A New Programming Approach based on Immune Plasma Algorithm for Symbolic Regression

Suat BAYIR[1], Hasan BADEM[2*†] and Selcuk ASLAN[3†]

[1*]Informatics, Kahramanmaras Sutcu Imam University, Kahramanmaras, 46100, Turkiye.
[2]Computer Engineering, Kahramanmaras Sutcu Imam University, Kahramanmaras, 46100, Turkiye.
[3]Department of Aeronautical Engineering, Erciyes University, Kayseri, 38210, Turkiye.

*Corresponding author(s). E-mail(s): hbadem@ksu.edu.tr;
Contributing authors: suatbayir1@gmail.com;
selcukaslan@erciyes.edu.tr;
†These authors contributed equally to this work.

Abstract

Finding an accurate mathematical model between the input and output values of a system is imperative for real-world problems. A mathematical model can be found by performing regression analysis with traditional statistical methods. However, modeling between input and output data is often not possible due to the limitations of traditional methods. This limitation can be overcome by using artificial intelligence-based symbolic regression methods. In the solution of symbolic regression problems, automatic programming methods such as genetic programming, ant programming, artificial immune system programming, and artificial bee colony programming are used. Automatic programming methods take as input the functions, terminals and algorithm-specific parameters that will be used in model creation. It aims to produce a solution for the problem studied with these values. In this study, automatic programming methods were developed, including population-based SIPP and GIPP, based on the immune plasma algorithm. The developed methods are tested on various symbolic regression test problems. In addition,

the test results were compared with the genetic programming and artificial bee colony programming methods, which are commonly applied in the literature. In addition, SIPP and GIPP methods are run on the Box-Jenkins data set, which is a real-world problem. Based on the findings obtained in the studies, it is predicted that the proposed SIPP and GIPP methods can be used in solving symbolic regression problems.

Keywords: Symbolic regression, automatic programming, immune plasma programming

1 Introduction

Regression methods are statistical approaches that aim to estimate the values of dependent variables over related independent variables [1]. Obtaining the mathematical expression by computing the relationship between the input values and the output values of the studied data set is defined as regression analysis [2]. The heuristic determination process of mentioned relationship by artificial intelligence methods is described as the symbolic regression problem [3]. To solve symbolic regression problems, automatic programming methods based on a heuristic approach are generally utilized. Automatic programming methods can be extracted from the mathematical model in real-world problems such as health [4], energy [5, 6], economics [7], engineering [8, 9], design [10, 11], and planning [12, 13]. The functions, terminals, and also specific parameters based on the algorithm are utilized as inputs for modeling in automatic programming methods. Automatic programming methods produce the candidate models for the handling problem to solve over the inputs. The accuracy of the models is evaluated with error functions based on the difference between the actual values and the predicted values by the models [14].

Genetic Programming (GP) [15, 16], Ant Colony Programming (AP) [17], Artificial Bee Colony Programming (ABCP) [18, 19], Artificial Immune System Programming (AISP) [20, 21] and Bio-geography Based Programming (BBP) [22, 23] are well-known the automatic programming techniques in the literature for symbolic regression problems.

The first automatic programming approach based on Artificial Intelligence (AI) was presented by J. R. Koza in 1992 [15]. In the this approach, genetic algorithm has been used as the prediction model. The developed method by Koza pioneers the use of automatic programming approaches in solving symbolic regression problems [24]. For the last 20 years, genetic programming-based methods have been used to solve symbolic regression problems [25–27]. The genetic algorithm based automatic modeling own a lot drawbacks including The low convergences, the large number of control parameters and the high computational cost of obtaining models with low error rates [28]. Stinstra et al. has proposed a new method for automatic programming based on the transformation-based pareto heat treatment algorithm [29]. In this method,

linear regression has been used to estimate the coefficients of the transformation functions. The ant programming (AP) method has been developed by Roux and Fonlupt. The AP models the intelligent behavior of ant colonies in finding the shortest path between the food source and the nest [17]. The search process is modeled according to the amount of pheromone, which indicates the distance of the ants from the food source in the AP. The computational complexity of the AP method is high due to the nature of the algorithm. Therefore, if the tree depth in the model increases, the computational cost of the algorithm increases considerably [30]. The artificial immune system programming has also been presented in the literature as an alternative method for solving symbolic regression problems [20]. Artificial immune system programming has been used in a lot of paper by other researchers and new versions have been developed [31–33]. It has been observed that some developed versions of the artificial immune system programming method have a local minima problem due to fast convergence[34]. Behnood and Golafshani developed a method to predict the dynamic modulus of asphalt mix based on bio-geography based programming method [23]. Thanks to the mentioned method, the researchers estimated the dynamic modulus of asphalt mixtures without the need for laboratory-based measurements and expensive equipment. Since this method is developed for a specific problem, there is no knowledge to general-purpose automatic programming problems.

The artificial bee colony programming (ABCP) method, which models the foraging behavior of bees and is based on swarm intelligence, was developed by Karaboğa et al [18]. In this algorithm, the colony is divided into three groups as employer, onlooker and scout bees. After leaving the colony, the employer bee goes to the responsible food source. When the employer bees return to the colony, the employer bees share the information about the food source to the onlooker bee over a series of dance movements. The Onlooker bees understands the location and the quality of the food source by watching this dance. Onlooker bees are responsible for choosing between food sources based on quality. The scout bees is responsible for finding new food sources by randomly around of colony [18]. The basic artificial bee colony algorithm is more successful in numerical optimization problems [35]. However, ABCP method can reach the best solution with high computational cost in symbolic regression problems.

The ABC algorithm is suitable for parallelization. When an algorithm is to be parallelized, new techniques should be developed or existing structures should be updated. In his work, Aslan introduced a new migrant creation strategy called the swap model [36]. The main idea behind the swap model is based on directly using the information sent by the topological neighbor to modify the best solution of the current subcolony. From the comparisons, it is concluded that the parallelization of ABC with the swap model significantly increases the convergence speed of the algorithm while preserving the qualities of the solutions, speedup and efficiency values [36].

The Immune Plasma Algorithm (IPA) is newly proposed a population-based metaheuristic optimization algorithm by Aslan in 2020 [37]. IPA has been inspired by immune plasma therapy. The obtained antibodies from the recovered patient has been used to treat by transferring antibodies to weak individuals in the immune plasma therapy. The success of the IPA in complex numerical and engineering problems supports the idea that IPA based automatic programming approach can also be successful in symbolic regression problems.

There exists a IPA-based programming approach in the literature. The information-sharing mechanism of the ABCP to the immune plasma programming was adopted by Arslan [38]. The performance of the mentioned method was examined just on the Box-Jenkins dataset without any benchmark test problem. The results of the mentioned were compared with artificial bee colony programming and artificial neural networks. Although the success of the mentioned method is close to the test error of artificial neural networks, the mentioned method is substantially more unsuccessful than ABCP [38]. Moreover, the fundamental step of the mentioned method has not been not clearly presented and also there not exist detailed the implementation concept of the mentioned method in the paper [38]. For this reason, this method is not suitable for a fair comparison.

When the IPA based an automatic programming method develop, it can be predicted that the method can be overcome the compared to the above-mentioned weaknesses of other algorithms. An effective automatic programming method with closest to the global solution and low computational cost is aimed to propose the based on IPA in this paper. In this study, Immune Plasma Programming (IPP) has been introduced and its performance has been examined through the solution of symbolic regression problems. Within the scope of the study, two different versions of the IPP algorithm, called as SIPP (Standard Immune Plasma Programming) and the improved version GIPP (Guided Immune Plasma Programming), have been proposed. While the proposed SIPP is based on the standard processing steps of the IP algorithm, the proposed GIPP is based on the starting purely the special initial solutions according to the type of handled problem. The performances of the two proposed methods have been evaluated on different benchmark test problems and compared with competitor algorithms. It is expected that the two proposed methods will be used in solving various real-world regression problems.

In the second part of this paper, the fundamental process of the IPA is presented. in the third part, the proposed automatic programming methods based on the IPA is introduced. In the fourth part, the experimental results and discussion are given. In the final part, a general evaluation has been concluded and future studies presented.

2 Immune Plasma Algorithm

The immune system challenges the virus and tries to protect the body by producing antibodies against a virus that the body has never encountered before [39]. Since an infection begins, the level of antibodies produced by the immune system slowly rises [40]. After a certain time, the level of antibodies reaches its peak and starts to decrease again. Meantime a similar virus is encountered, the immune system starts to react faster because of the antibody level is higher than the first encounter. Individuals with a weak immune system or with reduced immunity are also weak in resisting viruses. Therefore,these individuals cannot produce enough antibodies to challenge the disease. The antibody-rich portion of the blood extracted from recovered patients is defined as plasma. The plasma from recovered patients can be used as a source for individuals exposed to infection. The process of extracted antibodies from recovered individuals exposed to the same infection and transferring of extracted antibodies to weak individuals is described as immune plasma therapy. This biological and powerful process behind immune plasma therapy are composed the basis of a new meta-heuristic optimization algorithm developed by Aslan [37].

The Immune Plasma algorithm (IPA) is a population-based meta-heuristic algorithm. Each individual in the population is correspond to a possible solution for the problem. Moreover, The amount of antibodies produced by individuals in the population is correspond to the quality of the solution. In IPA, the spread of infection among individuals represents the exploration stages of the algorithm, and the plasma transfer represents the exploitation stages of the algorithm.

2.1 Generating Initial Individuals

Each individual of the IPA is correspond to a possible solution for the handling problem. For this purpose, Eq. (1) is used to create the initial population. Assuming that the optimization problem D is the different decision parameter to be solved, kth individuals of the PS population represented by x_k are created using Eq. (1) [37]. Eq. (1), x_{kj} is matched with the decision parameter jth of x_k. Also, x_j^{low} and x_j^{high} are the lower and upper bounds of the jth parameter, respectively. Finally, $rand(0,1)$ is a randomly determined number between 0 and +1 [37].

$$\begin{gathered} x_{kj} = x_j^{low} + rand(0,1)(x_j^{high} - x_j^{low}) \\ k = \{1,2,\ldots,PS\} \; and \; j = \{1,2,\ldots,D\} \end{gathered} \tag{1}$$

where, D is the parameter space of the problem, x_k is the index of the k^{th} indivudual of PS population.

2.2 Infection Spreading and Immune System Response

The spread of infection among individuals is provided to expand the research space In the IPA. The immune system of infected individuals develops an

immune response to protect against viruses. The spread of infection is modeled using Eq. (2) in IPA [37].

$$\begin{gathered} x_{kj}^{inf} = x_{kj} + rand(-1,+1)(x_{kj} - x_{mj}) \\ k = \{1,2,\ldots,PS\} \; and \; m = \{1,2,\ldots,PS\} - \{k\} \end{gathered} \tag{2}$$

In Eq. (2), x_{kj} is the randomly determined j^{th} parameter of the infected individual x_k. x_{kj}^{inf} is a candidate infected solution for individual x_k. x_{mj} is index of j^{th} of the randomly selected individual x_m [37]. In the infection spreading phase, greedy selection is applied between the infected individual (x_k^{inf}) and the individual (x_k), according to the immune responses which is corresponds to fitness values [37].

2.3 Plasma Extraction and Transfer

In immune plasma therapy, extracted antibodies infected individuals are transferred to individuals with weak immunity. In the IPA inspired by immune plasma therapy, the number of donors (NoD) and the number of receivers (NoR) are determined and the plasma transfer process is initiated [37]. In plasma transfer phases of the IPA, NoD donor and NoR receiver individuals are selected through the population according to fitness values of individuals. The selected donor individuals are correspond to the best NoD individual, while the selected receiver individuals are correspond to the worst NoR individuals. The plasma transfer process of IPA is computed over Eq. (3) [37].

$$\begin{gathered} x_{kj}^{rcv-p} = x_{kj}^{rcv} + rand(-1,+1)(x_{kj}^{rcv} - x_{mj}^{dnr}) \\ j = \{1,2,\ldots,D\} \end{gathered} \tag{3}$$

where individual x_k^{rcv} is a kth receiver individual and x_m^{dnr} is a randomly selected donor individual. After the first dose of plasma transfer, a greedy selection is applied between the plasma transferred individual (x_k^{rcv-p}) and the donor (x_m^{dnr}) [37]. If the immunity value of x_k^{rcv-p} is better than x_{mj}^{dnr}, the transferred receiver x_k^{rcv} is replaced with x_k^{rcv-p} and the plasma transfer process continues. Otherwise, the donor x_m^{dnr} is replaced by the receiver x_k^{rcv} and the plasma transfer process is terminated [37]. As the same time, after the first dose of plasma transfer, the decision to continue the plasma transfer process is depends on the immune response of x_k^{rcv-p}. If x_k^{rcv-p}'s immune response is worse than x_k^{rcv}, plasma therapy is complete. Otherwise, x_k^{rcv-p} is replaced with x_k^{rcv} and a second dose of plasma is transferred.

2.4 Control Stage of Immune Memory of Donors

The immune response of donors at the plasma transfer phases to infection may decrease over time in the real world. The changes of immune response of the donors is modelled in IPA by Eq. (4).

$$x_{mj}^{dnr} = x_{mj}^{dnr} + rand(-1,+1)x_{mj}^{dnr} \tag{4}$$

The time-dependent variation of donors immune response is evaluated by dividing the number of calls of the fitness function (t_c) by the maximum number of evaluations (t_{max}). If the value (t_c/t_{max}) determined for each donor is less than the randomly determined number between 0 and 1, the donor is completely replaced according to Eq. (1) . Otherwise, immune memory is checked by updating each parameter of individual x_m^{dnr} using Eq. (4).

The pseudo code of the IP algorithm, including the generation of the initial population, the spread of the infection, and the plasma transfer operations, is presented in the Alg. (1) [37].

Algorithm 1 Fundamental steps of the IPA

Assign values to PS, D, NoD and NoR control parameters.
Set x_{best} as the best individual of the PS individuals.
Select t_{max} and set t_{cr} to PS.
while $t_{cr} < t_{max}$ **do**
  //Infection distribution
  for $k \leftarrow 1 \ldots PS$ **do**
    if $t_{cr} < t_{max}$ **then**
      $t_{cr} \leftarrow t_{cr} + 1$ and $x_k^{inf} \leftarrow$ infect x_k with x_m using Eq. (2)
      if $f(x_k^{inf}) < f(x_k)$ **then**
        Update x_k with x_k^{inf}
        Update x_{best} with x_k if $f(x_k) < f(x_{best})$
      end if
    end if
  end for
  //Plasma transfer for critical individuals
  $doseControl[1 \ldots NoR] \leftarrow$ set each element to 1
  $dIndexes[1 \ldots NoD] \leftarrow$ get the indexes of the donors
  $rIndexes[1 \ldots NoR] \leftarrow$ get the indexes of the receivers
  $treatmentControl[1 \ldots NoR] \leftarrow$ set each element to 1
  for $i \leftarrow 1 \ldots NoR$ **do**
    $k \leftarrow rIndexes[i]$ and $m \leftarrow$ a random element from $dIndexes$
    $x_k^{rcv}, x_m^{dnr} \leftarrow$ get the kth and mth individuals from the population
    while $treatmentControl[i] == 1$ **do**
      if $t_{cr} < t_{max}$ **then**
        $t_{cr} \leftarrow t_{cr} + 1$ and $x_k^{rcv-p} \leftarrow$ plasma treatment to x_k^{rcv} with x_m^{dnr} using Eq. (3)
        if $doseControl[i] == 1$ **then**
          if $f(x_k^{rcv-p}) < f(x_m^{dnr})$ **then**
            $doseControl[i] \leftarrow doseControl[i] + 1$
            Update x_k^{rcv} with x_k^{rcv-p}
          else
            Update x_k^{rcv} with x_m^{dnr}
            Set $treatmentControl[i]$ to 0 for completing transfer
          end if
        else
          if $f(x_k^{rcv-p}) < f(x_k^{rcv})$ **then**
            Update x_k^{rcv} with x_k^{rcv-p}
          else
            Set $treatmentControl[i]$ to 0 for completing transfer
          end if
        end if
        Update x_{best} with x_k^{rcv} if $f(x_k^{rcv}) < f(x_{best})$
      end if
    end while
  end for
  //Donor update
  for $i \leftarrow 1 \ldots NoD$ **do**
    if $t_{cr} < t_{max}$ **then**
      $t_{cr} \leftarrow t_{cr} + 1$ and $m \leftarrow dIndexes[i]$
      $x_m^{dnr} \leftarrow$ get the mth individual from the population
      if $(t_{cr}/t_{max}) < rand(0,1)$ **then**
        Update x_m^{dnr} using Eq. (4)
      else
        Update x_m^{dnr} using Eq. (1)
      end if
      Update x_{best} with the x_m^{dnr} if $f(x_m^{dnr}) < f(x_{best})$
    end if
  end for
end while

3 Proposed Immune Plazma Programming

In this section, proposed Standard Immune Plasma Programming (SIPP) and Guided Immune Plasma Programming (GIPP) algorithms presented for symbolic regression problems. The proposed SIPP is developed on the native operations of IPA to find the best mathematical formula that represents the relationship between input values and output values. Each individual in the population is correspond to a candidate solution the mathematical equations and its data structures of each individuals are designed as tree in the SIPP. Although the performance of SIPP is satisfied, it has been observed that the SIPP does not provide the expected success in experimental on some test problems. In line with the modeling studies on the proposed SIPP method, the more successful the GIPP has been developed. The main difference between the proposed SIPP and GIPP is the guided initial operation is utilized in the GIPP. The initial population of the GIPP is generating according to the type of benchmark test problems. In initialization process of GIPP, the sets of the functions and terminals are limited according the natures of the handling problems. In the experimental results, it has been observed that more satisfied results have been obtained with GIPP than the competitor in the literature.

3.1 Proposed Standard Immune Plazma Programming (SIPP)

The proposed SIPP is a automatic programming method that adapts the IPA method to solve regression problems. In the SIPP method, each individual represents a solution to the problem that is being tried to be solved. Solutions to symbolic regression problems are represented in a special tree structure introduced by Koza [15]. The Koza tree structure consists of interconnected terminal, function, and expression nodes. Terminal nodes are leaf nodes at the end of the tree that have no children. Function nodes are nodes that connect terminal or expression nodes and represent the operation between them. A sample solution is illustrated in Fig. (1) to the symbolic regression problem for the $(cos(x)+(y-1))$ function. The nodes appearing on the tree are connected to each other to represented a mathematical formula. The trees are read in inorder.

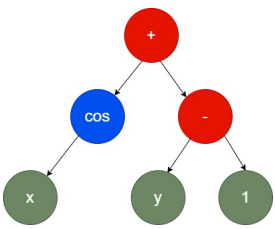

Fig. 1 A sample illustration of the Koza tree

The SIPP creates the initial population using the ramped half and half formatting method, which is a combination of full and grow methods. The initial minimum and maximum depth parameters are set for initialization of the individuals as tree before the SIPP is started. The Depth values of the individuals represent the level of trees. All nodes between the root node and the minimum depth must be functional nodes. The full method continues to generate functional nodes until it reaches the initial maximum depth and add terminals to last level. The grow method generate nodes with random terminals for minimum depth levels [41]. Two different sample trees are illustrated with full and grow methods in Fig. (2) for initial minimum and maximum depth $2, 4$, respectively.

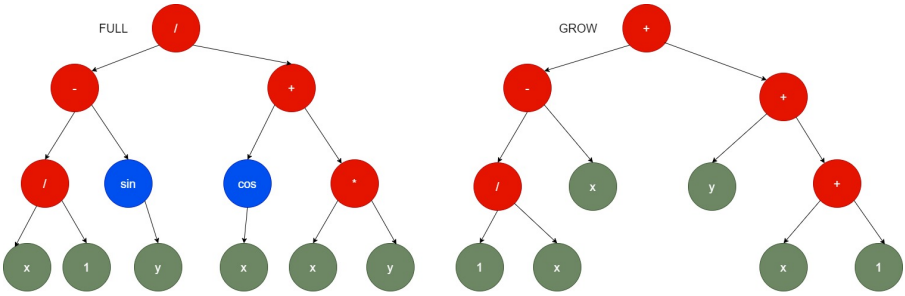

Fig. 2 The samples trees illustration of the Full and Grow methods

In the Grow method, terminal nodes can be found at the third and fourth level, while in the full method, they can only be found at the fourth level in Fig. (2). The half of the population with the grow method and the remaining with the full method are generated in The SIPP. Therefore, the SIPP aims to avoid similarity while initializing the population. After the initial population is performed, the fitness values of each individuals in population are calculated. The fitness values are computed according to Eq. (5). The fitness value depends on the sum of the standard error difference between the targeted ($f(x)$) and generated ($g(x)$) functions for each individual in the population.

$$fit_i = \sum_{n=1}^{k} |f(x_n) - g_i(x_n)| \tag{5}$$

After computing the fitness values of individuals in the population, the infection spreading phase is performed in the SIPP. In the infection spreading phase, primarily each individual in the population is exposed to the infection spread process with a randomly selected individual from the population. After that a randomly selected sub-tree is modified/replaced with randomly initialized sub tree in the SIPP. The infection spreading process is illustrated in Fig. (3). Here, firstly, a sub-branch is selected from tree K and replaced with a randomly selected sub-branch from tree M.

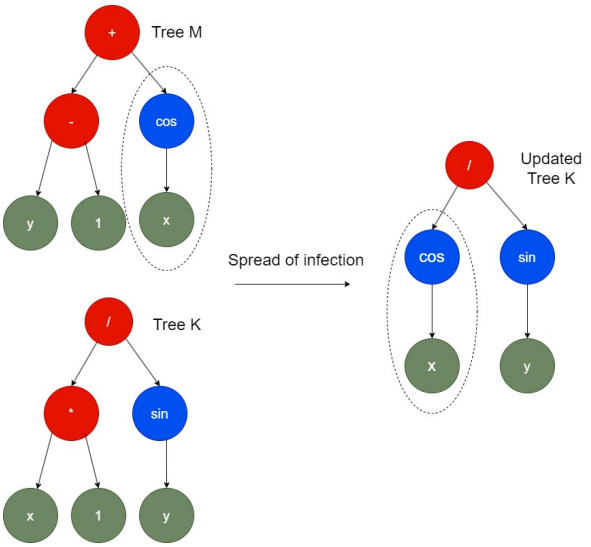

Fig. 3 The processing of infection spreading

After the infection spread, a sub-branch selected from the solution tree of the infected individual is randomly changed. A random modification of individual K's solution tree is illustrated in Fig. (4). Here, it is aimed to update the individual and renew itself by changing a randomly selected sub-branch from the K tree.

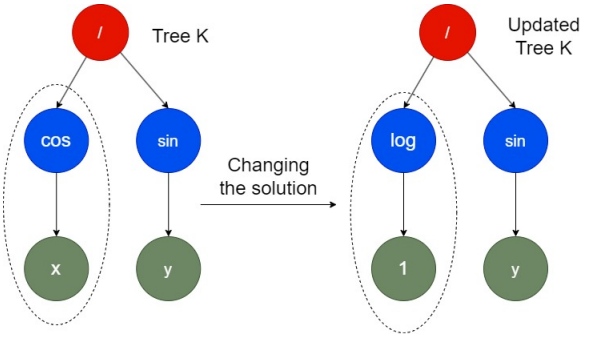

Fig. 4 the processing of random change of individual

After the infection spread phase is completed, the plasma transfer phase is performed in the SIPP. In the plasma transfer phase, a randomly selected sub-tree is transferred from NoD donor individuals to NoR individuals. While the donors are the best NoD individuals according to fitness values, the receivers are the worst NoR individuals according to fitness values in population. Hence,

strengthen the weak individuals in the population and to improve the population gradually are provided. The processing of plasma transfer is illustrated in Fig. (5).

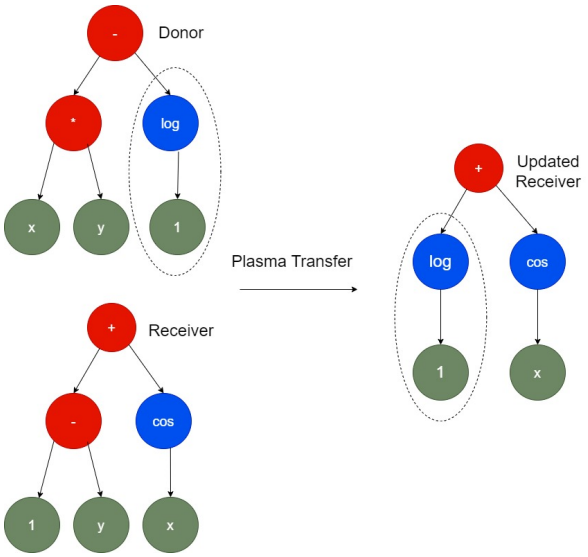

Fig. 5 The illustration of the plasma transfer phase

When same donors are used in the plasma transfer phase, the algorithm can be got stuck the local minimas. To overcome this problem, the donors are replaced with a random solution.Thus, the probability of convergence to the global minimum increases. At this phases, if the value (t_c/t_{max}) for each donor is less than the a randomly number between 0 and 1, a new individual is computed and the candidate individual is replaced with the donor. In the other case, randomly selected sub-tree of the donor is replaced with randomly computed new sub-tree as modification of spread infection.

3.2 Proposed Guided Immune Plasma Programming (GIPP)

Although the fundamental phases of the GIPP are the same as the SIPP method, the GIPP has been improved with the addition of the guiding operation to the initialization phase. The solution-generating ability of the GIPP has been increased thanks to the guided operation. The main purpose of the guided operation is to use only the functional nodes depends on nature of handled problem. Hence, the GIPP is started with more qualified the initial solutions, which are nearer to global solutions.

In the GIPP method, the processes of infection spreading and plasma transfer phases are utilized as the SIPP. However, the donor update phase is not

Algorithm 2 Fundamental steps of the proposed GIPP

```
Assign values to PS, NoD and NoR control parameters.
Set functional nodes to use based on test problems.
Compute the initial population with guided operation.
Calculate the fitness values of initial population.
Set x_best as the best individual of the PS individuals.
Select t_max and set t_cr to PS.
while t_cr < t_max do
    //Infection distribution
    for k ← 1 ... PS do
        Select random individual other than x_k from the population and assign it to the clone variable.
        Copy a subtree from the clone tree and replace it with a sub-branch of x_k. (spread of infection)
        Update the subtree of x_k. (changing the solution)
    end for
    //Plasma transfer for critical individuals
    doseControl[1 ... NoR] ← set each element to 1
    treatmentControl[1 ... NoR] ← set each element to 1
    for i ← 1 ... NoR do
        Select the individual with the lowest fitness value from the population as the receiver and the highest
    fitness value as the donor
        while treatmentControl[i] == 1 do
            Cross using reciver and donor individual and create a new individual named receiver_p.
            if doseControl[i] == 1 then
                if f(receiver_p) < f(donor) then
                    doseControl[i] ← doseControl[i] + 1
                    Update receiver with receiver_p
                else
                    Update receiver with donor
                    Set treatmentControl[i] to 0 for completing transfer
                end if
            else
                if f(receiver_p) < f(receiver) then
                    Update receiver with receiver_p
                else
                    Set treatmentControl[i] to 0 for completing transfer
                end if
            end if
            Update x_best with receiver if f(receiver) < f(x_best)
        end while
    end for
end while
```

used in the GIPP method. The pseudo code of the proposed GIPP is presented in Alg. (2).

First, the GIPP is started to set the control parameters including the total number of donors (NoD), total number of receivers (NoR), population size (PS), number of iterations (t_{max}). Next the initial population is computed on guided operation. Then the fitness values of the initial population are calculated. Final the best individual x_{best} of initial population is selected according to the fitness values in initialization.

After that the infection spread and plasma transfer phases of the GIPP is performed as the infection spread and plasma transfer phases of the SIPP for each individual in the population. In infection spread and plasma transfer phases of the GIPP, A greedy selection process is applied between candidate individuals and source individuals. Therefore, the individual with the highest fitness value is provided to store in the population.

4 Experimental Results and Discussion

In this section, the results of the experimental studies are presented to analyzing of the performances of the proposed SIPP and GIPP. The obtained experimental results have been compared with competitor algorithms. In this

study, the proposed SIPP and GIPP are coded with the C# programming language. The developed frameworks has run on a system with 64-bit Windows 10 operating system, including Intel i7-7500u processor and 8 GB memory.

4.1 The performance of the proposed methods on benchmark symbolic regression problems

To evaluate the performance of the proposed methods, the results of the GIPP have been compared with the results of artificial bee colony programming (ABCP) [18], genetic programming (GP) [42] and the proposed SIPP over symbolic regression benchmark problems. The proposed methods have been applied to a number of the most popular benchmark test problems in the literature. To fair comparison, the GA, ABCP and the proposed methods have been run the same conditions including maximum evaluations and population size. Moreover, to observe the ability of the proposed SIPP and GIPP, large scale and different types problem have been selected including linear, polynomial, trigonometric and logarithmic problems. The mathematical formulas of used benchmark test problems have been presented in Table (1).

Table 1 Symbolic regression benchmark functions.

Problems	Number of random points	Domain
$F_1 = x^3 + x^2 + x$	20	[-1, 1]
$F_2 = x^4 + x^3 + x^2 + x$	20	[-1, 1]
$F_3 = x^5 + x^4 + x^3 + x^2 + x$	20	[-1, 1]
$F_4 = x^6 + x^5 + x^4 + x^3 + x^2 + x$	20	[-1, 1]
$F_5 = sin(x^2)cos(x) - 1$	20	[-1, 1]
$F_6 = sin(x) + sin(x + x^2)$	20	[0, 1]
$F_7 = log(x + 1) + log(x^2 + 1)$	20	[0, 2]
$F_8 = \sqrt{x}$	20	[0, 4]
$F_9 = sin(x) + sin(y^2)$	100	[-1, 1] × [-1, 1]
$F_{10} = 2sin(x)cos(y)$	100	[-1, 1] × [-1, 1]

In the Table (1), the count of randomly generated data for the test and value ranges of the related problem are presented. The values of the control parameters of each benchmark test problem for generating population have been given in the Table (2). We use a protected version of the division that produces 1 as the value if the denominator of the division operation equals zero, and a protected version of the logarithm operation that returns 0 if the data is equal to zero in the rlog operation.

In guided operation of the initialization phases of the GIPP, the functional nodes including $+, -, \times, \div$ are used for the F_1, F_2, F_3 and F_4 test problems. The functional nodes including $+, -, \times, \div, sin, cos$ are used for the F_5, F_6, F_9 and F_{10} test problems. For the F_7 test problem, the functional nodes including $+, -, \times, \div, rlog$ are used. For the F_8 test problem, the functional nodes

including $+, -, \times, \div, sin, cos, exp, rlog$ are used. By using nodes that are suitable for the nature of the problem, individuals with higher fitness values are created and better results are obtained in the GIPP.

Table 2 The control parameters of symbolic regression problems.

Parameter	Value
Initial maximum depth	6
Maximum depth	15
Functional nodes	$+, -, \times, \div$ (protected ver.), sin, cos, rlog, exp
Terminal nodes	1-variable: $[x, 1]$, 2-variables: $[x, y]$
Raw fitness	Sum of absolute error on all fitness cases
Run	100

To evaluate of the standard version of proposed method and to clarify the contribution of guided operation to proposed method, the advised control parameters of the ABCP in [18] are utilized experimental setup. The used control parameters the SIPP and ABCP are presented in Table (3).

Table 3 The used control parameters of SIPP and ABCP

SIPP		ABCP	
Parameter	Value	Parameter	Value
Colony size	40	Colony size	40
NoR	10	Limit	1000
NoD	10		

The selection of NoR and NoD values according to the nature of test problems in the IPA are very important for the success of the IPA in numerical optimization. For this reason, the NoR and NoD values has been set to the test problems in 3 different combinations as 1, PS/2 and PS/4, respectively. The results of the SIPP algorithm for 10 different test problems are given over 100 different run in Table (4).

To compare the performances of the GIPP and SIPP methods, the control parameters of them have been set as equally and run on the same environment. For this purpose, the result of the GIPP with same control parameter to the SIPP are presented over 100 different run in Table (5).

When the analysis the Table 4, it is clearly seen that the best results are obtained from SIPP for F_1 - F_4, F_9 and F_{10} problems when the NoR and NoD parameters are set to 1. In addition for F_5 and F_8 problems, the best results are obtained from the setting of NoR and NoD to 10. Moreover, this table shows that the best result is obtained for the only F_6 when NoR and NoD values set 10. When the Table (5) is analysis, it is clearly seen that the

Table 4 Mean best fitness values of SIPP for some NoR and NoD values, population size = 40.

NoR & NoD	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
1	**1.30**	**1.79**	**2.06**	**2.36**	0.53	1.79	0.76	1.31	**1.37**	**0.26**
10	1.45	1.91	2.06	2.39	0.50	**1.42**	0.70	0.92	1.40	0.31
20	1.55	1.99	2.25	2.50	**0.48**	1.60	**0.67**	**0.90**	1.50	0.28

proposed GIPP found absolute zero in 4 test problems between F_1-F_4 and gives results close to zero in remaining test problems.

Table 5 Mean best fitness values of GIPP for some NoR and NoD values, population size = 40.

NoR & NoD	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
1	0	0	0	0	0.13	0.29	0.53	0.49	0.11	0.06
10	0	0	0	0	0.14	0.28	0.53	0.49	**0.01**	0.06
20	0	0	0	0	0.13	**0.26**	**0.52**	**0.48**	0.09	0.07

In this experiment, the initial minimum depth has not been determined for all problems. In addition, "ramped half and half" as tree creation method has been used. The number of runs for each problem is 100 and the maximum number of evaluations for each run is 80,000. Moreover, the population size is set as 40 and the NoR and NoD numbers for the SIPP have been tested in 3 different combinations as 1, PS/2 and PS/4. To fair comparison, the results of the ABCP algorithm are collected from the [18] publication. In the colony size as 40 and the limit value as 1000 for ABCP are set in [18] .

to compare the results of the proposed methods with literature, results of the ABCP[18] and the GP[42] methods are collected from in the literature. Because The competitor methods have been run the same condition and same benchmark test problems. the best results of the SIPP and the GIPP are selected from Table (4) and Table (5) for comparison. The comparative results are presented in Table (6).

Table 6 Mean error values by ABCP, GP, SIPP and GIPP population size = 40.

Method	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
ABCP[18]	0.21	0.12	0.25	0.15	**0.10**	0.21	**0.08**	0.27	6.16	4.24
GP[42]	N/A	0.30	0.40	N/A	0.27	**0.11**	0.15	**0.25**	1.67	1.19
SIPP	1.45	1.91	2.06	2.39	0.50	1.42	0.70	0.92	1.40	0.31
GIPP	**0**	**0**	**0**	**0**	0.14	0.28	0.53	0.49	**0.01**	**0.06**

As seen in Table (6), the results of the SIPP are better than ABCP in F_9 and F_{10} problems. In F_1 - F_8 problems, ABCP algorithm give more successful

results than the SIPP. It is understood that in trigonometric functions where the number of points is 100, the SIPP can achieve more successful results compared to the results found with ABCP. The GIPP method produce better results than SIPP, GP and ABCP for F_1, F_2, F_3, F_4, F_9 and F_{10}. Moreover, the results of the GIPP generally are better than the SIPP but are relatively same results with ABCP results for F_5, F_6, F_7, F_8. Consequently, the results of the GIPP are better than SIPP in all problems, better than ABCP in 6 of them. better than ABCP in 5 of them, respectively.

In order to analyze the performance of the GIPP in different population size and number of evaluations, the population size has been set as $100, 200, 500 and 1000$, and the number of evaluations was set to be population $\times$ 2000. The results of GIPP for different population sizes are presented in Table (7). It is clearly seen in this table that when the population size is higher, the success of the GIPP increases noticeably. However, the growth of the number of evaluations in GIPP does not directly contribute to the success of the GIPP.

Table 7 Mean best fitness values of the GIPP for various population sizes.

Colony Size	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
100	0	0	0	0.01	0.10	0.17	0.55	0.32	0.03	0.02
200	0	0	0	0	0.06	0.06	0.49	0.16	0	0.008
500	0	0	0	0	0.02	0.01	0.44	0.05	0	0.0003
1000	0	0	0	0	**0.01**	**0.001**	**0.40**	**0.02**	0	**0.0001**

The convergence speed is another important comparison for optimization algorithms. For this reason, the convergence graphics of the SIPP and the GIPP are illustrated over used benchmark symbolic regression problems in Fig. (6). This figure shows that in all test problems except F_7, the GIPP has been converged faster than SIPP. In the F_7 test problem, the SIPP has been converged zero faster than the GIPP. Based on this finding, it can be said that the GIPP converges to the global solution faster than the SIPP.

4.2 The performance of proposed method on the Modeling and Forecasting of real world problem

Time series are data that are collected chronologically according to time from a system. These numerical data are used to modeling a system to predict future situations to improve the system. In this section, the performances of the SIPP and the GIPP are compared using the Box-Jenkins time series dataset which is the well-known dataset for automatic programming. Thus, the ability of proposed methods is evaluated in terms of designed a mathematical model for real worlds problems.

The Box-Jenkins time series dataset has 296 pieces of data representing the amount of carbon dioxide (CO_2) produced by burning gas in the furnace. In

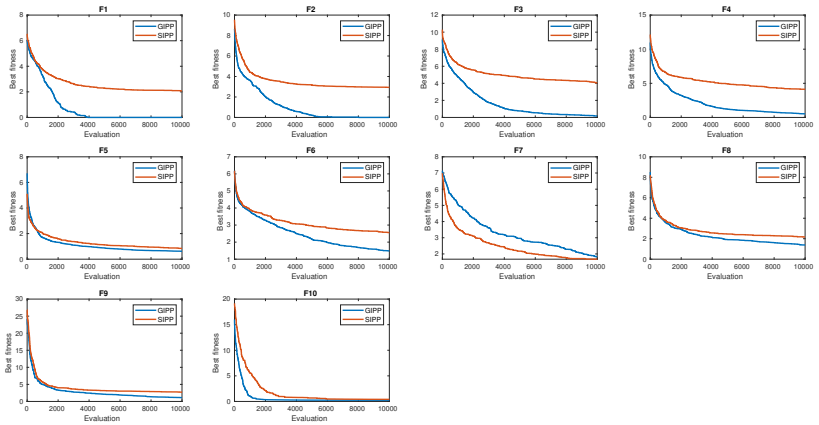

Fig. 6 The convergence graph of the SIPP and the GIPP for F_1-F_{10}

the dataset, the amount of gas burned at time t (U_t) is considered as input, and the amount of CO_2 produced is considered as output (Y_t). the attributes U_t and Y_{t-1} as inputs and the attribute Y_t as the output are used for modeling in the proposed methods. the parameter sets of the SIPP and the GIPP are given in Table (8). Within the scope of this study, two different experimental setups have been created by changing the maximum depth of the trees. The proposed methods have been run with a maximum tree depth as 6 in Experiment 1 and 17 in Experiment 2,respectively. While the maximum of 31 nodes in the trees can be produced due to the maximum tree depth has been set to 6 in Experiment 1, the maximum of 131071 nodes in the trees can be produced owning to the maximum tree depth is set to 17 in Experiment 2. Hence, very complex equations can be obtained by Experiment 2.

Table 8 the parameter sets of the proposed SIPP and GIPP on The Box-Jenkins experiments

Parameter	Experiment	Value
Population Size	1, 2	500
Evaluations Number	1, 2	1000000
NoD, NoR	1, 2	125, 125
Functional nodes	1, 2	$+, -, \times, \div$ (protected ver.), pow
Terminal nodes	1, 2	U_t, Y_{t-1}, R
Initial maximum depth	1, 2	6
Maximum depth	1	6
	2	17

To evaluate the performance of SIPP and GIPP, 290 samples of the dataset are split into two parts. The first 200 samples has been used for training data, the next 90 lines are utilized for test data. The *mse* and *std* values of 30 differently runs have been calculated on training and test data using the

Table 9 The results of the proposed SIPP and GIPP for Box-Jenkins experiments

		SIPP		GIPP	
		Experiment 1	Experiment 2	Experiment 1	Experiment 2
Training Set	MSE.	**0.000788**	0.000766	0.000792	**0.000521**
	STD.	0.000044	0.000067	**0.000029**	**0.000060**
	Best	0.000650	0.000569	**0.000595**	**0.000420**
	Worst	0.000808	0.000808	0.000808	**0.000595**
Testing Set	MSE.	**0.004713**	0.004755	0.004876	**0.002325**
	STD.	0.000965	0.001138	**0.000686**	**0.000660**
	Best	**0.002467**	**0.001448**	0.002685	0.001489
	Worst	0.005172	0.007234	0.005172	**0.003158**

best mathematical model which is produced by each proposed method. The obtained results of the proposed methods are reported in Table (9).

While the GIPP has been produced almost the same *mse* value as the SIPP, The *std* values of te GIPP are better than the the *std* values of te SIPP in experiment 1 as seen in Table (9). However, the *mse* and *std* values of the GIPP are considerably better results than the SIPP in experiment 2.

The prediction graph of the SIPP and the prediction graph of the GIPP are illutrated in Figs. (7)-(8), respectively. As the same time, the prediction graph of the SIPP is presented in Fig. (9) while the prediction graph of the SIPP is shown in Fig. (10). The best mathematical expression produced by the SIPP and the GIPP for experiment 1 and 2 are reported in Eqs. (6)-(9).

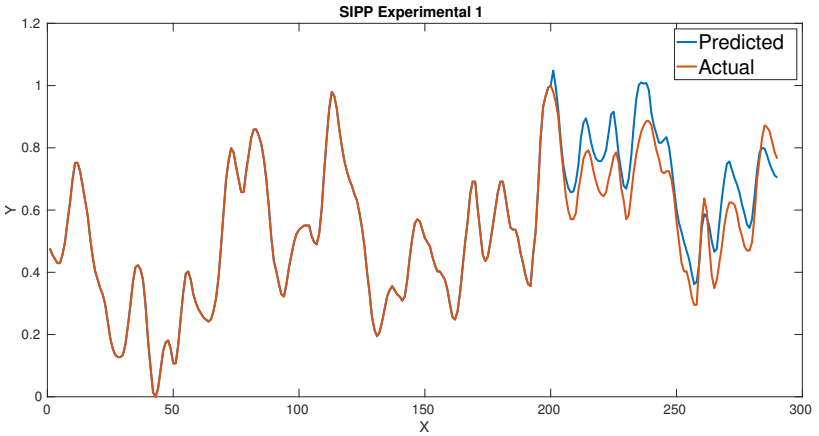

Fig. 7 The graphs of actual and predicted data by the SIPP on Box-Jenkins dataset for experiment 1

$$((y)/((y) + ((sin(exp(y))) * (x)))) \tag{6}$$

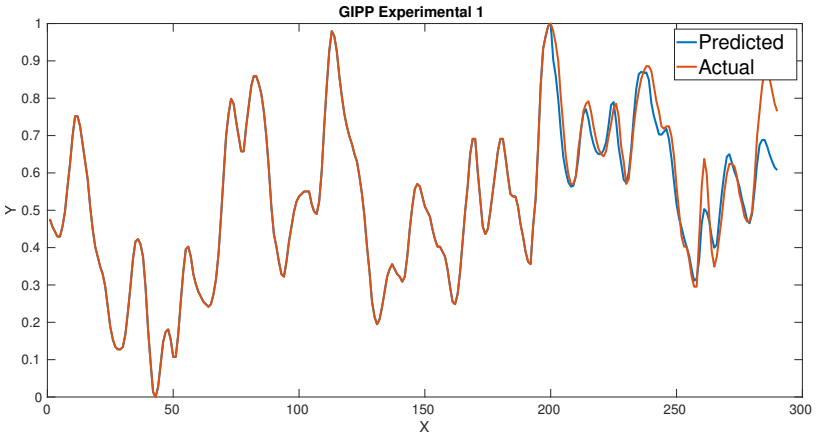

Fig. 8 The graphs of actual and predicted data by the GIPP on Box-Jenkins dataset for experiment 1

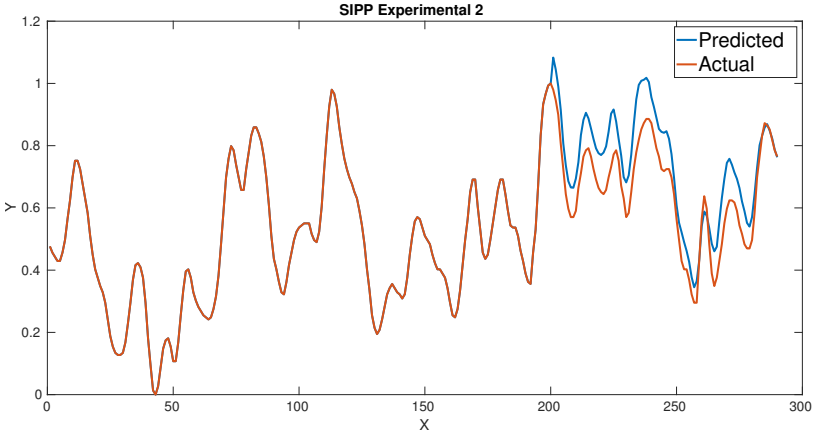

Fig. 9 The graphs of actual and predicted data by the SIPP on Box-Jenkins dataset for experiment 2

$$(((y)+(y))/(((y)+(x))+((x)+((y)/((x)+(y))))))) \tag{7}$$

$$((((0)-(rlog((x)+(y))))/(rlog((cos(y))+(((x)+(x))+((x)+(x)))))+((y)*(1))) \tag{8}$$

$$\begin{aligned}&((((((((((x)+(x)))*((y)))*(((x)*(x)))))*(((((((x)*(y)))))*(((((y)*(y)))\\&+(((x)*(y))))-((x))))))+((((((((((((y)+(x))-(1)))/(((y)+(1))))-(x))\\&+(x)))*((((((((y)+(1))-(1)))))*(((((x)+(y)))*((y)))-((x))))))\\&+((((((((((((y)+(1))-(1))))/(((x)+(y)))))-(x))+(x)))))))))\end{aligned} \tag{9}$$

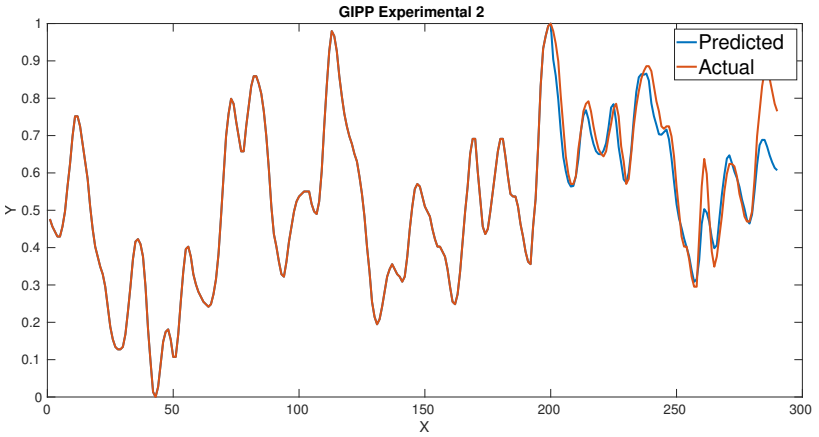

Fig. 10 The graphs of actual and predicted data by the GIPP on Box-Jenkins dataset for experiment 2

4.3 The Computational Time of The Proposed Methods

The computational time is very important metric for the artificial intelligence methods. For this reason, the computational times of the proposed methods are reported in Table (10). To examine the computational times of the proposed methods, the population size was set to 40, the number of evaluations was set to 80,000, and the NoR and NoD were set as 1, 10, and 20, respectively.

Table (10) shows the running times of the proposed methods for each evaluation for the different NoR and NoD values. As it can be seen in the table, the running time of the proposed methods increases as the NoR and NoD values increase. While the both proposed methods run the fastest when the control parameters NoR and NoD are set to 1. It seen from this table that When the computational times of the GIPP has been compared with the computational time of the SIPP, the running time the SIPP is less than the running time of GIPP for F_1 - F_8 test problems. In addition for the F_9 and F_{10} test problems, the GIPP has run relatively faster than the SIPP.

Table 10 Average computational time of the SIPP and GIPP on 30 different NoR and NoD values

Methods	NoR- NoD	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
SIPP	1	6.08	6.04	6.91	6.44	10.49	6.17	6	7.17	10	10
	10	10	10	10.99	10.02	9.02	10.02	9	9	13.01	12.37
	20	11.47	11.95	12.36	12	11.95	11.29	11	11	15.94	15
GIPP	1	14.35	18.07	20.64	22.34	15.14	13.94	13.41	12.49	9.93	9.3
	10	15.98	19.32	22.34	24.48	17.12	15.08	13.84	14.52	10.78	9.98
	20	17.61	21.14	23.89	27.23	18.45	16.87	13.76	16.45	11.55	9.76

The obtained run times of the SIPP and the GIPP for experiment 1 and experiment 2 on the Box-Jenkins dataset have been presented in Table (11). As can be seen from this table, the computational time of the SIPP is less than the computational time of the GIPP in both experiments.

Table 11 Average computational time of the SIPP and the GIPP on Box-Jenkins dataset

Method	Computational time
SIPP for Experimental 1	345
SIPP for Experimental 2	336.3
GIPP for Experimental 1	365.0571
GIPP for Experimental 2	753.4

5 Conclusion

In this study, a new automatic programming method based on Immune Plasma algorithm (IP algorithm or IPA) was proposed to be used in solving symbolic regression problems. The proposed method, called Immune Plasma Programming (IPP), enables the mathematical expression to automatically find the relationship between the input and output values of a system. The proposed approach consists of two models as SIPP and GIPP. Both SIPP and GIPP methods have been analyzed on the well-known Box-Jenkins dataset and several other benchmark test problems. Thus, the proposed methods are evaluated in synthetic problems and real-world problem. In addition, a detailed comparison has been realized with the GA and the ABCP, which are frequently used in the literature. Considering the successful performance of the proposed new approach, it is expected to be used by researchers in optimization and symbolic regression problems. It is aimed to improve the performance of the proposed SIPP and GIPP methods as a future study and to develop new versions.

Funding

This study was not funded by any organization.

Declaration of Conflicts

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and material

No data was used for the research described in the article.

Authors' contributions

Conceptualization, Badem Hasan (B.H.), Bayır Suat (B.S). and Aslan Selçuk (A.S.); methodology, B.S., B.H. and A.S; software, B.S. and B.H.; validation, B.S., B.H. and A.S.; formal analysis, B.S., B.H. and A.S.; investigation B.S., B.H. and A.S.; resources, B.S. and A.S;.; data curation, B.S., B.H. and A.S.; writing—original draft preparation, B.S. and A.S;.; writing—review and editing, B.S., B.H. and A.S;.; visualization, B.S. and B.H.; supervision,

B.H. and A.S;.; project administration, B.H.; funding acquisition, B.S., B.H. and A.S; All authors have read and agreed to the published version of the manuscript.

Ethics approval
Not applicable.
Consent to participate
Not applicable.
Consent for publication
Not applicable.
Code availability
Not applicable.

References

[1] Burlacu, B., Yang, K., Affenzeller, M.: Population diversity and inheritance in genetic programming for symbolic regression. Natural Computing, 1–36 (2023)

[2] Schmidt, M.D., Lipson, H.: Co-evolving fitness predictors for accelerating evaluations and reducing sampling. Genetic Programming Theory and Practice IV **5** (2006)

[3] Billard, L., Diday, E.: Symbolic regression analysis. In: Jajuga, K., Sokołowski, A., Bock, H.-H. (eds.) Classification, Clustering, and Data Analysis, pp. 281–288. Springer, Berlin, Heidelberg (2002)

[4] Mladenovic, N., Rusetskaya, O., Elleuch, S., Jarboui, B.: How the healthcare expenditure influences the life expectancy: Case study on russian regions. In: Masmoudi, M., Jarboui, B., Siarry, P. (eds.) Artificial Intelligence and Data Mining in Healthcare, pp. 71–82. Springer, Cham (2021)

[5] Liu, H., Zhang, Z.: Probing the carbon emissions in 30 regions of china based on symbolic regression and tapio decoupling. Environmental Science and Pollution Research **29**(2), 2650–2663 (2022)

[6] Firouzi, B., Abbasi, A., Sendur, P.: Improvement of the computational efficiency of metaheuristic algorithms for the crack detection of cantilever beams using hybrid methods. Engineering Optimization **54**(7), 1236–1257 (2022)

[7] Vázquez, D., Guimerà, R., Sales-Pardo, M., Guillén-Gosálbez, G.: Automatic modeling of socioeconomic drivers of energy consumption and pollution using bayesian symbolic regression. Sustainable Production and Consumption **30**, 596–607 (2022)

[8] Ansari, M., Gandhi, H.A., Foster, D.G., White, A.D.: Iterative symbolic regression for learning transport equations. AIChE Journal, 17695 (2022)

[9] Zhao, X., Zhou, G.: Dynamic schedule model and algorithm optimization method for linear engineering projects. Engineering Optimization, 1–24 (2023)

[10] Danai, K., La Cava, W.G.: Controller design by symbolic regression. Mechanical Systems and Signal Processing **151**, 107348 (2021)

[11] Altay, O., Cetindemir, O., Aydogdu, I.: Size optimization of planar truss systems using the modified salp swarm algorithm. Engineering Optimization, 1–17 (2023)

[12] Yamashita, G.H., Fogliatto, F.S., Anzanello, M.J., Tortorella, G.L.: Customized prediction of attendance to soccer matches based on symbolic regression and genetic programming. Expert Systems with Applications **187**, 115912 (2022)

[13] Liu, Y., Zuo, X., Li, X., Nie, S.: A genetic algorithm with trip-adjustment strategy for multi-depot electric bus scheduling problems. Engineering Optimization, 1–20 (2023)

[14] Cai, W., Pacheco-Vega, A., Sen, M., Yang, K.-T.: Heat transfer correlations by symbolic regression. International Journal of Heat and Mass Transfer **49**(23-24), 4352–4359 (2006)

[15] Koza, J.: Genetic programing. Programing of Computers by Means of Natural Selection (1992)

[16] Jalal, F.E., Xu, Y., Iqbal, M., Jamhiri, B., Javed, M.F.: Predicting the compaction characteristics of expansive soils using two genetic programming-based algorithms. Transportation Geotechnics **30**, 100608 (2021)

[17] Roux, O., Fonlupt, C.: Ant programming: or how to use ants for automatic programming. In: Proceedings of ANTS, vol. 2000, pp. 121–129 (2000). Springer Berlin

[18] Karaboga, D., Ozturk, C., Karaboga, N., Gorkemli, B.: Artificial bee colony programming for symbolic regression. Information Sciences **209**, 1–15 (2012)

[19] Görkemli, B., Citakoglu, H., Haktanir, T., Karaboga, D.: A new method based on artificial bee colony programming for the regional standardized intensity–duration–frequency relationship. Arabian Journal of Geosciences **15**(3), 1–19 (2022)

[20] Johnson, C.G.: Artificial immune system programming for symbolic regression. In: European Conference on Genetic Programming, pp. 345–353 (2003). Springer

[21] Vincent, F.Y., Qiu, M., Pan, H., Chung, T.-P., Gupta, J.N.: An improved immunoglobulin-based artificial immune system for the aircraft scheduling problem with alternate aircrafts. IEEE Access **9**, 16532–16545 (2021)

[22] Golafshani, E.M.: Introduction of biogeography-based programming as a new algorithm for solving problems. Applied Mathematics and Computation **270**, 1–12 (2015)

[23] Behnood, A., Golafshani, E.M.: Predicting the dynamic modulus of asphalt mixture using machine learning techniques: An application of multi biogeography-based programming. Construction and Building Materials **266**, 120983 (2021)

[24] Chaabene, W.B., Nehdi, M.L.: Genetic programming based symbolic regression for shear capacity prediction of sfrc beams. Construction and Building Materials **280**, 122523 (2021)

[25] Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.: Improving model-based genetic programming for symbolic regression of small expressions. Evolutionary computation **29**(2), 211–237 (2021)

[26] Mundhenk, T.N., Landajuela, M., Glatt, R., Santiago, C.P., Faissol, D.M., Petersen, B.K.: Symbolic regression via neural-guided genetic programming population seeding. arXiv preprint arXiv:2111.00053 (2021)

[27] Al-Helali, B., Chen, Q., Xue, B., Zhang, M.: Multitree genetic programming with new operators for transfer learning in symbolic regression with incomplete data. IEEE Transactions on Evolutionary Computation **25**(6), 1049–1063 (2021)

[28] Ghazouani, H.: A genetic programming-based feature selection and fusion for facial expression recognition. Applied Soft Computing **103**, 107173 (2021)

[29] Bringmann, B., Kramer, S., Neubarth, F., Pirker, H., Widmer, G.: Transformation-based regression. In: Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002), pp. 59–66 (2002). Morgan Kaufmann

[30] Zhou, X., Ma, H., Gu, J., Chen, H., Deng, W.: Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. Engineering Applications of Artificial Intelligence **114**, 105139 (2022)

[31] Gan, Z., Chow, T.W., Chau, W.N.: Clone selection programming and its application to symbolic regression. Expert Systems with Applications **36**(2), 3996–4005 (2009)

[32] Bernardino, H.S., Barbosa, H.J.: Grammar-based immune programming. Natural Computing **10**(1), 209–241 (2011)

[33] Bernardino, H.S., Barbosa, H.J.: Grammar-based immune programming for symbolic regression. In: International Conference on Artificial Immune Systems, pp. 274–287 (2009). Springer

[34] Park, H., Choi, J.E., Kim, D., Hong, S.J.: Artificial immune system for fault detection and classification of semiconductor equipment. Electronics **10**(8), 944 (2021)

[35] Boudardara, F., Gorkemli, B.: Solving artificial ant problem using two artificial bee colony programming versions. Applied Intelligence **50**(11), 3695–3717 (2020)

[36] Aslan, S.: A new emigrant utilization strategy for parallel artificial bee colony algorithm. Evolving Systems **12**(2), 337–357 (2021)

[37] Aslan, S., Demirci, S.: Immune plasma algorithm: a novel meta-heuristic for optimization problems. Ieee Access **8**, 220227–220245 (2020)

[38] Arslan, S.: Zaman serisi tahmin probleminin immün plazma programlama kullanılarak çözülmesi. Avrupa Bilim ve Teknoloji Dergisi (29), 219–224 (2021)

[39] Salvador, A.F., de Lima, K.A., Kipnis, J.: Neuromodulation by the immune system: a focus on cytokines. Nature Reviews Immunology **21**(8), 526–541 (2021)

[40] Ioannidis, J.P.: Reconciling estimates of global spread and infection fatality rates of covid-19: an overview of systematic evaluations. European journal of clinical investigation **51**(5), 13554 (2021)

[41] Soule, T., Heckendorn, R.B., Shen, J.: Solution stability in evolutionary computation. In: Proceedings of The 17th International Symposium on Computer and Information Sciences, pp. 237–241 (2022). CRC Press

[42] Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Phong, D.N.: On the roles of semantic locality of crossover in genetic programming. Information Sciences **235**, 195–213 (2013)