Parameter Tuning of PI-controller with Bat Algorithm

Dušan Fister
University of Maribor, Faculty of Mechanical Engineering
Smetanova 17, 2000 Maribor
E-mail: dusan.fister@student.um.si

Riko Šafarič, Iztok Jr. Fister and Iztok Fister
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor

Keywords: PI-controller, nature-inspired algorithms, optimization

Received: January 6, 2016

Correct input controller parameter settings are vital and in constant connection with output functions - e.g. robotic positioning. Optimal positioning of robotic arm automatically provides a high level of safety and functionality. The first prevents robot from hurting any people around or even itself, while the second ensures robot advantage. In order to improve both safety and functionality, we propose two nature-inspired algorithms for parameter tuning of PI-controller and test them on the laboratory robotic manipulator. However the manipulator is not designed to perform a real robotic work, it offers a detailed approach of positioning control. Our goal is to access the positioning control unit and combinatorially set the input controller parameters with the help of two implemented algorithms. This principle is called automatic parameter tuning, which firstly tests the corresponding setting, then evaluates it and finally tries to improve former result with new one.

Povzetek: Za natančno pozicioniranje zahteva robotski regulator pravilno nastavljene parametre. Ti zagotavljajo varno in funkcionalno delovanje robota. S testiranji, opisanimi v nadaljevanju, želimo določiti optimalne konstante regulatorja z algoritmom za nastavljanje parametrov, ki sloni na nelinearnem, dvoosnem robotskem mehanizmu. Implementirana optimizacijska algoritma temeljita na vzorih iz narave in določata parametre avtomatsko, brez človeške interakcije. Naš pristop je iterativen, kar pomeni, da se želimo s kombinatoričnim ugotavljanjem čimbolj približati idealni rešitvi, ki pa je sicer ne moremo doseči. Avtomatski postopek nastavljanja parametrov z optimizacijskim algoritmom predstavlja odskočno desko za zagotavljanje varnosti ter funkcionalnosti, povečanega obsega dela robota ter višje natančnosti in kakovosti izdelkov.

1 Introduction

A robot is typically an electro-mechanical device controlled by a computer program. It operates in an environment which can be changed using actions for whether it is delegated. The robot performs repeated actions that were before executed by humans. They have been displaced and upgraded by robots especially by performing dangerous tasks, e.g., coating cars in the automotive industry, aeronautics, etc.

A robotic arm, for instance, is moved and positioned using a closed-control loop. The control loop consists of a controller and a control plant. The controller is a part of the electrical scheme, which controls a mechanical part of the robotic arm (control plant). The control plant consists of electrical motors to lift and lower the arm. However, this process is subject of gravity forces.

Typically, the control loop is implemented by so called PID-controllers in the real-world applications (Fig. 1). This controller calculates an error value e between desired input value y_{des} and actually measured output y_{act}. Then, a derivative and integral of the error signal is calculated. Actually, the output signal u is obtained as follows

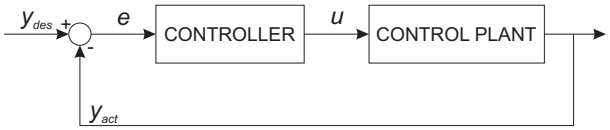

Figure 1: Scheme of a robot.

$$u = K_P(y_{des} - y_{act}) + K_I \int (y_{des} - y_{act}) + K_D(y_{des} - y_{act})'. \quad (1)$$

Eq. (1) consists of three parts, i.e., proportional, integral and derivative terms weighted by corresponding proportional gain K_P, integral gain K_I and derivative gain K_D, respectively. The control signal u is sent to the control plant in order to obtain the new output y that serves as the new y_{act} value for generating the new error signal e. This process continues until equilibrium is achieved. Let us notice

that the desired values are input variables that represent a reference generator output, while the desired values are obtained as a feedback of the control plant on the input variables. There are more types of the reference generators, e.g., micro-controllers, DSP, FPGA, etc. A discrete equation of PID-controller can be written (Eq. 2).

$$\begin{aligned} u(k) =& u(k-1) + q_0 \cdot e(k) + \\ & q_1 \cdot e(k-1) + q_2 \cdot e(k-2), \end{aligned} \tag{2}$$

where

$$q_0 = K_P \cdot (1 + K_I + K_D), \tag{3}$$

$$q_1 = -K_P \cdot (1 + 2 \cdot K_D - K_I)\ and \tag{4}$$

$$q_2 = K_P \cdot K_D. \tag{5}$$

Only PI-controller type is used for our application. In line with this, K_D gain should be set to zero. Eq. 3-5 can be then simplified to new form (Eq. 6-8):

$$q_0 = K_P \cdot (1 + K_I), \tag{6}$$

$$q_1 = -K_P \cdot (1 - K_I)\ and \tag{7}$$

$$q_2 = 0. \tag{8}$$

In the next chapters, parameters q_0 and q_1 of PI-controller will be optimized.

There are few strategies for parameter tuning of robotic controller. Using Bode plotting [16] and root locus method [4] a linear controller can be tuned. For non-linear control plants, an iterative approach of tuning should be employed. In this method, random parameters are entered into the robotic controller and according to mechanic response of manipulator little corrections are made through more iterations. A new set is then entered into controller and so on. The basic and the simplest strategy of tuning is a manual approach. Requires an experienced and patient engineer, what makes this strategy time-consuming. It can be automated using the micro-controller, which makes the process faster up to few times, but then an optimization algorithm is required. Every algorithm is intended to find an optimal solution of the problem, so the question is, how fast can algorithm approach to an ideal solution? Today, many algorithms are widely-known. They differ to each other by the ease of use, complexity, principle of working and most importantly, convergence speed. The last parameter could be simply compared to algorithm's efficiency. The fact is, faster than the algorithm is searching, more local is the search space becoming and slower it is searching, more global it can go.

In previous century a greater demand of quality, quantity and efficiency of making products was sensed. As a result, an optimization has been became more and more important. Using computer guided optimization, controlling machines have became easier and even more precisely to use. Optimization algorithms are today frequently and widely used in order to maximize quality and quantity of products, to minimize the production costs as well as increase the functionality, safety and duration of services. For solving the real-world problems, where the domain-specific knowledge is absent, a general problem solvers have been emerged. Today, evolutionary and swarm intelligence algorithms act increasingly in that role.

Basic principles of evolutionary algorithms (EA) were discovered a bit longer ago. In 1871, Charles Darwin published an article about natural selection [2]. Alan Turing was the first who successfully implemented the algorithm [20], that based on results of Charles Darwin's work. He implemented an optimizational algorithm, named $genetic\ search$, and has later also strived for other topics of artificial intelligence. His work was upgraded by John Holland in 1988, who created a $genetic\ algorithm$ (GA) that is today one of the most often worldwide used evolutionary algorithm [13].

On the other hand, a new way of optimization was being commenced in 1995, called the Swarm Intelligence (SI). Particle Swarm Algorithm (PSO) invented by Russell Eberhart and James Kennedy [3] became quickly widely used. Interestingly, the SI-based algorithms use the biological and social relations, since individuals collaborate between each other by learning of experiences. Many SI-based algorithms are known, e.g. ant colony, bee colony, bird flocks, bats, cuckoo search, termites and fish schools. The Bat algorithm (BA) is one of the newest, since it was proposed in 2010 by Yang. The BA quickly widened for testing purposes on various applications [23]. Firstly on numeric and discrete applications, after that also for multi-objective optimization [24]. The possibility of constrained optimization was proved by Gandomi et. al. [10]. The BA was hybridized with Differential Evolution strategies in 2013 by Fister et. al. [9] as well as by Random Forest Regression method [8]. The self-adaptation was proposed by Fister et. al. [5] and [6], which presents one of the most successful BA variants.

The remainder of the paper is divided into next sections: Section 2 presents a control plant and controller. Section 3 describes both nature-inspired algorithms used in this study, i.e., GA and BA.

2 Control plant

In this study, the 2 degree-of-freedom (2 DOF) Selective Compliance Assembly Robot Arm (SCARA) [18] depicted in Fig. 2 was taken into account. The robot arm is controlled in a 2-dimensional space and parameters of this controller are tuned by an optimization algorithm. Since 2005, four different optimization methods were proposed for parameter tuning of the same robot. Albin Jagarinec developed an adaptive regulator in 2005 [15], while Marko Kolar the fuzzy controller in the next year [17]. A neural sliding-mode controller was implemented on the system in the same year by Jure Čas [21]. Finally, the genetic algorithm was tested by Tomaž Slanič [19].

The control plant is responsible for moving the robotic arm that is enabled by two direct-current ESCAP 28 D2R

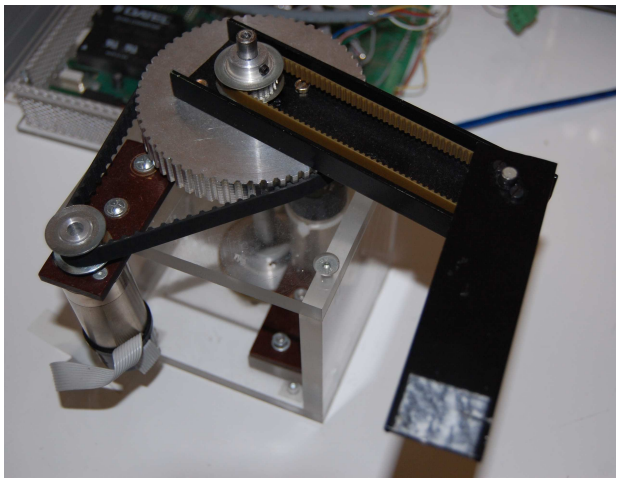

Figure 2: 2-DOF robot.

11 motors and the appropriate power electronics. Opposite to power electronics, two incremental decoders are connected, which transform an incremental encoder's signals from both motors to angles of rotation. These are both processed in a custom input/output interface card based on a digital signal processor (DSP-2 Roby) [22]. Besides decoding signals, the DSP-2 Roby serves also as motor's driver. In addition, the DSP-2 Roby retrieves other basic information, like angles of rotation and time to personal computer, which plots the step responses of the robot and outputs usable information for evaluation of robot arm behavior. Moreover, the optimization algorithm is also being run on this processor.

2.1 Robot's model

The robot's model can be written using Eq. (9) basing on the principles of Lagrangian mechanics, as follows

$$\begin{bmatrix} J_{m1}N_1 + \frac{a_1+a_2\cos(q_2)}{N_1} & \frac{a_3+a_2\cos(q_2)}{N_1} \\ \frac{a_3+a_2\cos(q_2)}{N_2} & J_{m2}N_2 + \frac{a_3+J_{3o}}{N_2} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} + \begin{bmatrix} \frac{-a_2\dot{q_2}(2\dot{q_1}+\dot{q_2})+\sin(q_2)}{N_1} \\ \frac{a_2\sin(q_2)\dot{q_1}^2}{N_2} \end{bmatrix} = \begin{bmatrix} \tau_{mot1} \\ \tau_{mot2} \end{bmatrix}, \tag{9}$$

where J means moment of inertia, l length of handles and m mass of handles with gears. Parameters q_i, $\dot{q_i}$ and $\ddot{q_i}$ mean position (angle of rotation), velocity and acceleration of specific axis [21]. Eq. 10 presents the meaning of parameters a_1, a_2 and a_3.

$$\begin{aligned} a_1 &= I_1 + I_2 + I_4 + m_2 \cdot l_{1T}^2 + (m_3 + m_4) \cdot l_1^2 + m_4 \cdot l_{2T}^2 \\ a_2 &= m_4 \cdot l_1 \cdot l_{2T} \\ a_3 &= I_4 + m_4 \cdot l_{2T}^2 \end{aligned} \tag{10}$$

As seen, equation is a two dimensional, which tends the control plant of robot as non-linear. The equation presents the motor's torque, necessary for correct motion of robot's peak. The full elaboration of this equation is presented in [21]. Note that only proportional and integral gains (PI-controller) were used in our study.

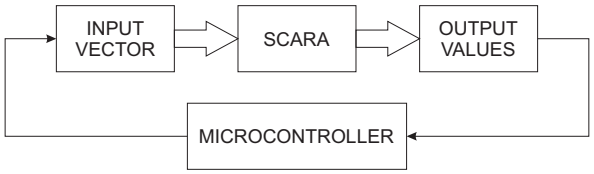

Figure 3: Searching for the optimal parameter setting of the PI-controller.

2.2 Optimization problem

In general, the optimization problem is defined as a quadruple $OP = \langle I, S, f, goal \rangle$ [11], where I presents a set of instances that can be arisen on the input, S is a set of feasible solutions, f objective function and *goal* denotes if the minimum or maximum of the objective function is searched for. The input vector is expressed as

$$\mathbf{x} = \{q_{0,1}, q_{1,1}, q_{0,2}, q_{1,2}\}, \tag{11}$$

where $q_{0,1}$ and $q_{1,1}$ mean the controller input parameters for the first axis and $q_{0,2}$ and $q_{1,2}$ for the second axis of a robotic manipulator. The task of the optimization is to maximize the fitness function, i.e., $max(f_i)$. The fitness function evaluates three different measures obtained as a feedback **y** from the control plant consisting of:

- $Over_i$ - actual overshoot,
- Ess_i - actual steady state error and
- $Time_i$ - actual settling time.

According to mentioned measures, the fitness function is expressed as follows:

$$\begin{aligned} f(\mathbf{y}) = \sum_{i=1}^{2} \frac{1}{2} (E_{1i}(1 - |P_i - Over_i|) + \\ E_{2i}(1 - Time_i) + E_{3i}(1 - Ess_i)), \end{aligned} \tag{12}$$

where E_{ij} are initialized constants representing weights that determine an influence of the specific outputs for each axis in Eq. 12. Obviously, the sum of these three constants of specific axis is:

$$\sum_{i=1}^{3} E_{ij} = 1, \tag{13}$$

where i is the specific output variable and j the specific axis.

The optimization problem can now be defined as searching the best input values in order to obtain the best output values estimated by the fitness function. The process is graphically presented in Fig. 3, from which it can be seen that the optimization algorithm puts the input vector **x** to SCARA robot arm controller that moves and its position is then being measured. The output vector **y** is obtained after the moving and positioning the arm. The value of fitness function is determined from this vector.

3 Algorithms for parameter tuning of PI-controller

The majority of real-world problems with which human are confronted today are NP-hard. This means that the time complexity of solving these problems increases by increasing a problem size. The problem size is estimated by the number of input variables. As a results, when the number of variables increased to some high value, the user can wait for the results indefinitely. Therefore, engineers responsible for solving these in practice are not interested for their optimal solutions, but they are satisfied with an approximate optimal solutions obtained in real-time as well. Consequently, a lot of heuristic algorithms have been emerged that are able to obtain the non-optimal solutions, but well enough for practical applications.

The stochastic nature-inspired population-based algorithms are heuristic methods that can be applied to problem domains, where no domain-specific knowledge has yet been discovered. In this study, we focus on searching for an optimal parameter tuning of PI-controller with EA and SI-based algorithms. Precisely, a comparative study of the bat algorithm (BA) and genetic algorithm (GA) for solving this problem has been performed, where the former belongs to a class of SI-based algorithms, while the latter is the member of EA-family.

In the remainder of the paper, characteristics of both algorithms are illustrated in details.

3.1 Bat algorithm

As already mentioned, BA is one of the newest representatives of SI-based algorithms. Since 2010, its reputation and visibility are highly rising. Bat algorithm is easy to implement and applicable to various applications. It offers solid results by solving of the low-dimensional problems and that is one of the reasons to be applied to tune the parameters of PI-controller. Thus, high convergence of the BA algorithm is expected.

3.1.1 Fundamentals of Bat algorithm

Bats are night animals. Nature has given them ability to navigate in darkness, using a so-called sonar, named $echolocation$. This phenomenon consists of generating an ultrasonic pulse, which echoes from obstacles and prey, bouncing back to the bat, who calculates the distance to either obstacle or prey. More information on bats behavior and their abilities can be found in [14].

The BA algorithm treats bats as a swarm of bats, searching for a prey. Since bats search for the prey individually, BA emphasizes the phenomena of echolocation by converging the whole swarm by approaching the found prey. This means that one random individual can achieve the whole swarm to divert for food. From the engineer's point of view, more food means higher fitness function and better solution of the problem. The whole swarm is converging to the best solution during generations by changing their current positions.

3.1.2 Model of Bat algorithm

The moving of bats, their attitude and acting in a swarm presented in previous section can be modelled using simple mathematical equations. The whole modelling process is described in [23], so only main results are shown here. At first, three different variables should be defined describing bat moving as follows:

- frequency of pulse $Q_i^{(t)}$,
- velocity $\mathbf{v}_i^{(t)}$ and
- position $\mathbf{x}_i^{(t+1)}$,

where $Q_i^{(t)}$ represents actual pulse frequency, $\mathbf{v}_i^{(t)}$ velocity of an individual bat and $\mathbf{x}_i^{(t+1)}$ position of the i-th bat at generation t.

The flight of a bat can be summarized in Eqs. 14-16:

$$Q_i^{(t)} = Q_{min}^{(t)} + (Q_{max}^{(t)} - Q_{min}^{(t)}) \cdot \beta, \tag{14}$$

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t)} + \left[\mathbf{x}_i^{(t)} - \mathbf{x}_{best}^{(t)}\right] \cdot Q_i, \tag{15}$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t)}. \tag{16}$$

Output pulse frequency can vary in the interval $Q_i^{(t)} \in [Q_{min}, Q_{max}]$. The random number $\beta \in [0, 1]$ specifies the output pulse and $\mathbf{x}_{best}^{(t)}$ presents the current best solution.

The BA search process consists of two components, i.e., exploration and exploitation. Exploration means a discovering of the new solutions, while the exploitation directs the search in the neighborhood of the existing solutions. Both processes cannot be run simultaneously because they typically depend on the variation operators, while balancing between exploration and exploitation performs a control parameter setting. There are more optimal parameter settings.

In the BA, the exploration and exploitation components of the search process are balanced by using two exploration strategies and parameter $r_i^{(t)}$. The first exploration strategy expressed by Eq. 16 is more explorative in its nature, while the second strategy expressed as

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \epsilon \cdot \bar{A}^{(t)}, \tag{17}$$

implements the random walk, i.e., a kind of the local search that is more focused on the exploitation of the current best solution. Let us notice that $\mathbf{x}_{new}$ in the equation presents new best solution, if applicable, and $\mathbf{x}_{old}$ presents current best solution. ϵ is the random number in range (-1,1) and $\bar{A}^{(t)}$ is the average loudness.

The last strategy is applied according to the pulse rate $r_i^{(t)}$.The pulse rate is normally being changed during generations, where simulates nature behavior of bats outputting loud pulses with low pulse rate when searching for preys and outputting silent pulses with high pulse rate when approaching to the prey.

3.1.3 Pseudocode of Bat algorithm

A pseudo-code of the BA algorithm is illustrated in Algorithm 1. This algorithm consists of the following elements:

- initialization of bat population (function 'init_bat' in line 1),
- generation of new solution according to Eq. 16 (function 'generate_new_solution' in line 6),
- the local search step according to Eq. 17 and parameter $r_i^{(t)}$ (function 'improve_the_best_solution' in lines 7-9),
- evaluation of the new solution (function 'evaluate_the_new_solution' in line 10),
- save the best solution conditionally (in lines 12-15),
- find the best solution (in line 15).

BA is a population algorithm, what means that a population size and maximal number of generations should be pre-defined. During the optimizational process, the new position is being calculated for every bat and generation value is incremented. The execution of algorithm stops when maximal number of generations are reached.

Algorithm 1 Original Bat algorithm.

Input: Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, MAX_FE.
Output: The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{max} = \max(f(\mathbf{x}))$.

```
1: init_bat();
2: eval = evaluate_the_new_population;
3: f_max = find_the_best_solution(x_best);
4: while termination_condition_not_met do
5:    for i = 1 to Np do
6:       y = generate_new_solution(x_i);
7:       if rand(0,1) > r_i then
8:          y = improve_the_best_solution(x_best)
9:       end if{local search step}
10:      f_new = evaluate_the_new_solution(y);
11:      eval = eval + 1;
12:      if f_new ≥ f_i and N(0,1) < A_i then
13:         x_i = y; f_i = f_new;
14:      end if{save the best solution conditionally}
15:      f_max=find_the_best_solution(x_best);
16:   end for
17: end while
```

3.2 Genetic algorithm

As already mentioned, GA was one of the first optimization algorithms, belonging to a family of EA [1]. Although GA is similar as BA a population-based algorithm, it differs in comparison to BA a lot.

3.2.1 Fundamentals of Genetic algorithm

GA searches for the global optimum using unique genetic operators. There are three common operators, i.e., a selection, a crossover and a mutation. The parent's selection is a part of algorithm, where so called parent genomes (solution of the problem) are chosen to enter into crossover procedure. The two common parent selections are roulette-wheel selection and tournament selection [12]. Crossover is the most important part of the algorithm, since it enables the two parents to vary their genetic material in order to improve the existing solutions. The last operator, mutation could be represented as a rescue method, which prevents algorithm from trapping in local optima, which usually stops improving process. A task of the survivor selection is to determine the better solutions for surviving and transferring their good characteristics in the next generations.

Nevertheless, the basic GA consists of six elements:

- initialization of individuals,
- parent's selection,
- crossover,
- mutation,
- evaluation and
- survivor's selection.

Initialization is a process, where random (starting) population is generated and evaluation is process where the fitness value determining success of each individual is calculated. The optimization runs until the number of generations has not reached its maximum specified value, or its optimal value was found or enough quality solution was discovered. The most important issue by using the EA is the representation of individuals. Although the original GA uses binary representation, where solutions are represented as binary strings, the real-coded GA is becoming more important today. This type of GA is applied also in this study.

3.2.2 Pseudocode of Genetic algorithm

The pseudo-code of the GA is presented in Algorithm 2, from which it can be seen the following elements of the GA:

- initialization of initial population (function 'init_population_with_random_candidate_solutions' in line 1),
- parent's selection (function 'select_parents' in line 4),
- crossover operator (function 'recombine_pairs_of_parents' in line 5),
- mutation operator (function 'mutate_the_resulting_offspring' in line 6),
- evaluation function (function 'evaluate_new_candidates' in line 7),

– survivor's selection (function 'select_individuals_for_the_next_generation' in line 8).

The main evolutionary cycle is performed until the 'termination_condition_not_met' is not met. Each solution is represented as real-coded vector $\mathbf{x}_i^{(t)} = \{x_{i,j}^{(t)}\}$, where $i = 1, \ldots, NP \wedge j = 1, \ldots, D$, NP is a population and D a dimension of the problem.

Algorithm 2 Evolutionary algorithm.

```
1: init_population_with_random_candidate_solutions;
2: eval = evaluate_each_candidate;
3: while termination_condition_not_met do
4:    select_parents;
5:    recombine_pairs_of_parents;
6:    mutate_the_resulting_offspring;
7:    eval += evaluate_new_candidates;
8:    select_individuals_for_the_next_generation;
9: end while
```

3.3 Parameter tuning of PI-controller with nature-inspired algorithms

In order to solve parameter tuning of PI-controller, the nature-inspired algorithms need to be modified properly. As we know, the results of our optimization problem depend on the input vector $\mathbf{x}_i$ according to Eq. 11. This means that only a representation of solution for the nature-inspired algorithms must be changed in order to adapt them for solving the parameter setting of PI-controller. In other words, the solution in these algorithms is represented as follows

$$\mathbf{x}_i^{(t)} = \{x_{i,j}\}, \text{ for } i = 1, \ldots, NP \wedge j = 1, \ldots, D, \qquad (18)$$

where $x_{i,1} = q_{0,1}$, $x_{i,2} = q_{1,1}$, $x_{i,3} = q_{0,2}$ and $x_{i,4} = q_{1,2}$, and NP denotes a population size and D is a dimension of the problem. All the other elements of the original nature-inspired algorithms do not demand any modifications.

4 Results

The goal of our experimental work was to show that the stochastic nature-inspired algorithms can successfully be applied for searching the optimal parameters of PI-controller that controls the robotic arm SCARA. In line with this, two population-based algorithms were developed and customized to this problem, i.e., BA and GA. The development was performed on a personal computer (PC) with installed Windows operating system and MATLAB/Simulink in language C/C++. The algorithms were loaded onto DSP2-Roby interface card, where these were also executed, while the results were received from the card via USB connection to the PC and displayed in MATLAB/Simulink.

The algorithm's parameters as presented in Table 1 were used during the simulation.

Parameter	Setting
NP	10
n_{gen}	10
Q	[0.5,1.5]
β	[0,1]
r_i	0.1
A_i	0.9

(a) BA

Parameter	Setting
NP	10
n_{gen}	10
p_c	0.8
p_m	0.01
Par.sel.	Tour.$m = 2$
Sur.sel.	Fittest

(b) GA

Table 1: Parameter setting by nature-inspired algorithms.

Let us notice that both algorithms used the same number of fitness function evaluations, i.e., $MAX\_FE = 10 \times 10 = 100$, where $MAX\_FE = NP \times n_{gen}$. This number of fitness function evaluations is relatively low, but this setting enables algorithms to run in the real-time. In order to limit the search space rationally, lower and upper bounds as presented in Table 2 were considered during the optimization for both algorithms.

Parameter	Bounds	
	Lower	Upper
$q_{0,1}$	0	400
$q_{1,1}$	0	40
$q_{0,2}$	0	400
$q_{1,2}$	0	40

Table 2: Limited values of parameters.

In the table, parameters $q_{0,1}$ and $q_{1,1}$ denote limited parameter values for axis 1, while $q_{0,2}$ and $q_{1,2}$ limited values for axis 2.

The results of the optimization using algorithms BA and GA are illustrated in Table 3, from which it can be observed that 10 independent runs were conducted for each of the algorithm in test. This is normally, when dealing with stochastic algorithms, where we are usually not interested for the best solution, but rather the average results of the optimization. However, in our case, we made an exception here and considered the best solutions. The obtained results for each of two axis are presented together with the corresponding average values and the average values according to all runs are presented. The best results are marked in the table as bold.

As can be seen from the table, the best result of the BA was obtained in seventh run, while the GA was the best in the third run. When comparing the results of both algorithms with each other, it can be observed that the result achieved by GA was slightly better than this achieved by the BA. However, when these results were estimated statistically using the Wilxocon signed rank non-parametric test with confidence $\alpha = 0.05$, it turned out that the BA outperformed the results of GA significantly (p-value$= 0.03 < 0.05$).

Run	BA			GA		
	Axis-1	Axis-2	Average	Axis-1	Axis-2	Average
1	0.9729	0.9726	0.9727	0.9804	0.9600	0.9702
2	0.9795	0.9497	0.9646	0.9729	0.9556	0.9642
3	0.9702	0.9847	0.9775	0.9726	0.9845	**0.9786**
4	0.9757	0.9802	0.9780	0.9455	0.9653	0.9554
5	0.9778	0.9592	0.9685	0.9638	0.9443	0.9540
6	0.9724	0.9790	0.9757	0.9782	0.9640	0.9711
7	0.9746	0.9810	**0.9778**	0.9752	0.9727	0.9740
8	0.9647	0.9887	0.9767	0.9774	0.9409	0.9592
9	0.9787	0.9733	0.9760	0.9721	0.9569	0.9645
10	0.9748	0.9430	0.9589	0.9662	0.9568	0.9615
Average	0.9741	0.9712	0.9726	0.9704	0.9601	0.9653

Table 3: Comparing the results of BA and GA.

The corresponding best results as obtained by BA and GA algorithms are presented in Table 4.

Alg.	Axis-1		Axis-2		Eff.
	$q_{0,1}$	$q_{1,1}$	$q_{0,2}$	$q_{1,2}$	
BA	257.064	18.9201	163.155	15.1531	0.9778
GA	56.4213	3.27043	108.571	7.68699	0.9786

Table 4: The best results obtained by BA and GA.

As can be observed from the table, there is not only one optimal solution, because both sets of input parameters were very different when compared between each other. However, it seems that more important is a relation between both pairs of input parameters.

In order to show how the optimal values are changed during the optimization, the convergence graphs in Fig. 4 are drawn that depict a changing of the best solution during one run according to increasing of generations for each of the observed algorithm.

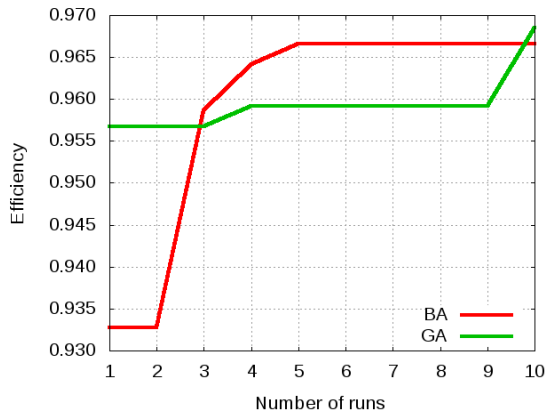

Figure 4: Convergence graph by BA and GA.

As can be seen from the graph, the BA showed a rapid convergence to the optimal value, which is already found in fifth generation. After this generation the algorithm shows signs of stagnation. On the other hand, the GA converge to the optimal value slower. Therefore, it can improve the results also in later generations.

5 Conclusion

The aim of our experimental work was to show that stochastic nature-inspired population-based algorithms can successfully be applied to tuning parameters of PI-controller of the robot arm SCARA. Two nature inspired algorithms were taken into consideration, i.e., the BA and GA algorithms. The former is simple and easy to implement and because of its rapid convergence interesting to use in robotics.

The results of experiments showed that both the algorithms can be used for optimal tuning parameters of PI-controller. Although the BA algorithm significantly outperformed the results of GA according to the best obtained results for each of two axis in ten runs, it turned out that the GA converges slower than the BA. This means that the GA demands the larger population as well as higher number of generations. On the other hand, increasing the population size and/or the maximum number of generations causes increasing the optimization time that can dramatically affect the real-time response of the system. In this context, the BA algorithm is more appropriate for this optimization as GA.

In the future, the BA algorithm could be hybridized with differential evolution mutation strategies [9] and thus the results of optimization would be improved. An adaptation of control parameters represents additional method, where these are encoded into representation of solutions and undergo acting the variation operators [7].

References

[1] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] C. Darwin. R.(1859): On the origin of species by means of natural selection. *Murray. London*, 1871.

[3] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[4] W. R. Evans. Control system synthesis by root locus method. *American Institute of Electrical Engineers, Transactions of the*, 69(1):66–69, 1950.

[5] I. Fister, S. Fong, J. Brest, and I. Fister. A novel hybrid self-adaptive bat algorithm. *The Scientific World Journal*, 2014, 2014.

[6] I. Fister, S. Fong, J. Brest, and I. Fister. Towards the self-adaptation in the bat algorithm. In *Proceedings of the 13th IASTED international conference on artificial intelligence and applications*, 2014.

[7] I. Fister, D. Strnad, X.-S. Yang, and I. Fister Jr. Adaptation and hybridization in nature-inspired algorithms. In *Adaptation and Hybridization in Computational Intelligence*, pages 3–50. Springer, 2015.

[8] I. Fister Jr, D. Fister, and I. Fister. Differential evolution strategies with random forest regression in the bat algorithm. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1703–1706. ACM, 2013.

[9] I. Fister Jr, D. Fister, and X.-S. Yang. A hybrid bat algorithm. *Elektrotehniški vestnik*, 2013.

[10] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari. Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, 22(6):1239–1255, 2013.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[12] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

[13] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[14] D. R. Griffin. Listening in the dark: the acoustic orientation of bats and men. 1958.

[15] A. Jagarinec. *Adaptivni regulator z mehko logiko za dvoosni SCARA mehanizem*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[16] L. H. Keel and S. P. Bhattacharyya. A bode plot characterization of all stabilizing controllers. *Automatic Control, IEEE Transactions on*, 55(11):2650–2654, 2010.

[17] M. Kolar. *Vodenje SCARA robota z mehko logiko*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[18] H. Makino, N. Furuya, K. Soma, and E. Chin. Research and development of the scara robot. In *Proceedings of the 4th International Conference on Production Engineering*, pages 885–890, 1980.

[19] T. Slanič. *Genetski regulator za dvoosnega SCARA robota*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2006.

[20] A. M. Turing. Intelligent machinery, a heretical theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, page 105, 1948.

[21] J. Čas. *Izdelava zveznega nevronskega sliding-mode regulatorja za teleoperiranje SCARA robota*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2006.

[22] M. Čurkovič. *Vgrajeni sistemi DSP/FPGA v sistemih vodenja*. Magistrsko delo Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2010.

[23] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[24] X.-S. Yang. Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3(5):267–274, 2011.