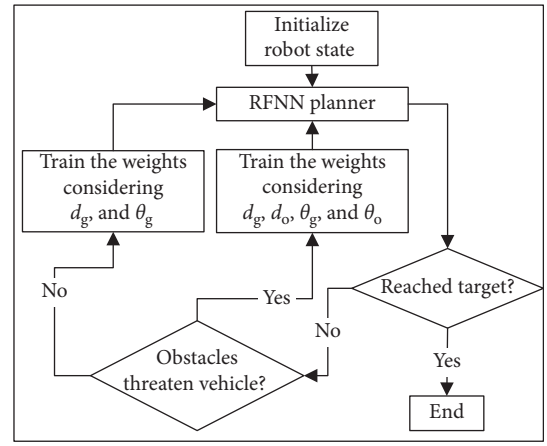


Figure 4: Flow diagram of EKF.

where $\mathbf{o}_k$ is the trace of $\mathbf{O}_k^T\mathbf{P}_k\mathbf{O}_k$. m is the dimension of $\mathbf{I}$ in (16), and r_k is a positive real number. As mentioned in the above section, the dimension of $\mathbf{O}$ changes depending on the position of obstacles related to the mobile robot. According to the calculate trace $\mathbf{o}_k$, the jumping change will not affect the stability of the training process.

From (25), we can see that the convergence of the training process is determined by $\mathbf{e}_k$, $\mathbf{o}_k$, and r_k. In order to guarantee $\Delta E_k < 0$, r_k should be set as

$$r_k > \frac{3\mathbf{o}_k\left\|\boldsymbol{\xi}_k\right\|^2}{m\left(\left\|\mathbf{e}_k\right\|^2 - 3\left\|\boldsymbol{\xi}_k\right\|^2\right)}. \quad (26)$$

If $\|\mathbf{e}_k\|^2 < 4\bar{\xi}_k^2$ ($\bar{\xi}_k \geq \|\boldsymbol{\xi}_k\|$), the error is bounded and the process is convergent.

If $\|\mathbf{e}_k\|^2 > 4\bar{\xi}_k^2$, the inequality will become

$$r_k > \frac{3\mathbf{o}_k}{m}. \quad (27)$$

If each element of $\boldsymbol{\xi}_k$ is of normal distribution, $\xi_{ik} \sim N(0, r_k)$, then,

$$\left\|\mathbf{e}_k\right\|^2 > 4\bar{\xi}_k^2 = 64r_k m, \quad (28)$$

where 99.99% ξ_{ik} are bounded. Then,

$$\frac{3\mathbf{o}_k}{m} < r_k < \frac{\left\|\mathbf{e}_k\right\|^2}{64m}. \quad (29)$$

r_k can be chosen as

$$r_k = \frac{\left(\left\|\mathbf{e}_k\right\|^2/64m + 3\mathbf{o}_k/m\right)}{2}. \quad (30)$$

Convergence of the training process is guaranteed, i.e., bounded $\mathbf{e}_k$.

4. Trajectory Optimization

As mentioned above, feasible trajectories are generated. But, these trajectories are always suboptimal and worthy of further improvement. So, in this section, the numerical optimization procedure is designed to obtain the optimal

trajectory. Considering the practical situation, power consumption, driving distance, and time are favored to determine optimization of the trajectory. A new variable object is established as $\mathrm{Tra} = \{\mathbf{p}, \mathbf{v}, \mathbf{w}\}$, indicating that each trajectory is represented by the mobile robot's state and linear and angular velocities.

Then, the objective function is structured as

$$J = \mathbf{W}_1 \sum_{k=1}^{N^l} \left\| \mathbf{p}_k^l - \mathbf{p}_{k-1}^l \right\|^2 + \mathbf{W}_2 \sum_{k=1}^{N^l} \left\| v_k^l - \overline{v}^l \right\|^2 + \mathbf{W}_3 \sum_{k=1}^{N^l} \left\| w_k^l - \overline{w}^l \right\|^2 + \mathbf{W}_4 N^l, \tag{31}$$

where $\mathbf{W}$ is the weight of each item. The first and last items in (31) represent moving distance and time, respectively. Furthermore, we want the motion of the mobile robot to be smooth in the dynamic environment. So, the second and third items are designed in (31). $\overline{v}$ and $\overline{w}$ are the mean values.

The autonomous mobile robot is driven by the target and needs to arrive at destination in limited area. During the process, obstacle avoidance is considered. Then, the target and obstacle constraints are

$$\left\| \mathbf{p}_\mathrm{f} - \mathbf{p}_\mathrm{t} \right\|_2 \leq \Delta_t, \tag{32}$$

$$\left\| \mathbf{p}_k - \mathbf{p}_\mathrm{o} \right\|_2 \geq \Delta_o, \tag{33}$$

where $\mathbf{p}_\mathrm{f}$ is the final state of the mobile robot, $\mathbf{p}_k$ is the state at each step, $\mathbf{p}_\mathrm{t}$ is the target state, and $\mathbf{p}_\mathrm{o}$ is the obstacle state. These are achieved according to (16), (17), and (21).

In the practical application, the linear and angular velocities of the mobile robot are limited, so

$$0 \leq \mathbf{q}_k \leq \mathbf{q}_\mathrm{max}, \tag{34}$$

which is achieved according to (9)–(15).

Then, the optimization problem becomes

$$\mathrm{Tra}^* = \arg\min_{\mathrm{Tra}} J. \tag{35}$$

If the dynamic environment can be predicted or motion of the obstacle is not too drastic, our method is able to predict optimal trajectory without supplement. This is analyzed in the simulation experiment below. However, the dynamic environment is unpredicted and motion of the obstacle is irregular. So, an extra algorithm should be designed.

In order to update the trajectory online, the detection of data in real time should be carried out. The following variable is established to measure changes in the environment:

$$\sigma = \left(\mathbf{o}_k^d - \mathbf{o}_k^m\right)^T \mathbf{W}_5 \left(\mathbf{o}_k^d - \mathbf{o}_k^m\right), \tag{36}$$

where $\mathbf{o}_k^d$ is the detection of the observation vector at each step and $\mathbf{o}_k^m$ is the memory value of the current trajectory. $\mathbf{W}_5$ is the matrix of weights. The upper bound is set

$$\sigma < \sigma_\mathrm{bound}. \tag{37}$$

5. Experiments

5.1. Motion Planning Based on RFNN. To prove the effectiveness of the proposed motion planning method, the simulation experiment is carried out in this section using the Matlab software. And, all experiments are performed on a computer with Intel i5 2.3 GHz processor and 8 GB RAM. The simulation area is limited in 10 m × 10 m. There contain obstacles and target. The autonomous mobile robot needs to reach the target and avoid obstacles. The detailed values of the variables that need to be set manually are listed in Table 1.

As illustrated in Figure 4, the navigation method proposed here is goal driven. The program strategy generates the trajectory guiding autonomous mobile robot to move from the starting point to the target. At each step, only the nearest position of the obstacle that threatens the mobile robot's safety is used in the training process of RFNN. So, the obstacle model is established using points at the edge as shown in Figure 5. Circle represents the dangerous region whose radius is related to the error term in (16). It is determined by the EKF algorithm's training speed. Distance between centers of adjacent circles depends on b. If it is too long, the model becomes invalid. If it is too short, the model is too compact to waste much time in computation. When a suitable distance is chosen, a sparse representation of the obstacle is established. Motion planning is first carried out in the static environment. The target and obstacles are supposed to be fully detected in real time.

The trajectories without the numerical optimization are shown in Figure 5. The two motion trajectories are listed here. Each mark on the path represents the planned position of the mobile robot at each step. Changes of motion are reflected clearly. The position error of the robot during the process is shown in Figure 6. When the robot moves close to the obstacle, it slows down at points A and B. This satisfies the actual requirement and guarantees safety. The corresponding linear velocity is shown at points A and B in Figure 7. The corresponding angular velocity is shown at points A and B in Figure 8. When the robot avoids the collision with the obstacle, it speeds up to shorten time to arrive the target. The corresponding linear velocity is shown at points C and D in Figure 7. The corresponding angular velocity is shown at points C and D in Figure 8. At points E and F, the robot slows down to arrive the target. The statistical analysis of the curvature is shown in Figure 9. It can be seen that the programmed motion is smooth and is fit to the actual action.

Information of the two trajectories is shown in Table 2. Distance, step, cost, and terminal error are chosen as evaluation criteria in this paper. In order to observe the state of RFNN during process, weights are listed in Tables 3 and 4. The weights are initially set in random. They keep changing during the process to drive the robot to the target and avoid obstacles. From $k = 40$ to $k = 50$, we can see that they are convergent at the destination.

Table 1: Initialization of the parameters.

b	0.2
$\mathbf{c}$	$\begin{bmatrix} 0 & 0 & -(\pi/2) & -(\pi/2) \\ 10 & 5 & \pi/2 & \pi/2 \end{bmatrix}$
$\boldsymbol{\sigma}$	$[2.5 \ 1 \ 1 \ 1]$
λ	0.3
α	0.3
$\mathbf{q}_{\max}$	$[1 \ 1]^T$
Δ_t	0.5
Δ_o	0.3
σ_{bound}	0.5
Δt	1

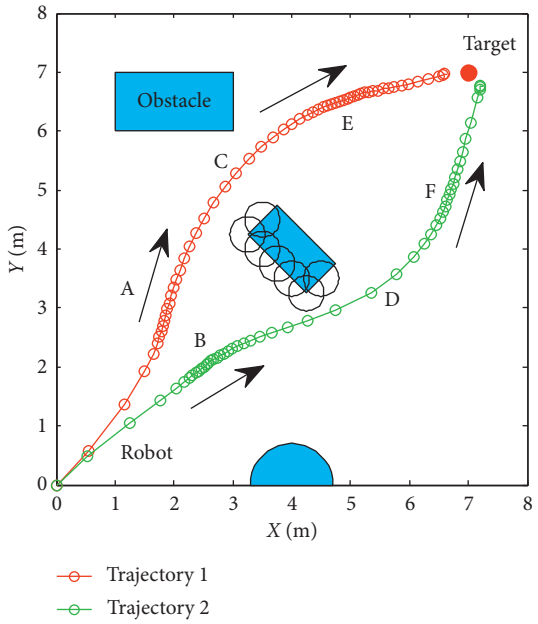


Figure 5: Motion trajectory of the mobile robot.

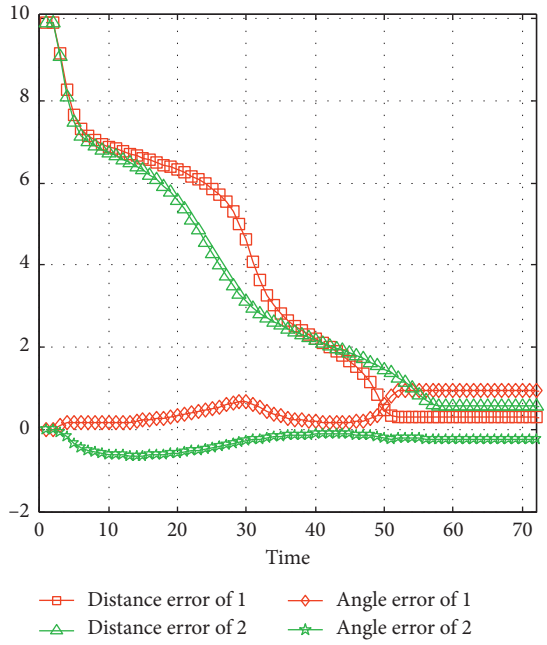


Figure 6: Distance and angle error of the mobile robot.

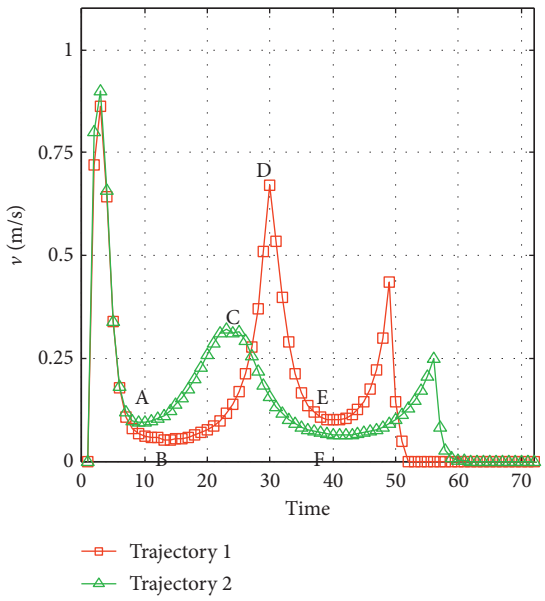


Figure 7: Linear velocity of the mobile robot.

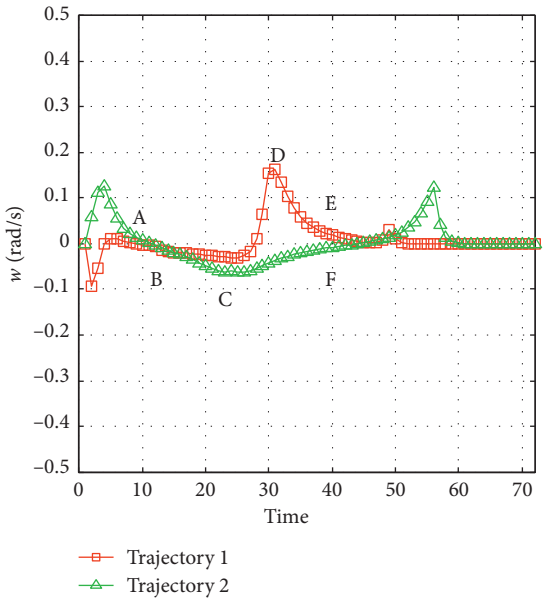


Figure 8: Angular velocity of the mobile robot.

5.2. Optimization of Trajectory. In order to illustrate the optimization of the solution, the trajectories are generated as shown in Figure 10. Among all trajectories, only the blue one is generated considering (31)–(35). The corresponding number in Table 5 is eight. The left nine cyan trajectories are generated randomly. Detailed information is listed in Table 5. By suitably choosing weights in (31), moving distance, step, and smoothness of trajectory are taken into comprehensive consideration. It can be seen that more computation is needed to generate optimal trajectory. The terminal error can be reduced by setting the target constraint in (32). In this paper, it is set as 0.5. As shown in the boxplot of velocity in Figures 11 and 12, the values of the eighth one are more concentrated and outliers are smaller.

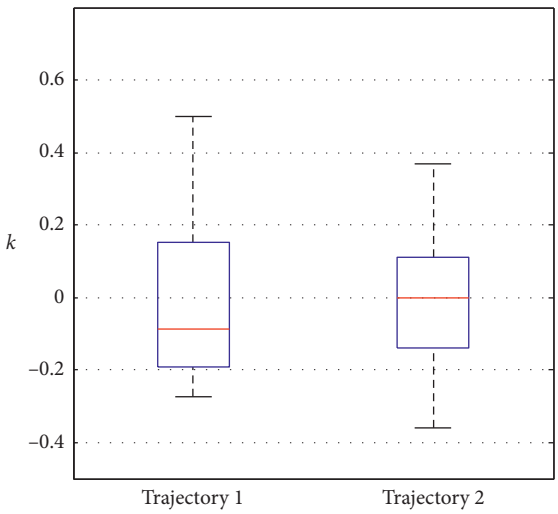


Figure 9: Statistical analysis of the curvature.

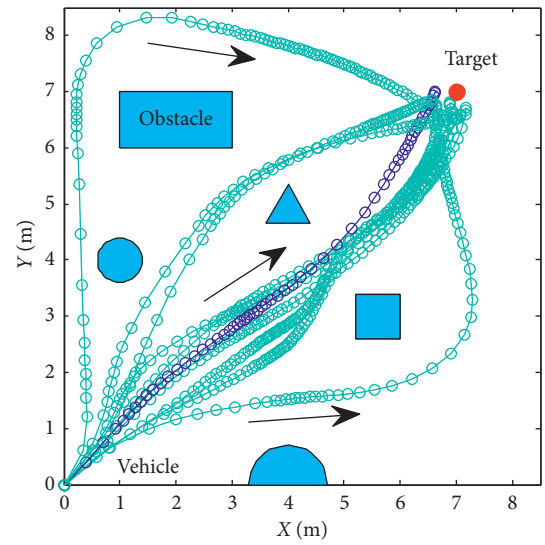


Figure 10: Motion trajectories of the mobile robot.

Table 2: Trajectory information.

	Distance (m)	Step	Cost (s)	Error (m)
Trajectory 1	10.20	62	0.228	0.389
Trajectory 2	10.44	59	0.347	0.303

Table 3: RFNN weights of trajectory 1.

$k = 1$	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 45$	$k = 50$
0.701	5.396	15.40	20.60	23.32	24.49	24.82
0.209	1.616	4.614	6.174	6.986	7.337	7.437
0.533	4.110	11.73	15.69	17.76	18.65	18.90
0.359	2.772	7.913	10.58	11.98	12.58	12.75
0.713	5.493	15.67	20.97	23.73	24.93	25.26
0.007	0.053	0.152	0.204	0.230	0.242	0.245
0.639	4.926	14.06	18.81	21.28	22.35	22.66
0.126	0.972	2.775	3.713	4.202	4.413	4.473
0.774	5.965	17.02	22.78	25.78	27.07	27.44
0.537	4.138	11.81	15.80	17.88	18.78	19.03
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.220	1.910	5.496	8.076	9.053	9.406	9.448

Table 4: RFNN weights of trajectory 2.

$k = 1$	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 45$	$k = 50$
0.101	0.882	2.538	3.730	4.181	4.344	4.363
0.756	6.545	18.82	27.66	31.01	32.22	32.36
0.866	7.494	21.55	31.67	35.50	36.89	37.05
0.996	8.624	24.80	36.45	40.86	42.45	42.64
0.098	0.852	2.451	3.601	4.037	4.195	4.213
0.172	1.493	4.295	6.312	7.075	7.351	7.383
0.484	4.192	12.05	17.72	19.86	20.64	20.73
0.937	8.110	23.32	34.28	38.42	39.92	40.10
0.509	4.409	12.68	18.63	20.89	21.70	21.80
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.466	4.036	11.61	17.06	19.12	19.87	19.95

The performance of the eighth motion trajectory is better than that of others, which is reflected by the curvature in Figure 13.

Table 5: Information of trajectories.

	Distance (m)	Step	Cost	Terminal error
1	9.936	76	0.091	0.439
2	14.72	87	0.059	0.483
3	9.866	96	0.022	0.402
4	10.01	74	0.038	0.221
5	12.13	62	0.129	0.471
6	9.744	75	0.102	0.384
7	10.15	78	0.067	0.331
8	9.683	58	1.342	0.369
9	10.36	50	0.089	0.463
10	9.957	65	0.011	0.451

5.3. Comparison with Other Methods. As mentioned above, there are many programming methods. In order to prove our method's effectiveness, other methods are compared here. OPTI supports for solving optimal problems and consists of popular optimization solvers. So, a numerical planner using the nonlinear program solvers in OPTI is designed. Kinematic constraint, target, and obstacle constraints are considered in the process. Furthermore, the classical A* method which is carried out using the grid map is also taken into comparison.

The results are shown in Figure 14. The numerical method is continuous in motion for the reasons that it plans the control command instead of the mobile robot's position. And, the state of the mobile robot is space free. A* plans only the position of the mobile robot and the path is not smooth. An extra controller considering kinematic constraints of the mobile robot should be added to track the path. Because it is based on the grid map, the planned path can be unsuitable for the robot to follow. A big curvature makes performance bad, such as the orange circle area in Figure 14. Although the performance of the numerical method is better, it takes up more computation time than A* as listed in Table 6. Compared to these two kinds of methods, our planner is continuous in motion and the generated trajectory is smooth. Computation time is much shorter than the

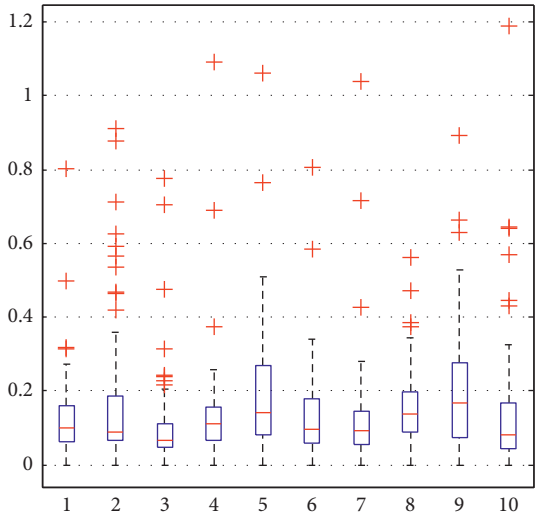


Figure 11: Statistical analysis of velocity.

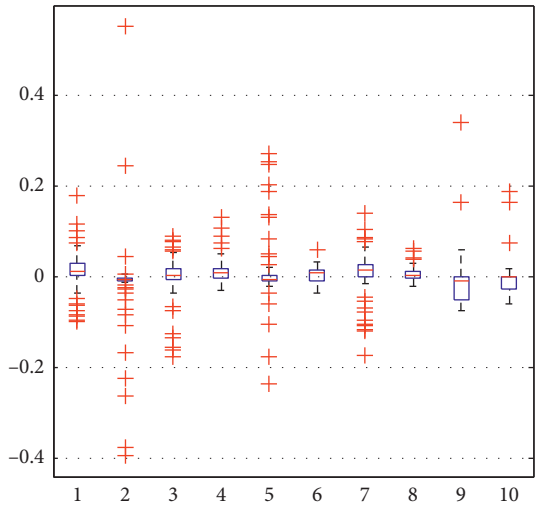


Figure 12: Statistical analysis of angular velocity.

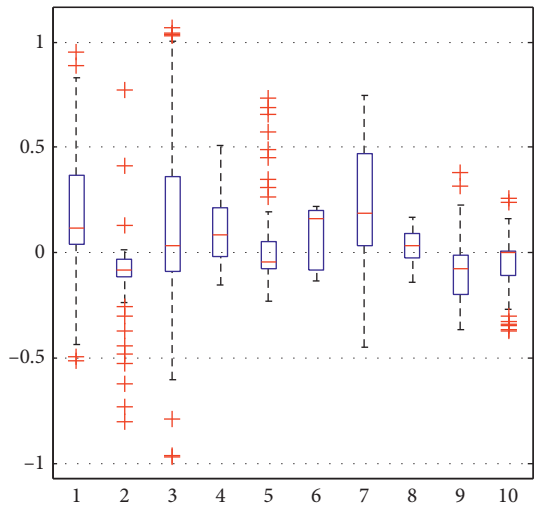


Figure 13: Statistical analysis of the curvature.

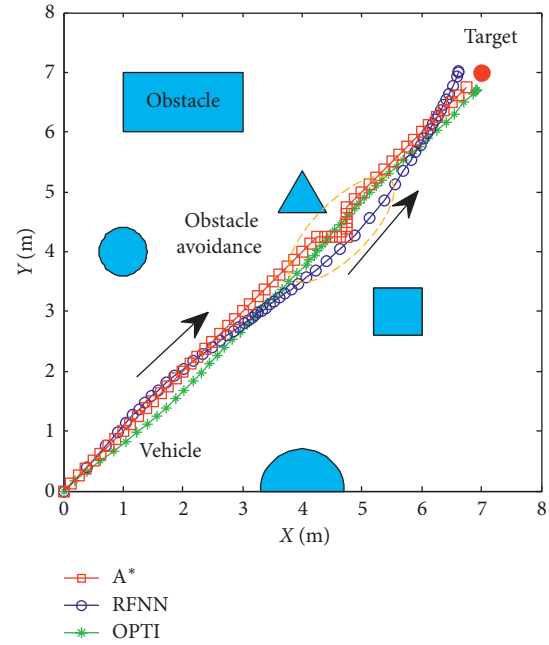


Figure 14: Motion trajectories of the mobile robot.

numerical method to achieve the same performance, but it is a little longer than A*. The qualitative comparison is illustrated in Figure 15.

5.4. Robustness of RFNN Planner. In this paper, we want to realize real-time motion planning. So, robustness has to be guaranteed. The weights of RFNN are trained online. Performance depends on initial weights. After initial weights are ensured, trajectory is fixed under the current condition. However, in the practical situation, dynamic environment and perception inaccuracy influence performance of motion planning. So, in this section, influence of these factors to our planning method is introduced.

In order to prove the robustness of RFNN, the FNN-based planner is compared in this section first. Initial weights $\mathbf{a}_0$ of RFNN and FNN are both set as [0.1057, 0.1420, 0.1664, 0.6209, 0.5737, 0.0520, 0.9312, 0.7286, 0.7378, 0.0634, 0.8604, 0.9344, 0.9843, 0.8589, 0.7855, 0.5133, 0.1776, 0.3985, 0.1339, 0.0308, 0.9391, 0.3013, 0.2955, 0.3329, 0.4670, 0.6481, 0.0252, 0.8422, 0.5590, 0.8540, 0.3478, 0.4460]. It is supposed that there is a perception error. In the actual motion period, the obstacles' position is biased contrasted to the prediction period. The performance is shown in Figure 16. Motion with RFNN under the perception error is the same as that under prediction. The recurrent frame in RFNN structure improves the robustness of the planner. On the contrary, motion with FNN is the biased contrasted prediction. This can cause dangerous in the actual situation.

To introduce our method's robustness in dynamic surroundings, the experiment is carried out as shown in Figure 17. Initial weight $\mathbf{a}_0$ is set as [0.4401, 0.6305, 0.2144, 0.6398, 0.9684, 0.6972, 0.8841, 0.9268, 0.9452, 0.5178, 0.1781, 0.8219, 0.0489, 0.1291, 0.9319, 0.2809, 0.9441, 0.6843, 0.6741, 0.3766, 0.8681, 0.5585, 0.3033, 0.8492, 0.7845, 0.9706, 0.9782,

Table 6: Trajectory information.

	Distance (m)	Step	Cost (s)	Error (m)
Opti	9.677	60	37.891	0.328
A*	9.838	62	0.512	0.354
RFNN	9.683	58	1.342	0.369

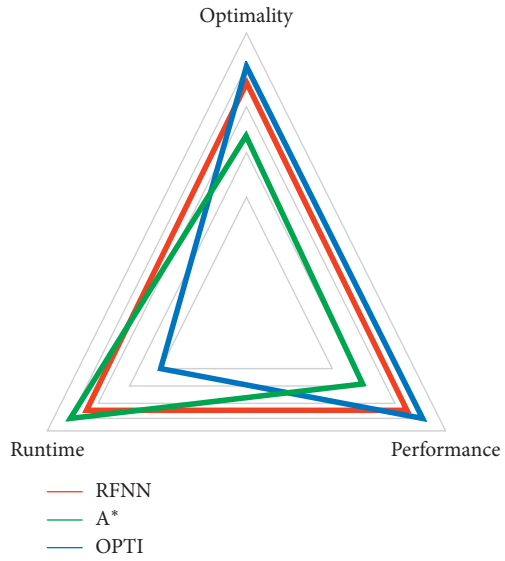


Figure 15: Qualitative comparison.

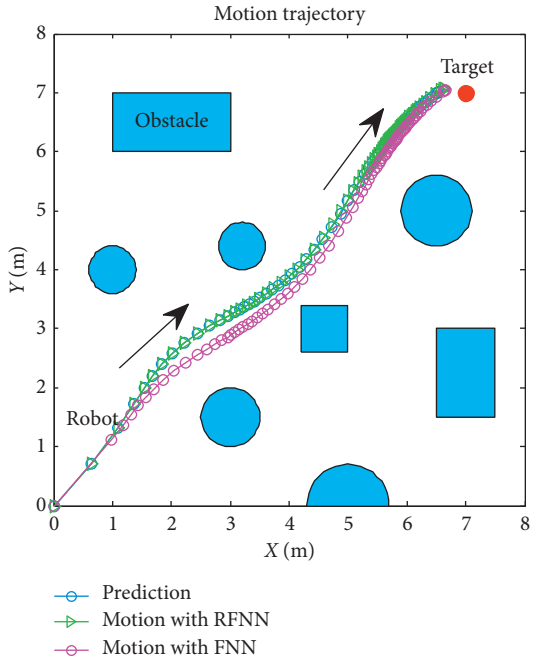


Figure 16: Qualitative comparison.

0.6032, 0.0149, 0.2569, 0.3101, 0.2272]. We assume that surroundings change to states 2 and 3. Corresponding trajectories are green and yellow, respectively. The changes of RFNN weights of each condition are, respectively, illustrated in Figure 18. Detailed information of each trajectory is shown in Table 7. It proves that our method

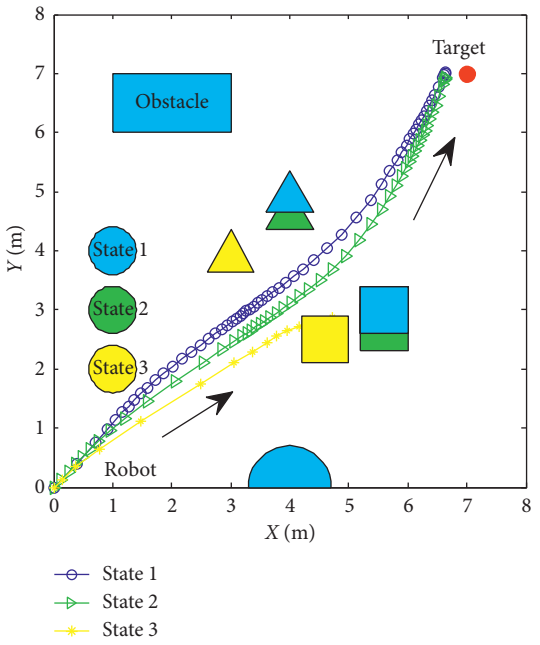


Figure 17: Motion trajectory of the mobile robot.

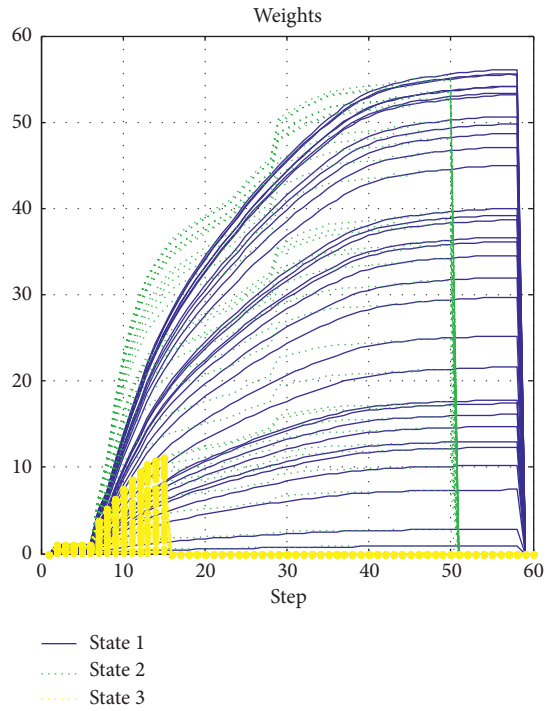


Figure 18: Motion trajectory of the mobile robot.

Table 7: Trajectory information.

	Distance (m)	Step	Error (m)	State
State 1	9.683	58	0.369	Success
State 2	9.699	50	0.374	Success
State 3	4.714	15	4.723	Fail

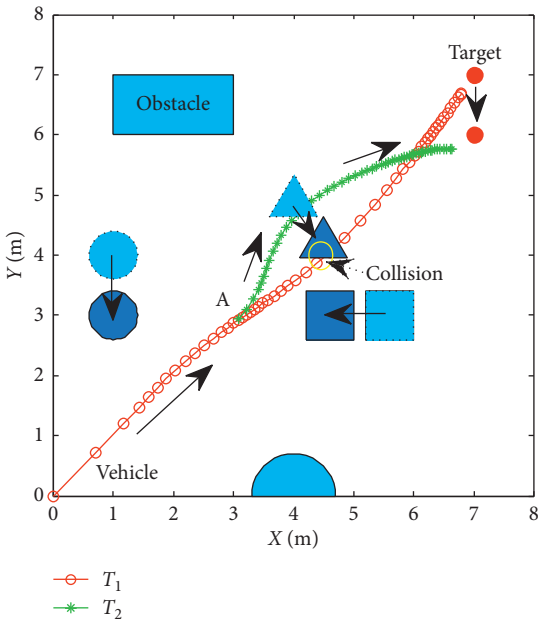


Figure 19: Motion trajectory of the mobile robot.

possesses robustness in the dynamic environment with little amplitude changes. It loses efficacy if the changes are drastic. So, the dynamic update mechanism of initial weights of RFNN should be designed necessarily.

5.5. Update of Motion Trajectory. Effectiveness of computation, continuity of motion, and smoothness of the predicted trajectory make the proposed planning method feasible and stable to update online. By applying update strategy to RFNN, robust optimality of prediction is guaranteed during the whole process. The performance is shown in Figure 19. The update period is ΔT which depends on σ. Optimal trajectory during T_1 is generated at the beginning. The position of the target and obstacles keeps changing. If the autonomous mobile robot keeps moving according to trajectory 1, collision will happen. Threshold settled in (37) is reflected in Figure 20. Then, weights of RFNN are retrained as shown in Figure 21. Optimal trajectory during T_2 is generated considering the updated position of obstacles and target. Then robot changes the route at point A in Figure 19. Corresponding changes of linear velocity is shown in Figure 22 at point A. Contrasted to linear velocity, change of angular velocity is drastic in Figure 23. Because robot needs to avoid collision. Position error during whole process is shown in Figure 24. Before A point, red lines represent robot's position error. After point A, the green line represents the robot's position error. More computation periods can be added until the mobile robot reaches the target. This guarantees the real-time optimization.

5.6. Hardware Application Based on Omnidirectional Vision. In order to examine the effectiveness of our planning method, we also implement it to the realistic mobile robot in Figure 25. The visual navigation has attracted many researchers [27]. A catadioptric omnidirectional camera is

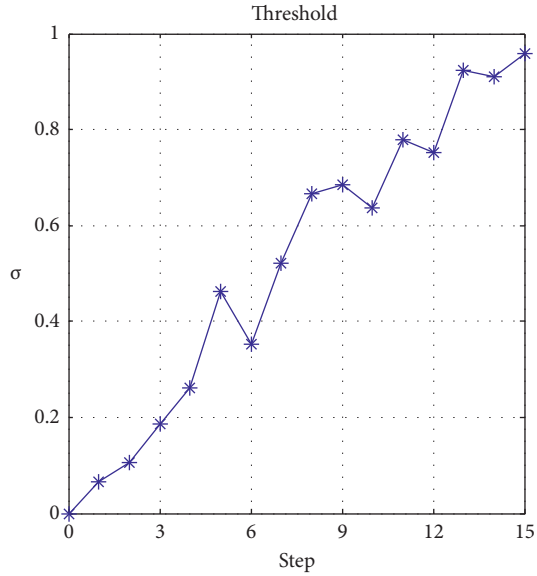


Figure 20: Changes of σ.

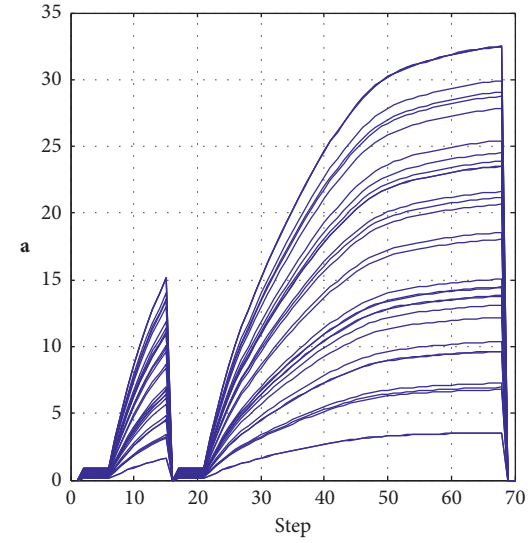


Figure 21: Changes of RFNN weights.

popularly used in recent years for its advantage of wide field of view [28]. It can capture horizontal field of view in a single image. So, the vision system based on the catadioptric omnidirectional camera is carried on the mobile robot to percept surroundings in this paper. Detailed imaging theory is introduced in our previous work [29, 30].

Although it is a binocular stereo vision system, we only use the lower camera of it because binocular stereo omnidirectional vision takes up much computation resource and runtime performance is not good in real-time application. When the mobile robot moves fast, binocular structure swings violently. In order to use monocular vision to achieve location, artificial landmarks are designed and SURF (speeded-up robust features) are extracted to guarantee rotation and scale invariance. The features of simple shapes are extracted firstly as shown in Figure 26. The landmarks are then designed using these shapes and are placed on the wall

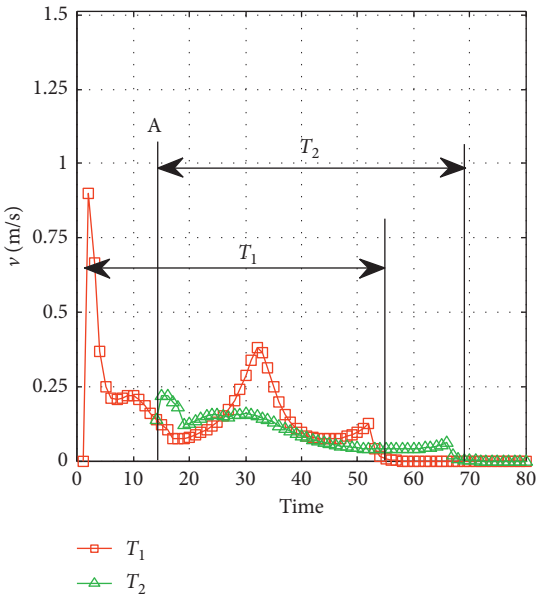


Figure 22: Linear velocity of the mobile robot.

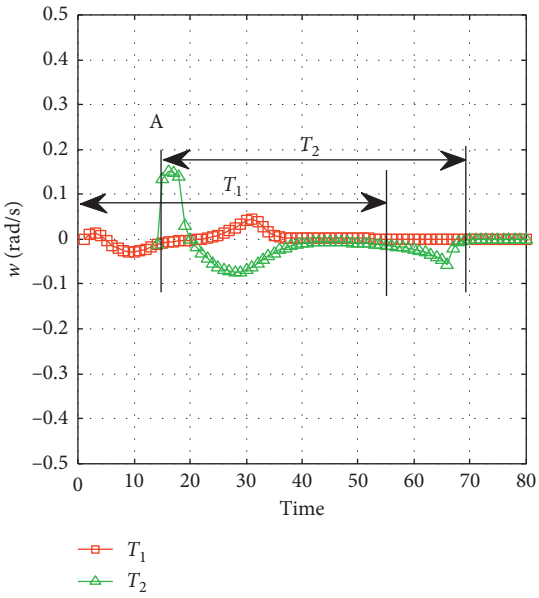


Figure 23: Angular velocity of the mobile robot.

forming the location system as shown in Figure 25. Detailed introduction of the location method is introduced in [31, 32].

The planning method proposed in this paper is carried out, and the performance of mobile robot is shown in Figure 27. The obstacles in blue are known to the mobile robot, and the green one is unknown. The target position keeps changing during the process. There are totally three predictions. One is generated at the beginning in red. After the unknown obstacle is detected, prediction 2 in magenta is generated at point A. Prediction 3 is generated to lead the mobile robot to reach the final station of the target at the B point. The blue circles represent the actual motion of the

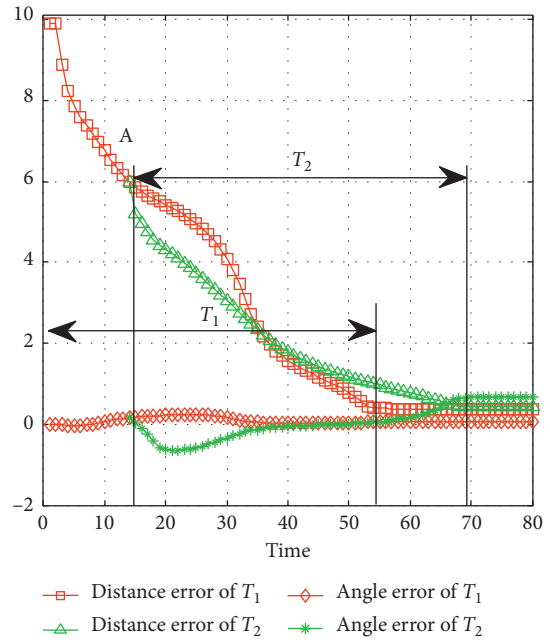


Figure 24: Error of the mobile robot.

Figure 25: Experiment environment with artificial landmarks.

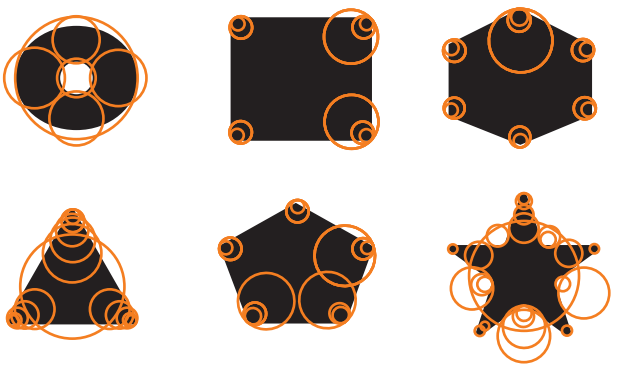


Figure 26: SURF of the sample landmarks.

mobile robot. Performance of the vision system is shown in Figure 28. Detailed information of the prediction and actual motion is listed in Table 8. During the process, the robot locates itself by tracking the three landmarks. The landmarks matching the precision curve in Figure 29 shows the percentage of correctly tracked frames for a range of distance

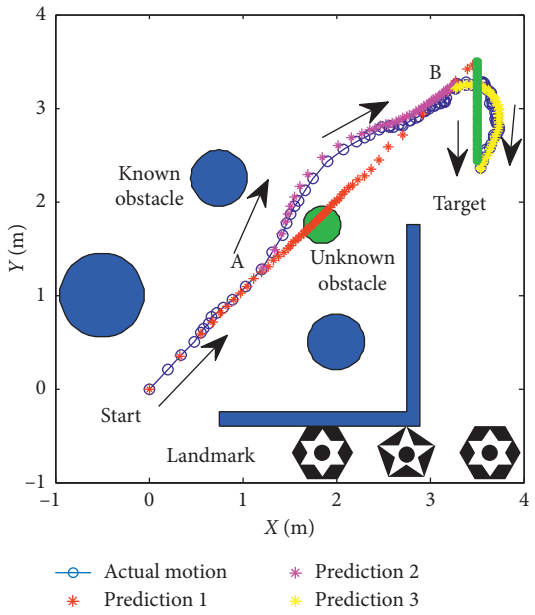


Figure 27: Motion trajectory of the mobile robot.

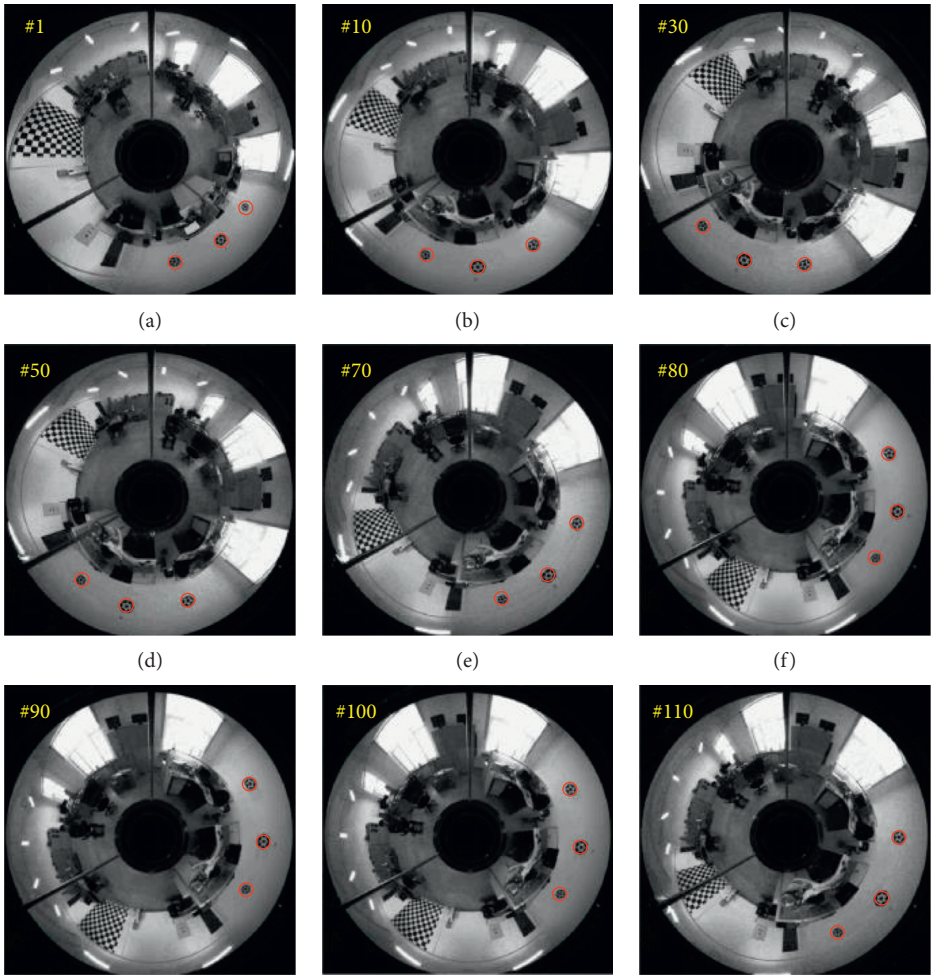


Figure 28: Monocular omnidirectional vision system.

Table 8: Trajectory information.

	Distance (m)	Step	Error (m)	Cost (s)
Prediction 1	4.747	57	0.155	1.261
Prediction 2	3.574	47	0.174	0.514
Prediction 3	1.274	52	0.123	0.081
Actual motion	6.569	103	0.099	110

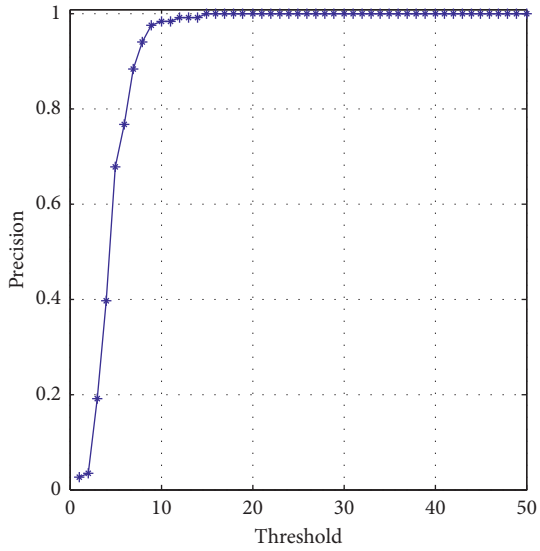


Figure 29: Landmark matching precision during process.

thresholds. Precision is 98.3% at 10 pixels. Update of trajectory is determined by settled threshold in (36) and (37). If σ is beyond threshold, weights of RFNN are retrained as mentioned above. Changes of σ are illustrated in Figure 30. σ is beyond threshold at A and B. It leads to update of RFNN weights as shown in Figure 31. Robot's motion is then changed in Figures 32 and 33. Figure 34 reflects the position error of the mobile robot. In actual application, maximum velocities are limited for safety. According to hardware experiment, our method's effectiveness is proved.

Among all experiment results above, the mobile robot's motion is drastic at the beginning. This is reflected by linear velocity. The reason is that weights of RFNN are generated randomly at the beginning. After learning using EKF, predicted motion of the mobile robot is stable. Experiments are carried out in limited area due to limited perception ability of the vision system. But, it can be popularized to large-scale navigation by combining the proposed program method with the topological mapping method introduced in [33].

6. Conclusion

By designing a simple RFNN structure and using an effective EKF-based learning method, a novel motion planning strategy is introduced. The greatest novelty is that the proposed method plans both motion and trajectory in real time. Robust optimization of solution is guaranteed. According to simulation and hardware experiments, effectiveness is proved. All the experiments in this paper are

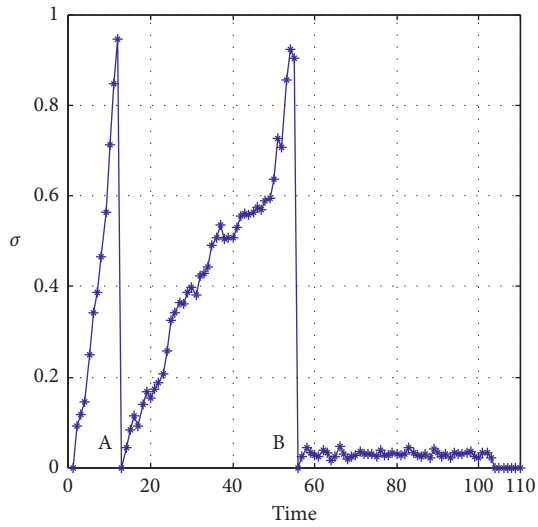


Figure 30: Changes of threshold during process.

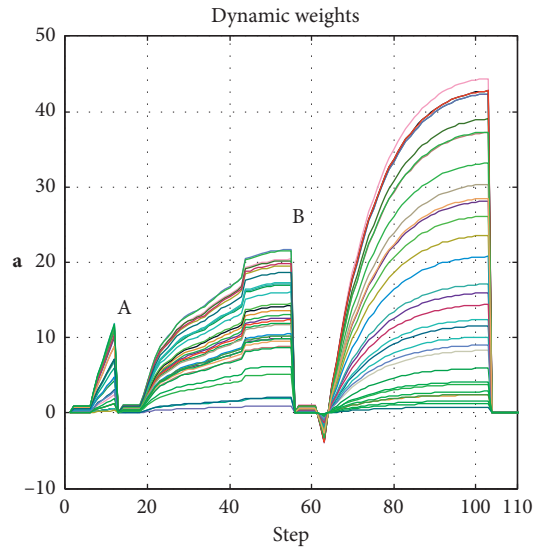


Figure 31: Changes of RFNN weights during process.

limited in a small area. Contrasted to path planning in autonomous driving area, it is a kind of a local planning method. Furthermore, the characteristics of planning in free-space make it a suitable supplement to other global methods. It can also be used as an obstacle avoidance method. Although the method is introduced by considering the unmanned ground mobile robot in 2D condition, it can be popularized to unmanned aircraft and underwater unmanned vehicle in 3D condition. Because our method does not need the previous map and offline computation, it takes up less memory space in contrast to other methods. Thus, it may have good performance in 3D condition where surroundings are complex. In order to improve runtime performance, initial weights of RFNN are generated randomly, which can be modified in future work. Some heuristic methods can be designed to choose initial weights of RFNN. The structure of RFNN used in this paper is simple. The

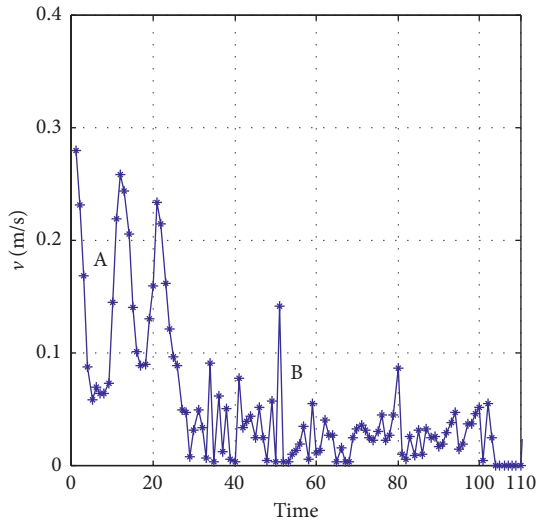


Figure 32: Linear velocity of the mobile robot.

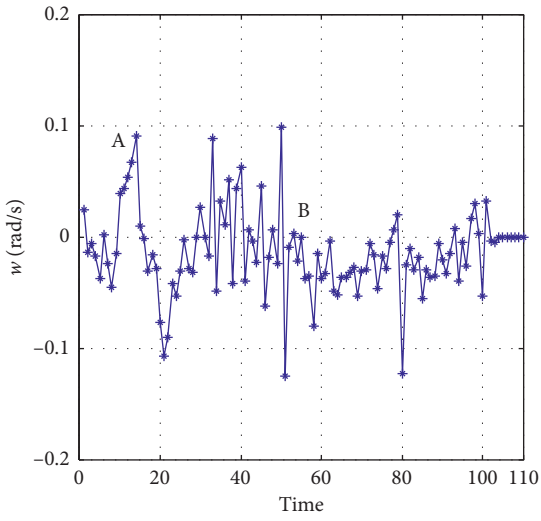


Figure 33: Angular velocity of the mobile robot.

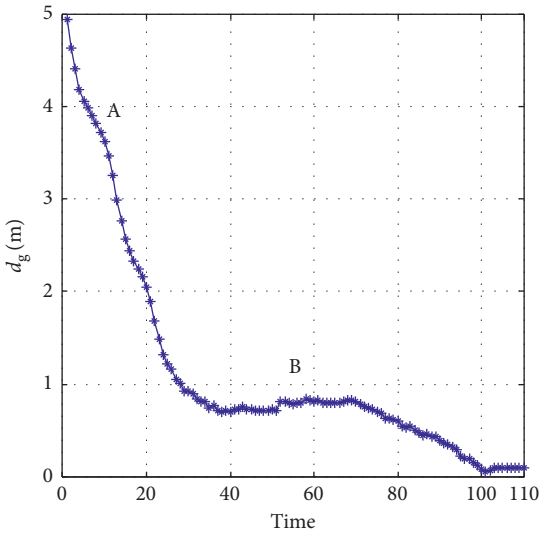


Figure 34: Position error of the mobile robot.

output of RFNN is not considered as the input of RFNN. More complex recurrent structures can be designed to improve effectiveness of the RFNN planning strategy in the future. Furthermore, the location method of the autonomous mobile robot used in this paper is too simple. It also needs to be improved in future. Simultaneous localization and mapping (SLAM) and visual-inertial odometry techniques will be designed in our system. The perception system will be developed more effectively in the future.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61673129 and U1530119, the Natural Science Foundation of Heilongjiang Province of China under Grant F201414, and the Fundamental Research Funds for the Central Universities under Grant HEUCF160418.

References

[1] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: a nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2017.

[2] J. Faigl, "An application of self-organizing map for multirobot multigoal path planning with minmax objective," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 2720630, 15 pages, 2016.

[3] J. Ni, L. Wu, P. Shi, and S. X. Yang, "A dynamic bioinspired neural network based real-time path planning method for autonomous underwater vehicles," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 9269742, 16 pages, 2017.

[4] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nuchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 540–555, 2014.

[5] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[6] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: a survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[7] O. Montiel, R. Sepúlveda, and U. Orozco-Rosas, "Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field," *Journal of Intelligent & Robotic Systems*, vol. 79, no. 2, pp. 237–257, 2014.

[8] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using Bacterial Potential Field for

avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177–5191, 2015.

[9] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[10] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking with geometric continuous-curvature path planning," in *Proceedings of 2014 IEEE Intelligent Vehicles Symposium*, pp. 465–471, Ypsilanti, MI, USA, June 2014.

[11] A. H. Karami and M. Hasanzadeh, "An adaptive genetic algorithm for robot motion planning in 2D complex environments," *Computers & Electrical Engineering*, vol. 43, pp. 317–329, 2015.

[12] F. Abdessemed, M. Faisal, M. Emmadeddine et al., "A hierarchical fuzzy control design for indoor mobile robot," *International Journal of Advanced Robotic Systems*, vol. 11, no. 3, p. 33, 2014.

[13] M. Wang and J. N. K. Liu, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 625–643, 2008.

[14] M. K. Singh and D. R. Parhi, "Path optimisation of a mobile robot using an artificial neural network controller," *International Journal of Systems Science*, vol. 42, no. 1, pp. 107–120, 2011.

[15] T. Dierks, B. Brenner, and S. Jagannathan, "Neural network-based optimal control of mobile robot formations with reduced information exchange," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1407–1415, 2013.

[16] Z. Peng, G. Wen, S. Yang, and A. Rahmani, "Distributed consensus-based formation control for nonholonomic wheeled mobile robots using adaptive neural network," *Nonlinear Dynamics*, vol. 86, no. 1, pp. 605–622, 2016.

[17] C. F. Juang, R. B. Huang, and Y. Y. Lin, "A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 5, pp. 1092–1105, 2009.

[18] Y. Y. Lin, J. Y. Chang, and C. T. Lin, "Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 310–321, 2013.

[19] C.-J. Kim and D. Chwa, "Obstacle avoidance method for wheeled mobile robots using interval type-2 fuzzy neural network," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 3, pp. 677–687, 2015.

[20] K. G. Jolly, R. Sreerama Kumar, and R. Vijayakumar, "Intelligent task planning and action selection of a mobile robot in a multi-agent system through a fuzzy neural network approach," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 6, pp. 923–933, 2010.

[21] M. A. Khanesar, E. Kayacan, M. Teshnehlab, and O. Kaynak, "Extended Kalman filter based learning algorithm for type-2 fuzzy logic systems and its experimental evaluation," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4443–4455, 2012.

[22] J. J. Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm," *Neurocomputing*, vol. 70, no. 13–15, pp. 2460–2466, 2007.

[23] X. Wang and Y. Huang, "Convergence study in extended kalman filter-based training of recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 22, no. 4, pp. 588–600, 2011.

[24] S. Mitsch, K. Ghorbal, D. Vogelbacher, and A. Platzer, "Formal verification of obstacle avoidance and navigation of ground robots," *International Journal of Robotics Research*, vol. 36, no. 12, pp. 1312–1340, 2017.

[25] Y. Han, Q. Zhu, and Y. Xiao, "Data-driven control of autonomous vehicle using recurrent fuzzy neural network combined with PID method," in *Proceedings of 37th Chinese Control Conference (CCC 2018)*, pp. 5239–5244, Wuhan, China, July 2018.

[26] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 279–297, 1994.

[27] F. B. Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: a survey," *Journal of Intelligent & Robotic Systems*, vol. 53, no. 3, pp. 263–296, 2008.

[28] I. Cinaroglu and Y. Bastanlar, "A direct approach for object detection with catadioptric omnidirectional cameras," *Signal, Image and Video Processing*, vol. 10, no. 2, pp. 413–420, 2015.

[29] Q. Zhu, C. Liu, and C. Cai, "A novel robot visual homing method based on SIFT features," *Sensors*, vol. 15, no. 10, pp. 26063–26084, 2015.

[30] C. Cai, B. Fan, X. Weng, Q. Zhu, and L. Su, "A target tracking and location robot system based on omnistereo vision," *Industrial Robot: An International Journal*, vol. 44, no. 6, pp. 741–753, 2017.

[31] Q. Zhu, H. Xie, C. Cai, and P. Liu, "A rapid and precise self-localization approach of mobile robot based on binocular omni-directional vision," in *Proceedings of 36th Chinese Control Conference (CCC 2017)*, pp. 26–28, Dalian, China, July 2017.

[32] Q. Zhu, P. Liu, and C. Cai, "Robust method of indoor robot localization based on artificial landmark," *Journal of Computer Applications*, vol. 37, pp. 126–130, 2017.

[33] Q. Zhu, X. Liu, and C. Cai, "Feature optimization for long-range visual homing in changing environments," *Sensors*, vol. 14, no. 2, pp. 3342–3361, 2014.

Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 1983897, 10 pages
https://doi.org/10.1155/2018/1983897

Research Article

Fuzzy Evaluation of Pharmacokinetic Models

Carlos Sepúlveda,[1] **Oscar Montiel**,[1] **José M. Cornejo Bravo**,[2] **and Roberto Sepúlveda**[1]

[1]*Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI-IPN), Av. Instituto Politécnico Nacional, No. 1310, Col. Nueva Tijuana, 22435 Tijuana, BC, Mexico*
[2]*Facultad de Ciencias Químicas e Ingeniería, Universidad Autónoma de Baja California, Calzada Universidad 14418, Parque Industrial Internacional Tijuana, 22390 Tijuana, BC, Mexico*

Correspondence should be addressed to Oscar Montiel; oross@ipn.mx

Received 4 July 2018; Revised 8 September 2018; Accepted 19 September 2018; Published 1 November 2018

Guest Editor: Fevrier Valdez

Copyright © 2018 Carlos Sepúlveda et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Population pharmacokinetic (PopPK) models allow researchers to predict and analyze drug behavior in a population of individuals and to quantify the different sources of variability among these individuals. In the development of PopPK models, the most frequently used method is the nonlinear mixed effect model (NLME). However, once the PopPK model has been developed, it is necessary to determine if the selected model is the best one of the developed models during the population pharmacokinetic study, and this sometimes becomes a multiple criteria decision making (MCDM) problem, and frequently, researchers use statistical evaluation criteria to choose the final PopPK model. The used evaluation criteria mentioned above entail big problems since the selection of the best model becomes susceptible to the human error mainly by misinterpretation of the results. To solve the previous problems, we introduce the development of a software robot that can automate the task of selecting the best PopPK model considering the knowledge of human expertise. The software robot is a fuzzy expert system that provides a method to systematically perform evaluations on a set of candidate PopPK models of commonly used statistical criteria. The presented results strengthen our hypothesis that the software robot can be successfully used to evaluate PopPK models ensuring the selection of the best PopPK model.

1. Introduction

Pharmacokinetics (PK) is a subdivision of pharmacology in which mathematical models are developed to study the processes of absorption, distribution, and elimination of a drug once a given dose has been administered to a given individual [1]. These mathematical models are derived from representing the body as a system of compartments [2] (Figure 1), where the transfer rates of absorption, distribution, redistribution, and elimination between these compartments can be used to determine the parameters of the PK model such as clearance (Cl) and volume of distribution (V) that allow us to predict the drug concentration in plasma (C_p) in a given time [3].

Let us consider the one compartment PK model of Figure 1, where the rate of elimination of the drug presented in the body decreases in proportion to the C_p, that is, with a first-order elimination process k_e and the drug administered as a single intravenous bolus dose (D); the kinetic of C_p in the compartment—the body—at time $t > 0$ is depicted by the following deterministic differential equation:

$$\frac{dC_{p_t}}{dt} = -k_e C_{p_t},$$
$$C_{p_0} = \frac{D}{V}. \tag{1}$$

In this case, the term k_e is given by

$$k_e = \frac{\mathrm{Cl}}{V}, \tag{2}$$

where V is the distribution volume of the drug within the body and Cl measures the ability of the liver and the kidney, mainly to extract a drug from the body. In this context, Cl and V are fixed parameters that describe the drug concentration of a given dose D over time t. Equation (1) has the following explicit solution:

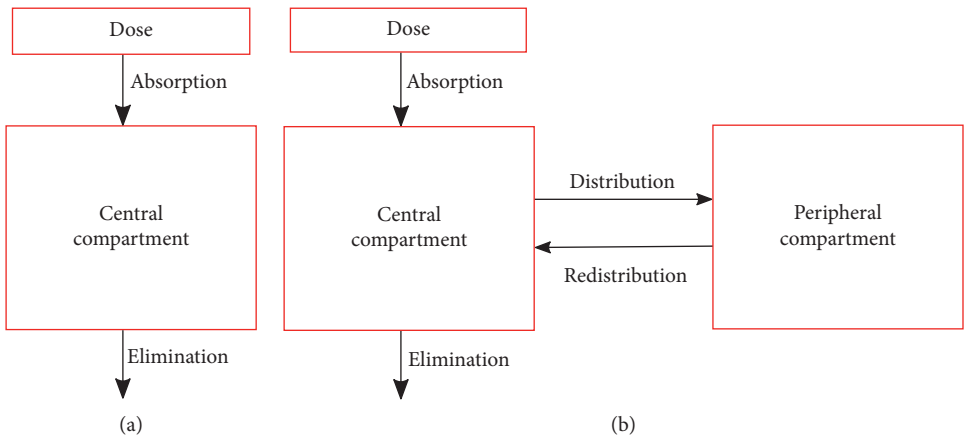


Figure 1: Example of the body viewed as (a) one-compartment model or (b) two-compartment model. The central compartment represents the plasma and tissues, and the peripheral compartment represents those tissues that take up the drug at a slower rate.

$$C_{p_t} = \frac{D}{V} e^{-(\mathrm{Cl}/V)\cdot t}. \tag{3}$$

The model (3) states a relationship between the independent variable t and the dependent variable C_p [1, 4].

PopPK models are designed to analyze drug behavior in a group of individuals; hence, it is possible to generalize the model for similar individuals that were not subject of the study. NLME modeling is the main current approach for PopPK model development; it allows estimation of parameters in the presence of different levels of variability, and it is commonly used when it is not possible to obtain the complete information on repeated measurements of individuals [2]. Most of the time, the process to find the final PopPK model that better represents the data behavior requires the development of several PopPK models. It is common to use interchangeable software programs like R or SAS to apply various evaluation criteria to select which PopPK model is "better" among all the possible PopPK models [5–7]. These evaluation criteria range from graphical analysis to statistical methods [8]. The graphical analysis is limited to expert modelers, and besides that, it can be time-consuming and is subject to human error or misinterpretation.

The statistical methods such as the estimated error variance (MSE), the objective function value (OFV), the Akaike's information criterion (AIC), or the Bayesian information criterion (BIC), also known as Schwarz criterion (CS) offer quantitative terms (values) which are much easier to interpret and for consequence more easily to exclude or accept. However, the respective values may be contrasted between PopPK models when more than one statistical criterion is used complicating in this way ensuring the selection of the best PopPK model, thus becoming a multiple criteria decision making (MCDM) problem [9].

Fuzzy expert systems is an area where artificial intelligence and fuzzy logic meet to make robots backing or even replace human experts that want to do automated tasks [10]. The fuzzy logic has been recognized as an essential problem-solving technique when human evaluations are required for interpretation of medical findings [11], as well as in problems where is necessary for managing data with uncertainty such as [12], where a type-2 fuzzy logic methodology is used to help a neural network to handle complex time series data.

We contribute with a new software robot to facilitate the selection of the best pharmacokinetic model by an automated process when the comparison of the implemented evaluation criteria values among the models cannot be categorically determined, addressing the problem as a decision problem. We present results using a software robot to select the best PopPK model of tobramycin by an automated process.

The organization of this work is as follows: in Section 2, an overview of nonlinear mixed-effects modeling is provided. In Section 3, the implementation of the software robot is explained. In Section 4, the parameter estimation results and the outcomes of the software robot are presented. In Section 5, an analysis of the results is exposed. Finally, in Section 6, the conclusions are given.

2. Nonlinear Mixed-Effects Model Framework

In mixed-effects models, the data set is longitudinal, i.e., it is composed of repeated samples of the same individuals [13, 14]. This is the case of the PopPK experimental data; it consists of a vector y_{ij} of samples of the $j-$th measurement (where, $j = 1, \ldots, n$) of C_p in the $i-$th individual (where, $i = 1, \ldots, n$). Therefore, we can obtain the sample y_{ij} for the individual i at the time t_{ij} by using the model:

$$y_{ij} = f\left(x_{ij}, \phi_i\right) + \varepsilon_{ij}; \quad \varepsilon_{ij} \sim \left(0, \sigma^2\right), \tag{4}$$

where $f(\cdot)$ is a nonlinear function as Equation (3) relating a vector of known values x_{ij} (e.g., dose and time) to the unknown parameter vector ϕ_i (e.g., $(\mathrm{Cl}_i, V_i)^{\mathrm{T}}$) and the variable ε_{ij} in the second term is the residual error that is assumed to be normally distributed with mean zero and variance σ^2. It is sensible to expect that each individual has a different parameter vector ϕ_i, and this is described in a second stage by the covariate model:

$$\phi_i = g(z_i; \theta) + \eta_i; \quad \eta_i \sim (0, \omega^2). \tag{5}$$

Equation (5) describes the variation among different individuals accounted through the individual-specific parameters ϕ_i.

The function $g(\cdot)$ relates the specific covariates z_i of the individuals to the population parameters θ. η_i is a $(q \times 1)$ vector containing the random effects that have a normal distribution with a covariance matrix ω^2; other distributions for random effects exist [14].

2.1. Estimation of Population Parameters. Since the PopPK data are longitudinal, we can assume that the individuals are sampled randomly from the population, and therefore it is practical to assume that the η_i's are also randomly sampled (even though they are not observable) to conduct the estimations of the parameters of (4) and (5) by optimizing the likelihood function defined by the following equation:

$$L(\theta, \Omega, \Sigma \mid \mathbf{y}) \triangleq p(\mathbf{y} \mid \beta, \Omega, \Sigma), \tag{6}$$

where $L(\cdot)$ is the likelihood function, Ω is the variance-covariance matrix of all the η_i's, and Σ the residual variance-covariance matrix of all the ε_i's where $p(\mathbf{y} \mid \beta, \Sigma, \eta)$ is the conditional probability density of all measurements [15]. If $p(\mathbf{y} \mid \beta, \Sigma, \eta)$ is explicit, the likelihood function $L(\cdot)$ is explicit too, and the exact maximum likelihood estimators can be applied [16], otherwise the estimation of model parameters is made using linear approximations for (6) [17]. Either way, using an explicit likelihood function or an approximation, the precision in the estimated vector $\widehat{\theta}$ depends not only in the residual variability but also in the estimated variance-covariance matrix $\widehat{\Omega}$, which is calculated using the inverse Hessian matrix of the $-\log L(\cdot)$.

2.2. Nonlinear Mixed-Effects Model Evaluation Criteria. In the literature referring on the analysis and development of NLME models and PopPK models, the researchers frequently highlighted three statistical criteria to carry out the different evaluations of the model and thus decide which PopPK model they should select within a collection of PopPK models they are considering.

The first two evaluation criteria that we are going to present in this work are the Akaike information criterion (ACI) [18] and the Schwarz's information criterion (SC) [19]. These evaluation criteria are derived from information theory and a Bayesian approach and are commonly used for evaluating nonlinear mixed-effects models when the parameters of the model were estimated by the maximum likelihood method. The AIC criterion provides a balance between the goodness of the fit of the model and the number of parameters required to obtain the fit [20] as is shown in the following equation:

$$\text{AIC} = N \cdot \log(\text{OFV}) + 2 \cdot \text{NP}, \tag{7}$$

where N is the number of observations and NP the number of parameters. The final and optimal model is the one with the smallest AIC value. Unlike the AIC criterion, the *SC* criterion accepts equal probability for each model and for every possible parameter value under the model as is shown in the following equation:

$$SC = N \cdot \log(\text{OFV}) + \text{NP} \cdot \log \text{NP}. \tag{8}$$

The third evaluation criterion is the estimated error variance also known as the mean square error (MSE). This is based on the simple variance estimator shown in the following equation:

$$\widehat{\sigma}_i^2 = \frac{\sum_{i=1}^{m} y_i - \widehat{y}}{n}. \tag{9}$$

The use of the MSE is due to the residuals, and it also contains essential information on the quality of the model. Specifically, the estimated variance in the responses, where y is a vector of observed values and $\widehat{y}_i$, represents a vector of n predictions.

The version of Equation (9) depends on the estimation method used to estimate the PopPK parameters, for example, the variance estimator when the restricted maximum estimation likelihood (REML) is used is shown in the following equation:

$$\widehat{\sigma}_i^2 = \frac{\sum_{i=1}^{m} y_i - \widehat{y}_{\text{REML}_i}}{n - p}. \tag{10}$$

Equation (10) takes into account the degrees of freedom by subtracting the total of parameters p used in the estimation process [14] where $\widehat{y}_{\text{REML}_i}$ is a vector of predicted values applying REML.

3. Implementation of the Software Robot

3.1. The Software Robot. The software robot contained a fuzzy expert system that takes the advantage of the human knowledge and their expertise in a given field to create linguistic descriptors for variables and create fuzzy sets that allow to control the behavior of phenomena under study or even as a tool for decision making [21]. In this work, we developed a software robot as shown in Figure 2 to evaluate indistinct versions of a PopPK model for an aminoglycoside antibiotic named tobramycin which has been the subject of many studies due to its narrow therapeutic window [22]. In this figure, first, we defined the number N of PopPK models to be developed taking into account the number of parameters and covariates. After that, we proceed to establish one by one the N PopPK models. Three statistical evaluation criteria are performed on each model, and they are AIC, SC, and MSE. The software robot uses them as the inputs for the fuzzy expert system, and it collects each defuzzified output into an array until all the N models have been evaluated. Then, the software robot chooses the best PopPK model which is the one that had the lowest value in the fuzzy evaluation, and this will be selected as the best PopPK model.

Thus, the evaluation made by our software robot will guide us to decide if a PopPK model of tobramycin is the best within a group of PopPK models of Tobramycin.

The variables are the evaluation criteria AIC, SC, and MSE presented in Equations (7), (8), and (10), respectively,

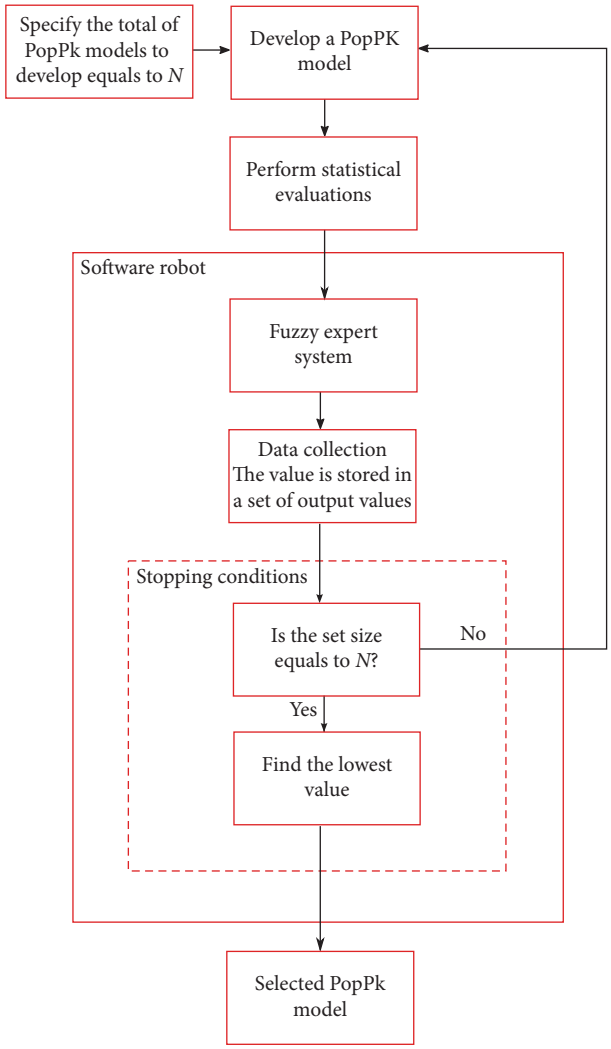


Figure 2: Block diagram.

and commonly used to evaluate a PopPK model, and their linguistic descriptors are the ranges of values. For example, the variable AIC has a range of values from 670 to 672.5 considered "Very Low", from 670 to 675 considered "Low," from 672.5 to 677.5 considered "Medium," from 675 to 680 considered "High," and from 677.5 to 680 considered "Very High." "Very Low," "Low," "Medium," "High," and "Very High" are the linguistic descriptors for the sets of values for the variable AIC.

As is typical of the fuzzy theory, the sets of values overlap, and in this way, a value may partially belong to a set and have a degree of membership $0 \leq \mu \leq 1$ that is any place between zero and one, where μ represents the degree of membership. Thus, the value belongs to several sets with the total membership adding to one. Going back to the above example, the linguistic descriptors of "Very Low" and "Low" are two fuzzy sets for the variable AIC that may overlap, so an AIC of 670.5 could be mostly "Very Low" with $\mu = 0.8$ and somewhat "Low" with $\mu = 0.2$.

Implementing fuzzy sets with linguistic descriptors to perform a generalization of the output evaluation criteria, results of the fitted PopPK model, we may establish relationships between variables to evaluate the PopPK model by extracting fuzzy rules in terms used by a human expert of the form $IF-THEN$. In other words, a fuzzy system can perform a description of the phenomena under study, based on the antecedents and consequents presented in a fuzzy rule, that is in the form IF **condition** $THEN$ **action** *rules* [23].

Using the three evaluation criteria AIC, SC, and MSE, we can set a knowledge base in such way that IF AIC is mostly "Very Low" AND CS is mostly "Very Low" AND MSE is also mostly "Very Low" $THEN$ Evaluation is "OPTIMAL". That is, under the fuzzy logic, we are not only going to have linguistic descriptors as the inputs of the fuzzy system but also output variables.

In summary, what a fuzzy system does is to transform crisp input values into fuzzy inputs through a fuzzification unit that establishes the degree of membership to the fuzzy sets for previously defined variables. Then, the fuzzy system uses a rulebase designed by the human expert to predict the fuzzy output of the phenomenon under study, whereby the fuzzy system has a defuzzification unit that transforms the fuzzy output into a crisp value. For this work, we created a fuzzy system with three crisp input variables as shown in Figure 3, the Mamdani max-min inferences to obtain the membership functions of the system and the centroid method to defuzzify the output.

3.2. Fuzzification. As we mentioned, the values of the three evaluation criteria for a developed PopPK model are used to perform a comparison between models, so we can finally decide the PopPK model that is "better," which leads the researchers to deal with an MCDM problem. As an example, a researcher can get to the point in which he needs to decide between two models that have the best evaluation criteria values (Table 1).

In the example of Table 1, the model 2 has a better result in the AIC than model 1 for 1.866 units and in the MSE for 0.048 units. However, model 1 performs better in CS for a significant difference of almost 3 units (2.794). This type of ambiguity in the statistical results and the pressure to get the optimal PopPK model can result in a misinterpretation of the evaluation criteria to select the optimal PopPK model.

In this work, the values of the linguistic variables AIC, CS, and MSE calculated from the development of indistinct versions of a PopPK model of tobramycin were used as our fuzzy input sets. Their maximum and their minimum values are shown in Table 2.

The membership function for the three input variables is tagged as "Very Low," "Low," "Normal," "Medium," "High," and "Very High," and their corresponding parameter values are shown in Table 3.

The membership functions for the output variable "Evaluation" are represented by five membership functions, and the corresponding types and parameter values are shown in Table 4.

3.3. Fuzzy Rulebase. We created the fuzzy rulebase using the human knowledge of an expert who judged "how much

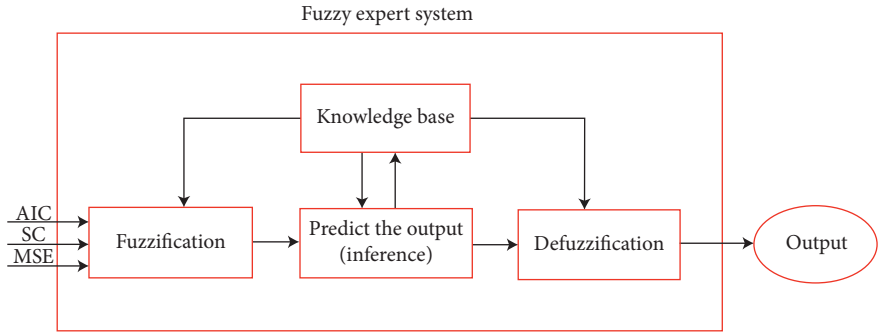


Figure 3: The crisp input values AIC, SC, and MSE are transformed into fuzzy inputs by the fuzzification unit which assigns the degree of membership to the fuzzy sets defined for the variables. The fuzzy system can then make inferences or predict outputs of the system by applying knowledge from the expert. Finally, the defuzzification unit transforms the obtained fuzzy output into a crisp value.

Table 1: Example of the final evaluation results for a PopPK model.

PopPK model	AIC	CS	MSE
1	652.470	665.300	0.14
2	650.604	668.094	0.092

The best results are the values with red color.

Table 2: AIC, CS, and MSE values used for the fuzzy expert system.

Variable	Minimum value	Maximum value
AIC	670	680
CS	680	795
MSE	0.095	0.14

Table 3: Parameters of the membership function values for the input variables.

Linguistic variables	Linguistic term names, type shapes, and parameters				
	Very Low trapezoidal	Low triangular	Medium triangular	High triangular	Very High trapezoidal
AIC	[670 670 670.3 672.5]	[670 672.5 675]	[672.5 675 677.5]	[675 677.5 680]	[677.5 679.7 680 680]
CS	[676.6 679.6 680.4 683.4]	[680 683.8 687.5]	[683.8 687.5 691.3]	[687.5 691.3 695]	[691 695 695 698]
MSE	[0.092 0.092 0.09397 0.104]	[0.0921 0.104 0.116]	[0.104 0.116 0.128]	[0.116 0.128 0.14]	[0.128 0.1384 0.14 0.14]

Table 4: Parameters of the membership function values for the output variable.

Linguistic variables	Lingustic term names, type shapes, and parameters				
	Optimal trapezoidal	High Acceptable	Acceptable triangular	Low Acceptable triangular	Rejected trapezoidal
Evaluation	[0 0 10 30]	[10 30 50]	[30 50 70]	[50 70 90]	[70 90 100 100]

more significant one attribute is than the other." The inference is made on the rules $IF-THEN$; one rule for each fuzzy output set. The total of rules stems from the heuristic $R = l^n$, where R is equal to the total of rules, l is equal to the number of linguistic descriptors, and n represents the number of input variables; thus in our case with $l = 5$ and $n = 3$ results in a total of 125 rules (Table 5).

3.3.1. Defuzzifier. The fuzzified functions obtained from fuzzy inference are converted into numeric values, for example:

IF the **AIC** is Low (677)
AND **CS** is Low (686)
AND **MSE** is High (.12)
THEN the PopPK model is Low Acceptable **(75.2)**

The above is achieved by applying the method of center of gravity defuzzifier for N rules using the formula:

$$C = \frac{\sum_{i=1}^{N} b_i \int \mu(i)}{\sum_{i=1}^{N} \int \mu(i)}, \tag{11}$$

where C represents the crisp output value, b_i represents the center of the membership function of the consequent of the rule i, and $\int \mu(i)$ is the area under the membership function $\int \mu(i)$ of the consequent in the rule i. In other words, the center of gravity method calculates the center of mass from the output membership functions.

Figure 4 shows the global behavior of the fuzzy system for the relation between the input variables CS and AIC,

Table 5: Fuzzy rules.

Rule number	Linguistic inputs			Linguistic output
	AIC	CS	MSE	Evaluation
1	Very Low	Very Low	Very Low	Optimal
2	Very Low	Very Low	Low	High Acceptable
3	Very Low	Very Low	Medium	Acceptable
4	Very Low	Very Low	High	Low Acceptable
5	Very Low	Very Low	Very	Optimal
7	Very Low	Low	Low	High Acceptable
8	Very Low	Low	Medium	Acceptable
9	Very Low	Low	High	Low Acceptable
10	Very Low	Low	Very High	Rejected
11	Very Low	Medium	Very Low	High Acceptable
12	Very Low	Medium	Low	High Acceptable
13	Very Low	Medium	Medium	Low Acceptable
14	Very Low	Medium	High	Low Acceptable
15	Very Low	Medium	Very High	Rejected
16	Very Low	High	Very Low	Acceptable
17	Very Low	High	Low	Acceptable
18	Very Low	High	Medium	Low Acceptable
19	Very Low	High	High	Rejected
20	Very Low	High	Very High	Rejected
21	Very Low	Very High	Very Low	Low Acceptable
22	Very Low	Very High	Low	Low Acceptable
23	Very Low	Very High	Medium	Rejected
24	Very Low	Very High	High	Rejected
25	Very Low	Very High	Very High	Rejected
26	Low	Very Low	Very Low	Optimal
27	Low	Very Low	Low	Optimal
28	Low	Very Low	Medium	High Acceptable
29	Low	Very Low	High	Low Acceptable
30	Low	Very Low	Very High	Rejected
31	Low	Low	Very Low	Optimal
32	Low	Low	Low	High Acceptable
33	Low	Low	Medium	Acceptable
34	Low	Low	High	Low Acceptable
35	Low	Low	Very High	Rejected
36	Low	Medium	Very Low	Optimal
37	Low	Medium	Low	High Acceptable
38	Low	Medium	Medium	Acceptable
39	Low	Medium	High	Low Acceptable
40	Low	Medium	Very High	Rejected
41	Low	High	Very Low	High Acceptable
42	Low	High	Low	Acceptable
43	Low	High	Medium	Low Acceptable
44	Low	High	High	Rejected
45	Low	High	Very High	Rejected
46	Low	Very High	Very Low	Acceptable
47	Low	Very High	Low	Low Acceptable
48	Low	Very High	Medium	Rejected
49	Low	Very High	High	Rejected
50	Low	Very High	Very High	Rejected
51	Medium	Very Low	Very Low	Optimal
52	Medium	Very Low	Low	High Acceptable
53	Medium	Very Low	Medium	Acceptable
54	Medium	Very Low	High	Low Acceptable
55	Medium	Very Low	Very High	Rejected
56	Medium	Low	Very Low	High Acceptable
57	Medium	Low	Low	High Acceptable
58	Medium	Low	Medium	Acceptable
59	Medium	Low	High	Rejected
60	Medium	Low	Very High	Rejected

Table 5: Continued.

Rule number	Linguistic inputs		Linguistic output	
	AIC	CS	MSE	Evaluation
61	Medium	Medium	Very Low	Acceptable
62	Medium	Medium	Low	Acceptable
63	Medium	Medium	Medium	Low Acceptable
64	Medium	Medium	High	Rejected
65	Medium	Medium	Very High	Rejected
66	Medium	High	Very Low	Acceptable
67	Medium	High	Low	Low Acceptable
68	Medium	High	Medium	Rejected
69	Medium	High	High	Rejected
70	Medium	High	Very High	Rejected
71	Medium	Very High	Very Low	Low Acceptable
72	Medium	Very High	Low	Low Acceptable
73	Medium	Very High	Medium	Rejected
74	Medium	Very High	High	Rejected
75	Medium	Very High	Very High	Rejected
76	High	Very Low	Very Low	Acceptable
77	High	Very Low	Low	Acceptable
78	High	Very Low	Medium	Low Acceptable
79	High	Very Low	High	Rejected
80	High	Very High	Very High	Rejected
81	High	Low	Very Low	Low Acceptable
82	High	Low	Low	Low Acceptable
83	High	Low	Medium	Rejected
84	High	Low	High	Rejected
85	High	Low	Very High	Rejected
86	High	Medium	Very Low	Low Acceptable
87	High	Medium	Low	Low Acceptable
88	High	Medium	Medium	Rejected
89	High	Medium	High	Rejected
90	High	Medium	Very High	Rejected
91	High	High	Very Low	Low Acceptable
92	High	High	Low	Rejected
93	High	High	Medium	Rejected
94	High	High	High	Rejected
95	High	High	Very High	Rejected
96	High	Very High	Very Low	Low Acceptable
97	High	Very High	Low	Low Acceptable
98	High	Very High	Medium	Rejected
99	High	Very High	High	Rejected
100	High	Very High	Very High	Rejected
101	Very High	Very Low	Very Low	Acceptable
102	Very High	Very Low	Low	Low Acceptable
103	Very High	Very Low	Medium	Rejected
104	Very High	Very Low	High	Rejected
105	Very High	Very Low	Very High	Rejected
106	Very High	Low	Very Low	Low Acceptable
107	Very High	Low	Low	Low Acceptable
108	Very High	Low	Medium	Rejected
109	Very High	Low	High	Rejected
110	Very High	Low	Very High	Rejected
111	Very High	Medium	Very Low	Low Acceptable
112	Very High	Medium	Low	Low Acceptable
113	Very High	Medium	Medium	Rejected
114	Very High	Medium	High	Rejected
115	Very High	Medium	Very High	Rejected
116	Very High	High	Very Low	Rejected
117	Very High	High	Low	Rejected
118	Very High	High	Medium	Rejected
119	Very High	High	High	Rejected

Table 5: Continued.

Rule number	Linguistic inputs		Linguistic output	
	AIC	CS	MSE	Evaluation
120	Very High	High	Very High	Rejected
121	Very High	Very High	Very Low	Rejected
122	Very High	Very High	Low	Rejected
123	Very High	Very High	Medium	Rejected
124	Very High	Very High	High	Rejected
125	Very High	Very High	Very High	Rejected

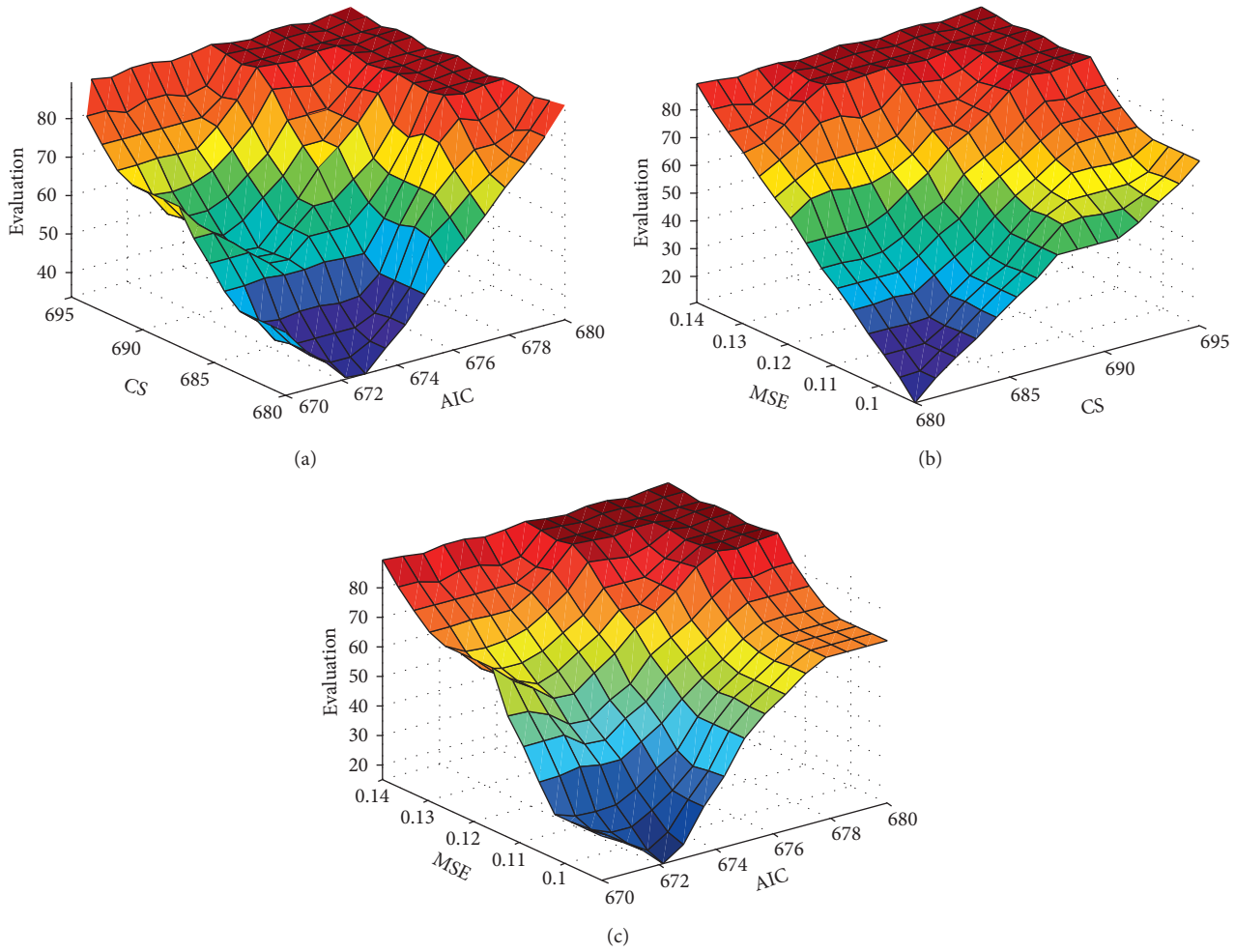


Figure 4: The figure shows the visualization of the system surface when the inputs AIC, CS, and MSE are interacting with each other. (a) The inputs AIC and CS are interacting. (b) The inputs MSE and CS are interacting. (c) The inputs MSE and AIC are interacting.

MSE and CS, and MSE and AIC considering all their rules. For example, if the value of CS is "Very Low," that is 681 and AIC is "High," that is 678.5, then the PopPK model obtains a bad evaluation, or if the value of MSE is "Low," that is 0.1 and the value of CS is also "Low", that is 684, the model obtains an excellent evaluation. Another example will be a scenario where the value of MSE is "Medium," that is 0.12, and the value of AIC is "Low," that is 674.5, and the model obtains a good evaluation.

If it was required, the search for the lowest value could be done according to Algorithm 1. The crisp output value can be stored in an array together with other output values taken from other PopPK models that were evaluated previously and then compare them in an automated way to find the lowest value which will determine the optimal PopPK model.

4. Experimental Results for the Evaluation of Several PopPK Models of Tobramycin

This section describes the experiments performed to evaluate and compare 21 PopPK models based on the tobramycin database. We developed the PopPK Model of tobramycin in Matlab applying the single compartment model, as it is

```
(1): i ← 0
(2): low ← i
(3): for j = i + 1 to total of candidate models–1 do
(4):     if array[j] is less than array[low] then
(5):         Set low to j
(6):     end if
(7): end for
```

Algorithm 1: Find the lowest value algorithm.

shown in Figure 1(a). The covariate model for both V and Cl were defined as linear.

The estimation of the parameters for all the generated PopPK models was made by applying REML. The optimization process was conducted with the quasi-Newton algorithm using the initial set values for fixed effects of 0.01 for Cl and 0.01 for V and with a maximum of 100 iterations. The same residual error model was applied to all the experiments. The only variations in the developed PopPK models are the covariate type and the number of parameters included in each model.

4.1. Experiment 1. After having evaluated 21 PopPK models using the evaluations criteria (7), (8), and (10), we end up with the 8 best PopPK models shown in Table 6.

Once we performed a simple analysis of the 8 PopPK models, we can easily decide that the PopPK models 1 and 2 have the worst values. However, considering the rest of the six PopPK models, it is hard to determine which model is the best given that some of their evaluations values are countered.

4.2. Experiment 2. In this experiment, we used the results of the evaluation criteria shown in Table 6 of the PopPK models 2, 4, 5, 6, 7, and 8, as input variables in our software robot as described in Section 3. The results of the evaluations made for these PopPK models are shown in Table 7, where now the fuzzy system evaluation (FSE) criteria results are presented for the six selected PopPK models.

5. Analysis of Results

The results of experiment 1 of Table 6 represent an example of the type of problem the researcher can face when working in the development of any PopPK model. In this particular case, Table 6 shows that the selection of the best PopPK model is not a trivial decision due to the time consumed to determine which PopPK model obtained the most significant evaluations in comparative with the rest of the models. For example, in Table 6, it can be seen that PopPK model 5 has best AIC evaluation than the PopPK models 4, 6, and 7, but worst evaluation in CS as well as worst evaluation in MSE than the PopPK model 6. The results of experiment 2 of Table 7 show that the fuzzy system gives the worst FSE value to PopPK model 7 (88.9). The PopPK model 7 has the worst evaluation of AIC, CS, and MSE than the PopPK model 4, which is the model with the best FSE value (31.6). It can be seen that the fuzzy system considered the trade-offs among the rest of standard statistical evaluation criteria: AIC, CS, and MSE provided a clearer picture of the order of the worst PopPK model to the best PopPK model.

Table 6: Results of the 8 best PopPK model of tobramicyn. The covariates used are weight (WT), age, height (HT), and body mass index (BMI).

PopPK	# of estimated				
Model	Parameters	Covariates	AIC	CS	MSE
1	6	(AGE/V)/(WTCl)	682.04	698.2	0.0936
2	6	(SEX,AGE/V)	679.9	695	0.093
3	7	(WT,AGE/V)/(WT/Cl)	683.6	701.1	0.093
4	6	(SEX,AGE/V)	673.1	689.2	0.09336
5	9	(SEX,AGE/V)/(SEX,AGE,WT/Cl)	670.4	693	0.0936
6	7	(SEX, AGE/V) (SEX/Cl)	672.7	691	0.093
7	7	(WT, AGE, SEX/Cl)	675.3	691.3	0.139
8	10	(WT, HT/V)/(WT, AGE, HT, BMI/Cl)	682.7	707.5	0.0928

Table 7: Summary of all evaluation criteria applied. Model 4 is the best evaluated by the fuzzy system.

PopPK	# of estimated					
Model	Parameters	Covariates	AIC	CS	MSE	FSE
2	6	(WT,AGE/V)	679.9	695	0.093	88.4
4	6	(SEX,AGE/V)	673.1	689.2	0.09336	31.6
5	9	(SEX, AGE/V)/(SEX, AGE,WT/Cl)	670.4	693	0.0936	55.6
6	7	(SEX, AGE/V) (SEX/Cl)	672.7	691	0.093	34.4
7	7	(WT, AGE, SEX/Cl)	675.3	691.3	0.139	88.9
8	10	(WT, HT/V)/(WT, AGE, HT, BMI/Cl)	682.7	707.5	0.0928	88.7

6. Conclusions

Our results bolster the hypothesis that the software robot can be successfully implemented to evaluate PopPK models ensuring the selection of the best PopPK model when the choice of this becomes an MCDC problem. Given that the FSE criteria is built taking into account the classical evaluation criteria (AIC, CS, and MSE), we are able to only use the FSE as our unique automated evaluation criteria. This reduced the time and the error in the selection of the best PopPK model.

Another advantage is that if we want to use the same fuzzy system methodology for the evaluation of a different case study, for example, a PopPK model that involves another type of drug as phenobarbital, or the amount of individuals in the study is different, we only need to evaluate

a certain amount of models to perform adjustments in the membership function ranges.

Our software robot performs fuzzy evaluations offering a stronger alternative to increase the efficiency in the selection of the best PopPK from a set. It can help the pharmaceutical scientist or the expert researchers in the area of computer intelligence to incorporate, expand, and improve the implementation of this software either in new versions of commercial software or in the development of new software to incorporate human expertise.

The proposed software robot can become a strong support to further studies regarding this novel approach by helping the pharmaceutical scientist or the expert researchers in the area of computer intelligence to incorporate, expand, and improve this development.

Data Availability

The data of our document are available upon request to any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The research leading to these results has received funding from Instituto Politécnico Nacional (IPN), under the Research Project SIP20180499, as well as the Commission of Operation and Promotion of Academic Activities of IPN (COFAA), and the National Council of Science and Technology (CONACYT).

References

[1] S. Rosenbaum, *Basic Pharmacokinetics and Pharmacodynamics: An Integrated Textbook and Computer Simulations*, Wiley, Hoboken, NJ, USA, 2012.

[2] G. Fitzmaurice, M. Davidian, V. Geert, and G. Molenberghs, *Longitudinal Data Analysis*, Chapman and Hall/CRC., Boca Raton, FL, USA, 1st edition, 2008.

[3] J. Gabrielsoon and D. Weiner, *Pharmacokinetic and Pharmacodynamic Data Analysis, Concepts and Applications*, Swedish Pharmaceutical Press, Sweden, Europe, 4th edition, 2007.

[4] K. Vučićević, B. Miljkovic, S. Vezmar Kovačević, Z. Todorovic, M. Prostran, and I. Grabnar, *Population Pharmacokinetic Analysis of Therapeutic Drug monitoring Data Otimizing Pharmacotherapy of Antiepileptic Drugs*, Intech, London, UK, 1st edition, 2011.

[5] F. Mentré and S. Escolano, "Prediction discrepancies for the evaluation of nonlinear mixed-effects models," *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 33, no. 3, pp. 345–367, 2006.

[6] P. J. Williams and E. I. Ette, "Determination of model appropriateness," in *Simulation for Designing Clinical Trials: A Pharmacokinetic-Pharmacodynamic Modeling Prospective*, H. C. Kimko and S. B. Duffull, Eds., Marcel Dekker, New York, NY, USA, pp. 68–96, 2003.

[7] A. Farkas, G. Daroczi, P. Villasurda, M. Dolton, M. Nakagaki, and J. A. Roberts, "Comparative evaluation of the predictive performances of three different structural population pharmacokinetic models to predict future voriconazole concentrations," *Antimicrobial Agents and Chemotherapy*, vol. 60, no. 11, pp. 6806–6812, 2016.

[8] K. Brendel, E. Comets, C. Laffont, C. Laveille, and F. Mentré, "Metrics for external model evaluation with an application to the population pharmacokinetics of gliclazide," *Pharmaceutical Research*, vol. 23, no. 9, pp. 2036–2049, 2006.

[9] A. Lazim, "Fuzzy multi criteria decision making and its applications: a brief review of category," *Procedia-Social and Behavioral Sciences*, vol. 97, pp. 131–136, 2013.

[10] E. P. Klement and W. Slany, "Fuzzy logic in artificial intelligence," in *Proceedings of IJCAI'97 Workshop Nagoya*, pp. 179–190, Nagoya, Japan, August 1997.

[11] R. Kunhimangalam, S. Ovallath, and P. Joseph, "A novel fuzzy expert system for the identification of severity of carpal tunnel syndrome," *BioMed Research International*, vol. 2013, Article ID 846780, 12 pages, 2013.

[12] F. Gaxiola, P. Melin, F. Valdez, and O. Castillo, "Generalized type-2 fuzzy weight adjustment for backpropagation neural networks in time series prediction," *Information Sciences*, vol. 325, pp. 159–174, 2015.

[13] G. C. Pillai, F. Mentré, and J.-L. Steimer, "Non-linear mixed effects modeling – from methodology and software development to driving implementation in drug development science," *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 32, no. 2, pp. 161–183, 2005.

[14] M. Davidian and D. M. Giltinam, *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, Boca Raton, FL, USA, 1995.

[15] M. Lavielle, *Mixed Effects Models for the Population Approach Models, Tasks, Methods and Tools*, Chapman and Hall/CRC Press, Boca Raton, FL, USA, 2014.

[16] S. Donnet and A. Samson, "A review on estimation of stochastic differential equations for pharmacokinetic/pharmacodynamic models," *Advanced Drug Delivery Reviews*, vol. 65, no. 7, pp. 929–939, 2013.

[17] Y. Wang, "Derivation of various nonmem estimation methods," *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 34, no. 5, pp. 575–593, 2007.

[18] H. Akaike, "A bayesian analysis of the minimum aic procedure," *Annals of the Institute of Statistical Mathematics*, vol. 30, no. 1, pp. 9–14, 1978.

[19] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[20] R. Gieschke and D. Serafin, *Development of Innovative Drugs via Modeling with MATLAB*, Springer, Berlin, Heidelberg, Germany, 2014.

[21] Z. Gong, J. Forrest, and T. Yao, *Uncertain Fuzzy Preference Relations and Their Applications,* Vol 281, Springer-erlag, Berlin, Heidelberg, Germany, 2013.

[22] D. Xuan, J. F. Lu, D. P. Nicolau, and C. H. Nightingale, "Population pharmacokinetics of tobramycin in hospitalized patients receiving once-daily dosing regimen," *International Journal of Antimicrobial Agents*, vol. 15, no. 3, pp. 185–191, 2000.

[23] J. Espinosa, J. Vandewalle, and V. Wertz, *Fuzzy Logic, Identification and Predictive Control (Advances in Industrial Control)*, Springer-Verlag, Secaucus, NJ, USA, 1st edition, 2004.