Applied Soft Computing Journal 89 (2020) 106074

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Chaos-based Vortex Search algorithm for solving inverse kinematics problem of serial robot manipulators with offset wrist

Metin Toz

Computer Engineering Department of Technology Faculty, Düzce University, Turkey

ARTICLE INFO

Article history:
Received 22 October 2018
Received in revised form 12 December 2019
Accepted 6 January 2020
Available online 13 January 2020

Keywords:
Vortex search
Chaos map
Serial robot manipulator with offset wrist
Inverse kinematics problem

ABSTRACT

Vortex Search (VS) algorithm is a single-solution-based optimization algorithm that requires the high maximum number of iterations (NOI) to solve optimization problems. In this study, two methods were proposed to reduce the required maximum NOI of the VS algorithm. These methods are based on using ten chaos maps with the VS algorithm and provide improvements in the exploration and exploitation abilities of the algorithm for reducing the required maximum NOI. Ten chaos-based VS algorithms (CVSs) were obtained by combining these methods with the VS algorithm. The performances of the CVS algorithms were tested by fifty benchmark functions. The results were evaluated in terms of some statistical values and a pairwise statistical test, Wilcoxon Signed-Rank Test. According to the results, it was found that the CVS algorithm obtained by using the Gauss–Mouse chaos map was the best algorithm. And also, it was shown that the proposed CVS algorithm performs better than the classical VS algorithm, even when its maximum NOI was ten times less than the maximum NOI of the VS algorithm. Additionally, the effects of the proposed methods in the exploration and the exploitation abilities of the VS algorithm were visually shown and a comparison about algorithm processing time was presented. In order to test the performance of the proposed CVS algorithm in solving the real-world optimization problems, the inverse kinematics problem of a six Degrees Of Freedom (DOF) serial robot manipulator with offset wrist was solved with both the proposed CVS algorithm and the VS algorithm for two different types of trajectories. The results showed that the proposed algorithm outperforms the VS algorithm in terms of the objective function values and position errors of the end-effector of the serial robot manipulator.

1. Introduction

Complexity, difficulty, diversity and time dependency in real-world problems are increasing day by day in parallel to the rapid advances and changes in scientific disciplines. Especially, multi-disciplinary optimization problems are getting to be more and more complex and difficult to be solved. Robot mechanism design optimization problem and real-time control optimization problem can be given as examples of such problems. Meta-heuristic optimization algorithms are frequently used for solving these types of real-world problems. These algorithms are preferred to find acceptable solutions to the problems in a reasonable time instead of generating optimal solutions independently from time. However, although there are many meta-heuristic algorithms that can be used to solve any type of optimization problems, according to the NFL theorem [1], there is no algorithm that finds the best solution for all the optimization problems. Therefore, in the literature, several meta-heuristic optimization algorithms have been proposed for different kinds of optimization problems. These algorithms can be grouped into different categories such as physics-based algorithms [2,3], evolution-based algorithms [4–6] and swarm-based algorithms [7–9] or in a different manner, single-solution-based algorithms [2,10–12] and population-based algorithms [7–9]. Single-solution-based optimization algorithms search the optimum solution by improving a single solution during their search process [13]. Improving the current solution is performed by iteratively moving from the current solution to another one in the search space of the problem according to the search strategy of the algorithm. Some examples of these algorithms can be mentioned as; Simulated Annealing [2] which mimics the annealing technique of the materials used by the metallurgists. Tabu Search [10], that is inspired from the mechanisms used by the human memory and is based on the history of the search [13]. And, Variable Neighborhood Search [11] that uses a dynamically changing neighborhoods strategy for its search process. On the other side, the population-based algorithms perform the search behavior by using a population that includes a number of candidate solutions. These algorithms, different from the single-solution-based algorithms, move all the members of the population towards the optimum solution. Thus, such an

E-mail address: metintoz@duzce.edu.tr.

https://doi.org/10.1016/j.asoc.2020.106074
1568-4946/

algorithm needs to follow the movements of all the members of the population through the iterations and select the member that represents the best solution as the optimum at the end of the iterations. In the literature, there are several population based algorithms. Particle Swarm Optimization (PSO) [7] that mimics the search behaviors of the creatures from the nature such as flock of birds or school of fish is one of the most-known population based algorithms. PSO has a simple formulation and perform a fast convergence characteristic to the optimum point. Therefore it has been used to solve nearly all of the types of the optimization problems in the literature. Some of the other algorithms that are inspired from the nature can be mentioned as Artificial Bee Colony (ABC) algorithm [14], Ant Colony Algorithm (ACO) [9], Gray Wolf Optimizer (GWO) [8], Whale Optimization Algorithm (WOA) [15] and Firefly Algorithm (FA) [16]. Genetic Algorithms (GA) [4] is one of the other well-known population based algorithms to solve the optimization problems. It is based on the evolution of natural genetics and uses the crossover and mutation operators to obtain the next generation of the population from the current one. Biogeography-Based Optimization [6] and Genetic Programming [5] can be given as two other examples of the evaluation based algorithms. There are many other optimization algorithms in the literature that we cannot express here because of the limited space. Although the optimization algorithms differ from each other in terms of several aspects, there are two main searching characteristics to all of them have in common. The first one is the exploration that shows how efficiently the algorithm searches the entire search space and the second is the exploitation that indicates the ability of the algorithm to find a point near the optimum [15]. In an efficient optimization algorithm, these two search behaviors, exploration and exploitation, should be balanced [17]. From this point of view, there is a basic difference between the population-based and the single-solution-based algorithms. Population-based algorithms generally show an exploration oriented search while the single solution-based-algorithms have a search characteristic that is based on the exploitation [17]. And, although the both type of these algorithms have been efficiently used to solve many kinds of optimization problems, their performance can be improved by changing their search characteristics [18]. Such improvements have been performed in the literature by using different techniques such as modifying the search operators of the algorithms, hybridization of two or more optimization algorithms and/or techniques or embed a natural/scientific fact in the algorithm such as chaos theory [19].

Vortex Search (VS) optimization algorithm is a new single solution optimization algorithm proposed by Doğan and Ölmez [12] to solve numerical function optimization. VS is based on the behavior of the stirred fluids and it formulates the vortex pattern composed by the flow of the fluids. Although VS is a relatively new optimization algorithm, it has been used to solve many optimization problems in the literature. Ozkaya and Seyfi [20], used VS for optimum design of leaf-shaped microstrip patch antenna's. They used four different optimization algorithms, PSO, Differential Evolution (DE), GWO and VS with artificial neural networks and compared the performance of the optimization algorithms. They concluded that the VS was the best algorithm among the others for optimum antenna design. Garcia et al. [21], proposed to use the VS algorithm for estimation of optimum thermal conductivity and the thermal diffusivity properties within a solid parallelepiped during a microwave heating process. They compared the performance of the VS algorithm with Spiral Optimization Algorithm (SOA), and the Weighted Attraction Method (WAM) and mentioned that according to their results the VS outperformed the other algorithms. Doğan and Ölmez [22] proposed using VS for solving optimum selection problem of the passive components in analog active filter design problem and compared the VS algorithm with PSO, ABC, Differential Evolution (DE) and Harmony Search (HS) in terms of the objective function values. They showed that the VS algorithm obtained the best objective function values. Ali et al. [23] solved the optimization problem of a single mixed refrigerant natural gas liquefaction process by using the VS algorithm and found that it significantly reduced the required energy for the process and improved the coefficient of performance in comparison with the base case. They also compared the VS algorithm with GA, PSO and Knowledge-Based Optimization (KBO), and concluded that it showed superior performance over the other algorithms. Some of the other studies that used the VS algorithm to solve optimization problems can be found in [24–27]. Although the VS algorithm has high computational precision and fast convergent speed, sometimes it may stick to local minima or requires the large maximum NOI for some optimization problems [28]. Therefore, in the literature, some improved versions of the VS algorithm have also been proposed. Li et al. [28] proposed to generate candidate solutions by using the logic self-map chaos instead of using Gaussian distribution and to use Lévy flight strategy for obtaining the centers of the circles in VS algorithm in order to increase local and global search abilities of the algorithm. They named their algorithm as I-VS algorithm and solved the boiler combustion optimization problem and compared their algorithm with PSO, Krill Herd (KH), VS and Gravity Search Algorithm (GSA). They concluded that their algorithm showed the best performance. Huang et al. [29] improved the local and global search abilities of the VS algorithm (VS-G) by using a gradient-based approximation technique for updating the centers of the circles defined in the search process of the VS algorithm. They solved the optimization problem of the fore part of the KCS container ship as a real-world problem and also shown that their algorithm outperforms the original form of the VS algorithm in solving thirty benchmark functions. Sajedi et al. [30] proposed a multiple VS algorithm (MVSA) which uses a number of vortices in parallel to increase the ability to escape from local minima. They tested their algorithm on eleven benchmark function and compared it with GA, SA, ABS, and VS. As observed from the above studies, the VS algorithm has a powerful search technique, basic formulations, and an open structure to development; on the other hand, it may require the large maximum NOI to solve the real-world optimization problems. Therefore, in this study, we choose the VS algorithm in order to make it be more powerful in solving real-world optimization problems. To this end, we aimed to decrease the required maximum NOI for the VS algorithm by increasing its searching abilities in both its exploration and exploitation stages.

Chaos is the "randomness" that generated by the simple deterministic systems [31]. And, a chaotic system may be considered as a source of the randomness [32]. Therefore, as the randomness is one of the most important mechanisms of the search processes of the meta-heuristics, in the literature chaos has been used to improve the performance of these algorithms. Wang et al. [33], proposed a chaos-based version of the KH optimization algorithm. They used twelve chaotic maps to tune two inertia weight parameters of the algorithm and found that the singer map was the best choice to improve the performance of the KH. They tested their algorithm on fourteen benchmark functions and compared its performance with nine other algorithms. Tang et al. [34] used chaos theory to solve the premature convergence problem of the Dynamic Group Optimization (DGO) algorithm, they used ten chaotic maps and found that the circle map was the best chaotic map for DGO algorithm. They also test the performance of their algorithm in solving several benchmark functions and the clustering problem of multimedia data set as a real-world problem. Saha and Mukherjee [35] integrated a

chaotic local search technique with Symbiotic Organisms Search algorithm (SOS) and tested the proposed method on twenty-six unconstrained benchmark functions and on a real-world power system optimization problem. They showed that the proposed chaos based SOS algorithm outperforms the classical form of the SOS form in terms of solving both the benchmark functions and the real-world optimization problem. Metlicka and Davendra [36] proposed a chaos-based discrete artificial bee colony algorithm (CDABC). They used four chaos maps as chaos pseudo-random number generators for the algorithm and solved two optimization problems, quadratic assignment and capacitated vehicle routing problems. They concluded that the Tinkerbell chaos map based variant of the algorithm showed the best performance for solving both of the problems. Nie et al. [37], proposed an adaptive chaos based PSO algorithm for optimization of the parameters of the Proportional-Integral-Derivative controller. Some of the other studies can be found in [38–40]. As seen from these researches, generally chaos theory can help to improve the performance of the optimization algorithms by changing their random selection strategies. Thus, in this study, we aimed to use chaos theory to change the search characteristics of the VS algorithm and improve its performance.

Serial robot manipulators are very popular mechanisms both in the industry and the literature. These manipulators are constructed by joining rigid bodies with revolute or prismatic joints, successively. Wrist structures of these mechanisms can be designed for different Degrees of Freedom (DOF) in different types according to their usage purposes [41]. Generally, the manipulator's wrists are designed in two different types, one is the Euler wrist which composed of three successive joints and their axes intersect at a common point, and the other is the offset wrist which constructed by three joints that their axes do not intersect at a common point [42]. Offset wrists are used with the robot manipulators that needed to get long horizontal reach while maintaining the appropriate angle in their workspaces while Euler wrists provide regularly shaped workspaces [41]. Inverse kinematics is defined as finding the joint parameters by using kinematics equations of the robot for a pre-determined end-effector position. The solution to this problem is very important for the serial robot manipulators since it is the basic component of the many analyses for these manipulators such as workspace determination and dynamic analysis. Solving inverse kinematics problem can be very difficult for some types of serial robot manipulators, especially for the robot manipulators that have offset wrist.

As many real-world problems, the inverse kinematics problem of the serial robots is also time-dependent and it should be solved in an acceptable time. Thus, in this study, we aimed to solve the inverse kinematics problem of a six DOF serial robot manipulator with offset wrist in a reasonable time using an improved version of the VS algorithm by chaos. We proposed to change two strategies of the VS algorithm. The first one is the updating strategy of the current solution and the second is the decreasing strategy of the radius of the circle that is used as the border of the area that neighbors of the current solution are located in. To do so, we preferred to add chaos effect to these strategies and proposed ten chaos based VS algorithms using ten chaos maps. Then, we presented a comparison between the performance of these ten CVS algorithms using fifty benchmark functions that also used for testing the performance of the VS algorithm by Doğan and Ölmez [12]. According to the results of solving the benchmark functions, we determined the best CVS algorithm. Then, the determined best CVS algorithm was used to visually show "how the proposed methods affect the performance of the VS algorithm". After that, a comparison between the algorithm processing times of the proposed CVS and the VS algorithms was presented for solving the fifty benchmark functions. And, finally, we solved the inverse kinematics problem of a six DOF serial robot manipulator with offset wrist by the best CVS algorithm and compared its performance with the classical VS algorithm.

In the second section firstly the formulation of the VS algorithm was given. Then the two strategies proposed in this study were explained in detail and the ten chaos based VS algorithms are obtained. In the third section, the performances of the ten CVS algorithms were compared and the best CVS algorithm was determined. The solution of the inverse kinematics problem of the six DOF serial robot manipulator with offset wrist was given in the fourth section, a discussion about evaluating the results of the proposed methods was performed in the fifth section and finally, the study is concluded by the last section.

2. Improvement of the performance of the VS algorithm by using chaos maps

In this section, we first presented the formulation of the classical VS algorithm and then explained the two strategies that proposed to improve the performance of the VS algorithm. Finally, we obtained the ten chaos based VS algorithms.

2.1. Vortex Search algorithm

Vortex Search algorithm is a new single-solution based optimization algorithm developed by Doğan and Ölmez [12]. Because of its single-solution based structure, for optimal solution convergence, at each one of the iterations, new candidate solutions are randomly determined around the current best solution, and then the next best solution is selected from these solutions and the process continues until the stopping criterion is provided. The formulation of the VS algorithm can be summarized for a two-dimensional optimization problem as follows;

(1) The initial best solution is determined as the midpoint of the search space. This point can be considered as the center of a circle for a two-dimensional problem. In the VS algorithm, this point is constantly updated so that to allow for the formation of nested circles [12].

$$\mu_0 = \frac{ub - lb}{2} \tag{1}$$

where μ_0 is the initial best solution and ub and lb are minimum and maximum limits of the search space, respectively.

(2) After the initial best solution is determined, the iteration process of the algorithm starts and the best solution is updated at each one of the iterations. In order to determine the best solution at each one of the iterations, some candidate solutions are defined in the neighborhood of the current best solution [12].

$$C_t(s) = (s_1, s_2, s_3, \ldots, s_k) \tag{2}$$

where, t = 1, 2, 3, …, m is the iteration number, $C_t(s) = (s_1, s_2, s_3, \ldots, s_k)$ represent the set of the candidate solutions at tth iteration and k is the number of the candidate solutions. In the VS algorithm, the candidate solutions are determined by a Gaussian distribution formula [12].

$$p\left(x\middle|\mu, \sum\right) = \frac{1}{\sqrt{(2\pi)^d \left|\sum\right|}} exp\left\{-\frac{1}{2}(x-\mu)^T \sum^{-1} (x-\mu)\right\} \tag{3}$$

where, $\sum$, d, x and μ are covariance matrix, dimension, $d \times 1$ dimensional vectors consist of random numbers and

the sample mean(center), respectively. In the VS algorithm, $\sum$ is defined as follows in order to obtain a spherical Gaussian distribution;

$$\sum = \sigma^2 I_{d\times d} \tag{4}$$

where σ^2 is the variance of the distribution and I is the identity matrix. σ value is used to determine the border of the search area for the neighbors of the current best solution, this value is the radius of a circle for a two-dimensional problem. The first value of this parameter (σ_0) is determined as follows [12];

$$\sigma_0 = \frac{\max(ub) - \min(lb)}{2} \tag{5}$$

(3) In this step the fitness values of the candidate solutions are determined by using the objective function defined for the problem and if there is a better solution than the current best solution among the candidate solutions, the current best solution is updated.

(4) The radius of the search area for the candidate solutions is iteratively decreased by the following incomplete gamma function in order to approach the best solution [12].

$$\gamma(x, a) = \int_0^x e^{-t} t^{a-1} dt \quad a > 0 \tag{6}$$

where a is shape parameter and $x \geq 0$ is a random number in the interval [0–1]. In the VS algorithm, radius values at the first and tth iterations are calculated by using the inverse of the incomplete gamma function and σ_0, as given in Eqs. (7) and (8) [12].

$$r_0 = \sigma_0 \left(\frac{1}{x}\right) gammaincinv(x, a_0) \tag{7}$$

$$r_t = \sigma_0 \left(\frac{1}{x}\right) gammaincinv(x, a_t) \tag{8}$$

where $gammaincinv$ is the inverse of the incomplete gamma function and a_t is calculated as follows [12];

$$a_t = a_0 - \frac{t}{Maxt} \tag{9}$$

where $Maxt$ is the maximum NOI. In order to ensure to cover all the search space at the first iteration, a_0 value is determined as 1. Therefore, a_t is iteratively decreased from 1 to 0.

(5) In the VS algorithm, the steps between 2 and 4 repeat until the stopping criterion is provided.

2.2. Chaos based VS algorithms

If we look at the VS algorithm in terms of exploitation and exploration, which are two basic features of an optimization algorithm, it is seen that the balance between these two features is provided by randomly selecting of candidate solutions using Gaussian distribution and by iteratively decreasing the radius of the circle of the search area. Two fundamental problems arise in the search process when using these two methods. These can be expressed as;

(1) The resolution of the search space is controlled by the radius reduction method. That is, the slower the radius reduction, the greater the search resolution in the search space [12]. However, in the current algorithm this can only be achieved with a maximum NOI. Therefore, a higher maximum NOI is needed for higher search resolution. For example, Doğan and Ölmez [12] have set the number of iterations to 100.000 in some of their experimental work. However, although 100.000 iterations seem to be a reasonable number to solve for numerical benchmark functions, this number is a very high iteration number for the real-world engineering problems requiring many intermediate operations, such as inverse kinematics problem of the serial robot mechanisms. Thus, the solution time of the real world problems may not be provided in an acceptable time interval with this form of the VS algorithm.

(2) Another disadvantage of the VS algorithm is in the process of choosing the new best solution at each one of the iterations. In the current algorithm, the best solution remains the same until a new best solution is found. This increases the likelihood that new candidate solutions will always be searched around the same center and thus, the possibility of stuck to the local optimal solution.

In this study, we proposed two methods based on using ten chaos maps to solve these two basic problems of the VS algorithm. The chaos maps used in this study were given in Table 1 [43].

2.3. Proposed methods for improving the performance of the VS algorithm

As explained in the previous section, there are two basic problems with the search process of the classical form of the VS algorithm. In this section, we propose two methods to improve the searching performance of the VS algorithm by adding chaos effect to the algorithm formulations.

Proposed method 1: In order to increase the exploration ability of the algorithm, in this study, we propose to use chaos maps so as to add the chaos effect at the decreasing process of the r_t value instead of increasing the maximum NOI. In this way, the change on r_t value occurs both in positive and negative directions according to the used chaos map while it is maintaining its tend to decrease iteratively. Therefore, instead of increasing the resolution of the search space, the sample points of the search space defined according to the current resolution are changed at each one of the iterations with the help of the chaos map. Thus, although the number of the points accessed in that resolution remains the same, the search space is better scanned than the classical approach. The proposed approach was realized with the help of the normalization procedure given in the study [43]. In this approach, the original values of the r_t values that found according to Eq. (8) is added with the normalized chaos map value. As an example, let the $Maxt = 10\,000$, $x = 0.1$ and $a_0 = 1$. With these values, the graphic of the r_t value will be as given in Fig. 1.

From the figure, it can be seen that the value of the r_t variable is rapidly decreasing at the first half of the maximum NOI to provide the exploration ability to the algorithm while at the second half it is decreasing at very small rates in order to increase the local search ability of the algorithm. In this study, we propose to apply the chaos effect to this decreasing procedure of r_t value while keeping its original form in order to increase the exploration ability of the algorithm. In this approach r_t value was calculated as given in Eq. (10);

$$\begin{aligned} &if\ r < 0.5\ then \quad r_t = \left(\frac{\max(ub) - \min(lb)}{2} + abs\left(f_{b_{ratio}}\right) C_m(t)\right) \\ &\times \left(\frac{1}{x}\right) gammaincinv(x, a_t) \\ &else\ r_t = \left(\frac{\max(ub) - \min(lb)}{2} - abs\left(f_{b_{ratio}}\right) C_m(t)\right) \\ &\times \left(\frac{1}{x}\right) gammaincinv(x, a_t) \end{aligned}$$

Table 1
Chaos maps and their definitions (all the chaos maps start from 0.7 ($x_0 = 0.7$)) [43].

Chaotic Maps		*Definition*	*Interval*
1	Chebyshev	$x_{i+1} = \cos\left(i\cos^{-1}(x_i)\right)$	$(-1,1)$
2	Circle	$x_{i+1} = mod\left(x_i + b - \left(\frac{a}{2\pi}\right)\sin 2\pi x_i, 1\right),\ a = 0.5\ b = 0.2$	(0, 1)
3	Gaus–Mouse	$x_{i+1}\begin{cases} x_i = 0 \\ \frac{1}{mod(x_i, 1)} \quad otherwise \end{cases}$	(0, 1)
4	Iterative	$x_{i+1} = \sin\frac{a\pi}{x_i} \quad a = 0.7$	$(-1,1)$
5	Logistic	$x_{i+1} = ax_i(1 - x_i) \quad a = 4$	(0, 1)
6	Piecewise	$x_{i+1} = \begin{cases} \frac{x_i}{P} & 0 \le x_i < P \\ \frac{x_i - P}{0.5 - P} & P \le x_i < \frac{1}{2} \\ \frac{1 - d - x_i}{0.5 - P} & \frac{1}{2} \le x_i < 1 - P \\ \frac{1 - x_i}{P} & 1 - P \le x_i < 1 \end{cases}$ P=0.7	(0, 1)
7	Sine	$x_{i+1} = \frac{a}{4}\sin(\pi x_i) \quad a = 4$	(0, 1)
8	Singer	$x_{i+1} = \mu\left(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4\right),\ \mu = 2.3$	(0, 1)
9	Sinusoidal	$x_{i+1} = ax_i^2\sin(\pi x_i),\ a = 2.3$	(0, 1)
10	Tent	$x_{i+1} = \begin{cases} \frac{x_i}{0.7}x_i < 0.7 \\ \frac{10}{3}(1 - x_i)x_i \ge 0.7 \end{cases}$	(0, 1)

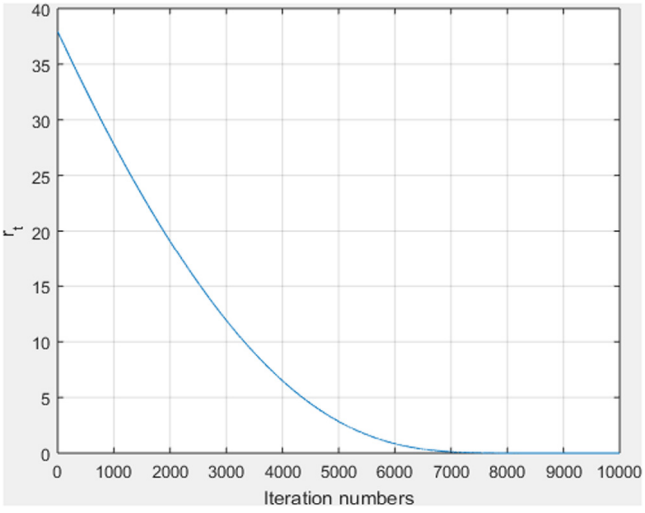

Fig. 1. r_t value for $Maxt = 10\,000$, $x = 0.1$ and $a_0 = 1$.

(10)

where $C_m(t)$ is the normalized chaos map value as defined in [43] according to the current iteration number, r is a random number in [0, 1] interval and abs represent absolute value function. The formulation of the $C_m(t)$ was given as follows;

$$C_m(t) = \frac{(CM(t) - CM_{min})(N_{ce}(t))}{(CM_{max} - CM_{min})} \tag{11}$$

where CM represents the chaos map, CM_{min} and CM_{max} are the maximum and minimum values of the chaos map and $N_{ce}(t)$ was calculated as follows to normalize the effect of the chaos map to the preferred range.

$$N_{ce}(t) = N_{ce_{max}} - \left(\frac{t}{Maxt}\right)\left(N_{ce_{max}} - N_{ce_{min}}\right) \tag{12}$$

where $N_{ce_{max}}$ and $N_{ce_{min}}$ are maximum and minimum limit values of the preferred range of the chaos effect. Example graphics of the Gauss–Mouse chaos map and the C_m that calculated by using the Gauss–Mouse map with $Maxt = 100$ were given in Fig. 2a and b.

In Eq. (10), $f_{b_{ratio}}$ is a variable that represent the ratio between the best fitness values of two consecutive iterations defined according to the following conditions.

$$\begin{aligned} &\left(if\ t == 1\ then\ f_{b_{ratio}} = 1\right)\ and\ \Big(if\ f_{min} < f_{best}\ then \\ &\quad f_{b_{ratio}} = \frac{fmin}{fbest}\ and\ f_{best} = f_{min}\Big)\ and \\ &\left(if\ abs\left(f_{b_{ratio}}\right) > 2\ then\ f_{b_{ratio}} = 2r\ elseif\ abs\left(f_{b_{ratio}}\right) < 1\ then\right. \\ &\quad \left.f_{b_{ratio}} = r\right) \end{aligned} \tag{13}$$

where f_{best} is the best fitness value obtained so far to the current iteration while f_{min} is the minimum fitness value obtained at that iteration. By using $f_{b_{ratio}}$ the effect of the chaos map on the decreasing process of the r_t is balanced. So, if the algorithm cannot find a better solution than the current solution ($f_{min} \ge f_{best}$) at the current iteration, $f_{b_{ratio}}$ remains unchanged in order to increase the effect of the chaos map according to the last $f_{b_{ratio}}$ value. On the other hand, as the algorithm continues to minimize the f_{min} value, $f_{b_{ratio}}$ will decrease and therefore the effect of the chaos map will also decrease. In this study, the maximum and minimum value of the $f_{b_{ratio}}$ are determined as 2 and 1 respectively, in order to prevent excessive increases and decreases effects of the chaos map on the decreasing process of the r_t value. The proposed decreasing process of the r_t value is shown in Fig. 3 by using $Maxt = 1000$, $x = 0.1$ and $a_0 = 1$ and the Gauss–Mouse chaotic map.

As can be seen from Fig. 3, in the proposed CVS algorithm decreasing of the r_t value can be changed by the effect of the chaos map and the $f_{b_{ratio}}$ while preserving its original form. Therefore the exploration ability of the algorithm increases while the maximum NOI remains unchanged.

Proposed method 2: Since the best solution at each one of the iterations is used to determine the other candidate solutions, improving the ability of the escaping from the local minimum of the algorithm can be provided by changing determination strategy of the best solution. Thus, in this study we applied the chaos effect to the determination of the best solution at each one

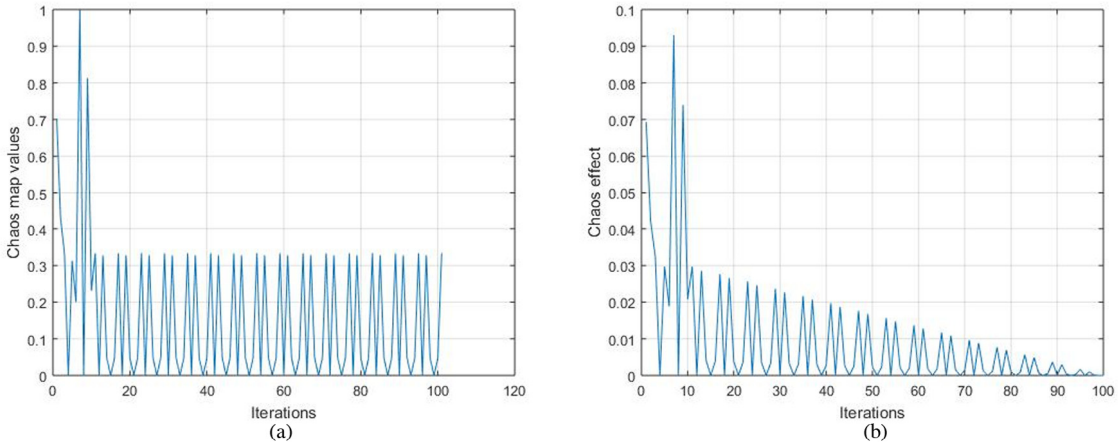

Fig. 2. Sample graphs of the chaos map (a) and the C_m variable (b) for Gauss–Mouse chaotic map and $Maxt = 100$.

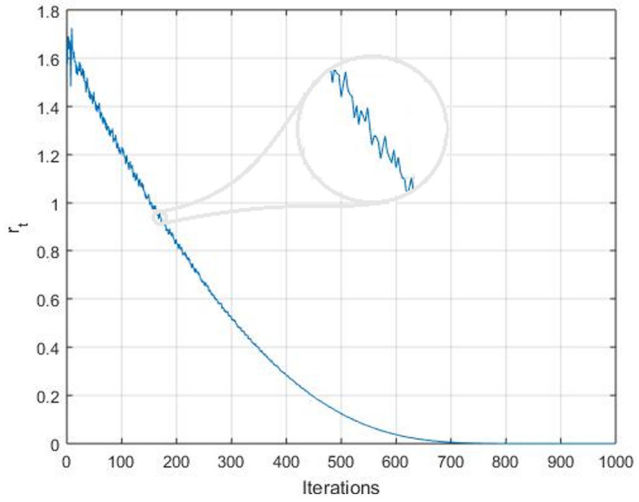

Fig. 3. The proposed decreasing process of the r_t value for $Maxt = 1000$, $x = 0.1$ and $a_0 = 1$ and the Gauss–Mouse chaotic map.

of the iterations as follows;

$$\begin{aligned}
&\textit{if}\ \ (f_{min} < g_{min})\ \ \textit{then}\ \ g_{min} = f_{min}\ \textit{and}\ \mu_t = \textit{itrBest}\\
&\quad \textit{elseif}\ \ (r < 0.5)\ \ \textit{then}\ \ C\mu_t = C_m\,(t)\,\textit{itrBest}\\
&\qquad \textit{if}\ f\,(C\mu_t) < g_{min}\ \textit{then}\ g_{min} = f\,(C\mu_t)\ \ \textit{and}\ \mu_t = C\mu_t\\
&\qquad \textit{endif}\\
&\quad \textit{endif}\\
&\textit{endif}
\end{aligned} \tag{14}$$

where, μ_t is the best solution determined for the current iteration while *itrBest* is the best solution obtained so far to that iteration, $C\mu_t$ is the proposed solution and $f\,(C\mu_t)$ is fitness value of the $C\mu_t$. In the VS algorithm, if the algorithm stuck at a local minimum, *itrBest* and therefore μ_t will not be changed and the new solutions will be produced around the same best solution at the ongoing iterations. On the other hand in the proposed solution given above, when the algorithm stuck in a local minimum another solution near the current solution is provided by using the chaos map and the current best solution, with a %50 probability. If the proposed solution produces a better solution than the current best solution, the algorithm continues from this new solution. Therefore escaping from the local minimum can be achieved more successfully than the original form.

3. Experimental studies

In this section, firstly fifty benchmark functions were used to test the performance of the proposed chaos based VS algorithms. And, the CVS algorithm that shows the best performance was determined according to the results of this first test. Then, in order to show "how the chaos map affects the search abilities of the VS algorithm", a second test was performed. After that, the effect of the proposed methods on reducing the maximum NOI was presented in a third test. Finally, a time complexity analysis was given to show the effect of the proposed methods on the computational cost of the VS algorithm. The selected benchmark functions list were conducted by the study in [12] from some sets of the unimodal separable functions (US), unimodal nonseparable functions(UN), multimodal separable functions(MS) and multimodal nonseparable functions (MN). All the benchmark functions were given in Table 2.

3.1. Selection of the best CVS algorithm

In the first experiment both the VS and the ten CVS algorithms (CVS1-CVS10) were run 30 times to solve the test functions and the minimum objective function values were recorded. In all the executions the number of the maximum NOI was defined as 10 000, the number of the candidate solutions was defined as 50 and $N_{ce_{max}}$ and $N_{ce_{min}}$ were defined as 0.1 and $1E-21$, respectively. The obtained results were evaluated according to mean, standard deviation and the best solution for 30 runs. These statistics were given in Table 3. In the table, the best of the mean values were written in bold type. And, the numbers of the benchmark functions that each algorithm obtained the best objective function values for were given at the end row of the table as wn.

According to Table 3, it can be seen that the proposed methods improve the performance of the VS algorithm in terms of all the function groups, US, UN, MS, and MN. The only two benchmark functions that the VS algorithm showed better performance than the CVS algorithms are F33 and F37. On the other hand, in terms of the other functions, the CVS algorithms get better or equal results than the VS algorithm. The wn numbers of the algorithms can also be evaluated to compare the performance of the algorithms and to select the best CVS algorithm. According to these numbers, it is seen that the VS algorithm obtained the best results for twenty-four benchmark functions while all the CVS algorithms showed the best performance for more than twenty-six benchmark functions. Further, according to the mean and the standard deviations of the results it can be observed that the

Table 2
Benchmark functions from [12].

No	Range	D	Type	Function	Min	Formulation	No	Range	D	Type	Function	Min	Formulation
F1	$[-5.12,\ 5.12]$	5	US	Stepint	0	$f(x) = 25 + \sum_{i=1}^{5} [x_i]$	F26	$[0, \pi]$	10	MS	Michalewicz10	−9.6602	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}\ m = 10$
F2	$[-100, 100]$	30	US	Step	0	$f(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	F27	$[-100,\ 100]$	2	MN	Schaffer	0	$f(x) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5\right)}{(1+0.0001(x_1^2+x_2^2))^2}$
F3	$[-100, 100]$	30	US	Sphere	0	$f(x) = \sum_{i=1}^{n} (x_i)^2$	F28	$[-5,\ 5]$	2	MN	Six Hump Camel Back	−1.03163	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
F4	$[-10, 10]$	30	US	SumSquares	0	$f(x) = \sum_{i=1}^{n} (ix_i)^2$	F29	$[-100,\ 100]$	2	MN	Bohachevsky2	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
F5	$[-1.28, 1.28]$	30	US	Quartic	0	$f(x) = \sum_{i=1}^{n} (ix_i)^4 + random[0, 1]$	F30	$[-100,\ 100]$	2	MN	Bohachevsky3	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
F6	$[-4.5, 4.5]$	5	UN	Beale	0	$f(x) = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$	F31	$[-10,\ 10]$	2	MN	Shubert	−186.73	$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$
F7	$[-100, 100]$	2	UN	Easom	−1	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2 - (x_2-\pi)^2)$	F32	$[-2,\ 2]$	2	MN	GoldenStein–Price	3	$f(x) = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]\left[30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$
F8	$[-10, 10]$	2	UN	Matyas	0	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	F33	$[-5,\ 5]$	4	MN	Kowalik	0.00031	$f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1\left(b_i^2 + b_ix_2\right)}{b_i^2 + b_ix_3 + x_4}\right)^2$
F9	$[-10,\ 10]$	4	UN	Colville	0	$f(x) = 100\left(x_1^2 - x_2\right)^2 + (x_1-1)^2 + (x_3-1)^2 + 90\left(x_3^2 - x_4\right)^2 + 10.1\left((x_2-1)^2 + (x_4-1)^2\right) + 19.8(x_2-1)(x_4-1)$	F34	$[0, 10]$	4	MN	Shekel5	−10.15	$f(x) = -\sum_{i=1}^{5}\sum_{j=1}^{4}\left[\left(x_j - a_{ij}\right)\left(x_j - a_{ij}\right)^T + c_i\right]^{-1}$
F10	$[-D^2, D^2]$	6	UN	Trid6	−50	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_ix_{i-1}$	F35	$[0, 10]$	4	MN	Shekel7	−10.4	$f(x) = -\sum_{i=1}^{7}\sum_{j=1}^{4}\left[\left(x_j - a_{ij}\right)\left(x_j - a_{ij}\right)^T + c_i\right]^{-1}$
F11	$[-D^2, D^2]$	10	UN	Trid10	−210	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_ix_{i-1}$	F36	$[0, 10]$	4	MN	Shekel10	−10.53	$f(x) = -\sum_{i=1}^{10}\sum_{j=1}^{4}\left[\left(x_j - a_{ij}\right)\left(x_j - a_{ij}\right)^T + c_i\right]^{-1}$
F12	$[-5, 10]$	10	UN	Zakharov	0	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$	F37	$[D,\ D]$	4	MN	Perm	0	$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k + \beta\right)\left((x_i/i)^k - 1\right)\right]^2$
F13	$[-4, 5]$	24	UN	Powell	0	$f(x) = \sum_{i=1}^{n/k}(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$	F38	$[0, D]$	4	MN	PowerSum	0	$f(x) = \sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n} x_i^k\right) - b_k\right]^2$
F14	$[-10, 10]$	30	UN	Schwefel 2.22	0	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	F39	$[0, 1]$	3	MN	Hartman3	−3.86	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
F15	$[-10, 10]$	30	UN	Schwefel 1.2	0	$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	F40	$[0, 1]$	6	MN	Hartman6	−3.32	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
F16	$[-30, 30]$	30	UN	Rosenbrock	0	$f(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	F41	$[-600,\ 600]$	30	MN	Griewank	0	$f(x) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F17	$[-10, 10]$	30	UN	Dixon–Price	0	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$	F42	$[-32,\ 32]$	30	MN	Ackley	0	$f(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
F18	$[-65.536, 65.536]$	2	MS	Foxholes	0.998	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)6}\right]^{-1}$	F43	$[-50,\ 50]$	30	MN	Penalized	0	$f(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\ y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$

(*continued on next page*)

Table 2 (*continued*).

No	Range	D	Type	Function	Min	Formulation	No	Range	D	Type	Function	Min	Formulation
F19	$[-5, 10] \times [0, 15]$	2	MS	Branin	0.398	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}{x_1}^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	F44	$[-50, 50]$	30	MN	Penalized2	0	$f(x) = 0.1\left\{\sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F20	$[-100, 100]$	2	MS	Bohachevsky1	0	$f(x) = {x_1}^2 + 2{x_2}^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	F45	$[0, 10]$	2	MN	Langerman2	−1.08	$f(x) = -\sum_{i=1}^{m} c_i \left(exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right) \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$
F21	$[-10, 10]$	2	MS	Booth	0	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	F46	$[0, 10]$	5	MN	Langerman5	−1.5	$f(x) = -\sum_{i=1}^{m} c_i \left(exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right) \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$
F22	$[-5.12, 5.12]$	30	MS	Rastrigin	0	$f(x) = \sum_{i=1}^{n}[{x_i}^2 - 10\cos(2\pi x_i) + 10]$	F47	$[0, 10]$	10	MN	Langerman10	NA	$f(x) = -\sum_{i=1}^{m} c_i \left(exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right) \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$
F23	$[-500, 500]$	30	MS	Schwefel	−12569.5	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	F48	$[d]$	2	MN	Fletcher Powell2	0	$f(x) = \sum_{i=1}^{n}(A_i - B_i)^2 \quad A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right) \quad B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
F24	$[0, \pi]$	2	MS	Michalewicz2	−1.8013	$f(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m} \quad m = 10$	F49	$[d \times 1]$	5	MN	Fletcher Powell5	0	$f(x) = \sum_{i=1}^{n}(A_i - B_i)^2 \quad A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right) \quad B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
F25	$[0, \pi]$	5	MS	Michalewicz5	−4.6877	$f(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m} \quad m = 10$	F50	$[d \times 1]$	10	MN	Fletcher Powell10	0	$f(x) = \sum_{i=1}^{n}(A_i - B_i)^2 \quad A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right) \quad B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$

CVS algorithms shown more stable performances than the VS algorithm. Thus, it can be said that the CVS algorithms outperform the classical form of the VS algorithm. The results given in Table 3 can also be used to determine the best CVS algorithm. According to the table, it is seen that the CVS3 is the best algorithm among the other CVS algorithms in terms of the wn numbers since it obtained the best results for thirty-eight benchmark functions while the second is CVS6 by thirty benchmarks. Moreover, from the table, it can be observed that the CVS3 algorithm obtained the exact global optimum point for eighteen benchmark functions while the CVS6 shows the same performance for only twelve functions. In order to perform a more detailed analysis to present the reason behind the success of the CVS3, we selected fourteen benchmark functions; F4 and F5 from US group, F8, F12 and F17 from UN group, F18, F19 and F20 from MS group and finally F30, F37, F38, F41, F42 and F50 from the MNS group. Then we drew the convergence graphics of the ten CVS algorithms in solving these benchmarks for $Maxt = 1000$. It should be noted that the other parameters were left as defined in the first test. The results were given in Fig. 4 and evaluated in terms of two criteria, the fast convergence to the optimum and the obtained best solution. From the figure, it is seen that CVS3 shows significantly best performance for the F4, F8, F12 F17, F18, and F42 for both of the criteria. The CVS algorithm in the second order is the CVS6 which provides the best performance for the F37 for the both of the criteria and for the F19 for the fast convergence characteristics. Although, the other CVS algorithms show acceptable results in terms of the both of the comparing criteria, their performances were not sufficient to be selected as the best CVS algorithm. Therefore, in this study, the CVS3 algorithm defined by the Gauss–Mouse map was selected as the best CVS algorithm to solve the inverse kinematics problem of a six DOF serial robot manipulator with offset wrist.

3.2. How the chaos map affects the search abilities of the VS algorithm

As explained in the first test, the proposed methods improved the performance of the VS algorithm. In this second test, we aim to visually present this improvement by showing the effects of the proposed methods on the searching characteristics of the VS algorithm. Therefore, we selected four benchmark functions, namely F5, F8, F36, and F48 and then we solved these functions by both of the CVS3 and the VS algorithms for their 2-dimensional forms. As the aim was to visually show the searching behaviors of the algorithms, we choose the maximum NOI only as 100 and left the other parameters to be as given in the first test. In this test, the objective function values and μ_t and r_t values were recorded through the iterations. It should be noted that for a 2-dimensional problem, μ_t and r_t values represent the center and the radius values of the circle that is used to search for the best solution in one iteration. In Fig. 5, we first draw the surfaces for the selected benchmark functions between their upper and lower limits. Then we draw the nested circles obtained by using the recorded μ_t and r_t values on these surfaces. The circles were drawn not only according to their center and the radius values but also according to the objective function values obtained at the related iterations. Each one of the circles that have the same objective function value was drawn by using the same color. Thus, it was provided to show that the algorithm is at a local optimum point if the nested circles have the same color. And also the numbers of the circles show how many iterations are used at that local optimum. Moreover, the change in the distances between the edge of the circles shows how fast the algorithm reduces the r_t and changes the μ_t values to escape the local optimum. In the figure, for the F5 function, it can be seen that there are only three circle groups of different colors in the drawing of the VS algorithm while there are five circle groups in the drawing of the CVS3 algorithm. And also it can be seen that the circle groups of the VS algorithm have more than five circles in each one of the groups while the groups of the CVS3 include different numbers of circles from one to more. Moreover, it is seen that the CVS3 obtained 1.0823e−05 value as the best solution while VS reached 0.0030603 value at the end of the iterations. Therefore, it can be said that the CVS3 algorithm was more capable than the VS algorithm in terms of exploring the search space efficiently and escaping the local optimums. The effect of the chaos map on the exploration ability of the VS algorithm can also be seen more clearly in the figure that was drawn for the F36. In that figure, it can be observed that although the CVS3 algorithm encountered more local optimum points than the VS algorithm while searching the problem space, it was able to successfully escape those points. On the other hand, it is seen that the VS algorithm used many more iterations to escape from fewer numbers of local optimums. The effects of the proposed methods on the exploitation ability of the VS algorithms can be observed from the drawings of the functions F8 and F48. If it is looked at the distances between the circles drawn on these two functions, it can be seen that the circles that have the same color and are drawn for the VS algorithm have equal distances from each other. These cause the algorithm to be stuck more likely in the local optimum. On the other hand, from the figure, it can be realized that the circles drawn for the CVS3 algorithm for these functions have different distances from each other. This is occurred because of the second method proposed in this study and it helps the algorithm to escape faster from the local optimums. As this test was performed only for 2-dimensional problems, we performed an additional test to show the differences between the search characteristics of the VS and the CVS3 algorithms. In this test we run the algorithms for 1000 maximum NOI and drew the convergence graphics of the algorithms in Fig. 6 for the selected fourteen benchmark functions in the first test. From the figure, the improvements in the searching abilities of the VS algorithm can be seen by comparing the changes that occurred on the convergence graphics through the iterations and also by comparing the best objective function values obtained at the end of the iterations. As can be seen, the CVS3 algorithm reaches better solutions than the VS algorithms for all the four function groups. Besides, in the figure, it is seen that VS algorithm convergence to the optimum points towards the end of the iterations while CVS3 reaches near the optimum points at the approximately mid of the iterations. Thus, the convergence rate of the VS algorithm to the optimum was increased with the help of the proposed methods. As a final point, if all the results of the second test evaluated together, it can be said that the proposed methods improved the exploration and exploitation abilities of the VS algorithm successfully.

3.3. Effect of the proposed methods on reducing the maximum NOI of the VS algorithm

So as to see the effect of the proposed methods on reducing the maximum NOI, in this test, the performance of the VS and CVS3 algorithms by using the different maximum NOI were compared. In this test, the VS algorithm was executed 30 times for $Maxt = 100\,000$ and the results of this experiment were compared with the results of the both of the VS and CVS3 algorithms getting from Table 3. The other parameters of the algorithm, except the maximum NOI, were used as defined in the first experiment. The obtained results evaluated according to the mean, standard deviation and minimum of the objecting function values and also evaluated by using a pairwise statistical test, Wilcoxon Signed-Rank Test with a statistical significance value $p = 0.05$.

Table 3
Statistical results of the VS and ten CVS algorithms for the benchmark functions.

F	Func Min	Statics	VS	CVS1	CVS2	CVS3	CVS4	CVS5	CVS6	CVS7	CVS8	CVS9	CVS10
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f1	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	1,6	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f2	0	std	1,404753834	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	2,19212E−36	4,9E−100	6,2E−134	**0**	3,1E−141	9E−137	2,5E−127	8,7E−157	2,5E−114	1,73E−75	1,1E−141
f3	0	std	3,58142E−36	2,7E−99	3,3E−133	0	1,6E−140	4,8E−136	1,3E−126	0	1,3E−113	6,86E−75	6,2E−141
		min	1,93E−38	9,6E−127	2E−163	0	3,3E−181	6,2E−178	7E−160	1E−198	5,2E−135	9,94E−86	2,4E−184
		mean	2,59412E−15	1,1E−100	6,5E−135	**0**	2,4E−140	1E−142	1,4E−129	3E−162	1,5E−115	3,27E−69	9E−133
f4	0	std	6,43069E−15	5,6E−100	2,5E−134	0	1,1E−139	4,7E−142	7,7E−129	0	6,8E−115	1,76E−68	4,8E−132
		min	3,7E−21	2,3E−129	7,8E−171	0	3,3E−186	6E−182	5,8E−162	9,3E−209	1E−137	8,27E−83	3,2E−184
		mean	0,005872765	0,000302	0,00022	**0,000137**	0,000212	0,000211	0,000218	0,000188	0,000313	0,000371	0,000243
f5	0	std	0,003166123	0,00028	0,000194	9,01E−05	0,000148	0,000193	0,000173	0,000137	0,000497	0,000329	0,000256
		min	0,001624907	8,23E−06	2,28E−06	1,88E−05	2,24E−05	7,02E−06	2,53E−05	4,23E−06	1,09E−07	2,97E−05	9,99E−06
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f6	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**
f7	−1	std	0	0	0	0	0	0	0	0	0	0	0
		min	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1
		mean	1,9278E−119	9,5E−158	2,2E−174	**0**	1,2E−172	5,2E−178	1E−172	1,6E−185	3E−153	2,1E−143	4,4E−183
f8	0	std	1,0338E−118	0	0	0	0	0	0	0	1,6E−152	1,1E−142	0
		min	2,09E−160	1,1E−191	1,2E−211	0	1,7E−225	1,6E−213	1,4E−210	6,4E−222	1,9E−215	1,9E−180	1,4E−224
		mean	3,25014E−16	3,44E−16	1,4E−15	1,09E−15	2,07E−15	**2,56E−16**	5,7E−16	4,42E−16	1,63E−15	2,46E−15	3,37E−16
f9	0	std	8,36282E−16	1,13E−15	4,81E−15	4,76E−15	9,47E−15	7,4E−16	9,5E−16	7,88E−16	7,45E−15	9,19E−15	5E−16
		min	1,11E−24	2,69E−22	9,21E−24	6,36E−22	1,15E−21	3,22E−21	1,26E−21	2,3E−20	2,13E−22	3,96E−23	1,43E−20
		mean	**−50**	**−50**	**−50**	**−50**	**−50**	**−50**	**−50**	**−50**	**−50**	**−50**	**−50**
f10	−50	std	0	4,43E−14	4,45E−14	4,64E−14	4,71E−14	4,57E−14	4,57E−14	4,84E−14	4,51E−14	4,81E−14	4,57E−14
		min	−50	−50	−50	−50	−50	−50	−50	−50	−50	−50	−50
		mean	**−210**	**−210**	**−210**	**−210**	**−210**	**−210**	**−210**	**−210**	**−210**	**−210**	**−210**
f11	−210	std	0	9,19E−13	9,58E−13	6,7E−13	9,89E−13	1,22E−12	8,83E−13	1,46E−12	1,44E−12	7,95E−13	1,43E−12
		min	−210	−210	−210	−210	−210	−210	−210	−210	−210	−210	−210
		mean	3,09792E−51	3,4E−104	7,1E−138	**0**	2,3E−136	1,7E−144	1,8E−137	4,4E−159	8,3E−117	3,26E−84	6,7E−137
f12	0	std	8,89299E−51	1,8E−103	3,8E−137	0	1,2E−135	9,3E−144	9,5E−137	0	4,1E−116	1,15E−83	3,6E−136
		min	3,28E−59	1,9E−126	2E−168	0	9,9E−174	9,7E−186	1,9E−162	1,1E−196	1E−144	9,02E−95	5,7E−188
		mean	0,007024223	6,4E−106	1,4E−132	**0**	2,5E−137	2,3E−142	1,5E−135	4,3E−157	2,2E−117	1,9E−80	2,3E−142
f13	0	std	0,001191264	3,5E−105	6,4E−132	0	1,3E−136	1,2E−141	8,2E−135	0	8,8E−117	5,25E−80	1,2E−141
		min	0,004002179	1,2E−128	2,5E−169	0	5,4E−174	4,2E−183	1,9E−163	3,1E−206	1,2E−144	3,34E−93	2,5E−183
		mean	1,55242E−09	2,25E−49	4,35E−69	**4E-211**	2,31E−70	1,86E−67	7,25E−63	1,25E−76	1,07E−58	1,99E−37	2,33E−73
f14	0	std	5,14585E−09	1,21E−48	2,32E−68	0	9,07E−70	1E−66	3,9E−62	6,72E−76	3,86E−58	6,34E−37	1,22E−72
		min	1,71E−15	2,06E−64	2,62E−83	0	5,87E−89	9,78E−95	9,37E−79	6,95E−95	7,3E−70	2,64E−42	7,14E−90
		mean	2,45938E−11	2,4E−103	1,9E−131	**0**	2,5E−137	1,2E−138	1,1E−128	4,6E−140	2,6E−110	8,37E−70	9,3E−135
f15	0	std	4,72956E−11	1,3E−102	1E−130	0	1,3E−136	6,2E−138	5,7E−128	2,5E−139	9,9E−110	4,24E−69	5E−134
		min	5,1E−13	8,7E−129	1,5E−161	0	1E−173	1,6E−176	1,9E−160	3,9E−198	4,1E−137	4,75E−83	3,3E−171
		mean	34,061437	25,80616	25,89813	**25, 753**	25,85653	25,85967	25,86765	25,81812	25,92035	25,9307	25,91957
f16	0	std	29,73415692	0,248239	0,216031	0,204932	0,254647	0,203636	0,190027	0,210111	0,183827	0,188315	0,20533
		min	20,01080875	25,28427	25,29765	25,31584	25,40616	25,38295	25,45278	25,26193	25,62714	25,55433	25,40888
		mean	0,668062501	**0,666667**	**0,666667**	**0,666667**	**0,666667**	**0,666667**	**0,666667**	**0,666667**	**0,666667**	**0,666667**	**0,666667**
f17	0	std	0,007153691	3,9E−07	3,05E−07	3,01E−07	3,41E−07	2,32E−07	2,02E−07	2,39E−07	3,63E−07	2,21E−07	2,03E−07
		min	0,666666695	0,666667	0,666667	0,666667	0,666667	0,666667	0,666667	0,666667	0,666667	0,666667	0,666667
		mean	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**	**0,998004**
f18	0, 998	std	4,44089E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16	3,33E−16
		min	0,998003838	0,998004	0,998004	0,998004	0,998004	0,998004	0,998004	0,998004	0,998004	0,998004	0,998004
		mean	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**	**0,397887**
f19	0, 398	std	1,11022E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16	1,11E−16
		min	0,397887358	0,397887	0,397887	0,397887	0,397887	0,397887	0,397887	0,397887	0,397887	0,397887	0,397887
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f20	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f21	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	70,3434074	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	39,2345	**0**
f22	0	std	21,09076411	0	0	0	0	0	0	0	0	19,07392	0

(*continued on next page*)

Table 3 (*continued*).

F	Func Min	Statics	VS	CVS1	CVS2	CVS3	CVS4	CVS5	CVS6	CVS7	CVS8	CVS9	CVS10
		min	32,83357323	0	0	0	0	0	0	0	0	9,949591	0
		mean	−10555,2341	−10500	−10442,7	**−10738,2**	−10626,3	−10496,6	−10474,3	−10470,3	−10500	−10505,2	−10374,3
f23	−12.569, 5	std	599,1476133	590,8378	450,1847	515,155	407,3875	525,912	713,7995	445,3067	495,9537	549,6553	703,8505
		min	−11819,3636	−11839,1	−11444,3	−11523,2	−11444,3	−11464,1	−11404,8	−11582,5	−11661,5	−11464	−11799,6
		mean	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**	**−1,8013**
f24	−1,8013	std	6,66134E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16
		min	−1,80130341	−1,8013	−1,8013	−1,8013	−1,8013	−1,8013	−1,8013	−1,8013	−1,8013	−1,8013	−1,8013
		mean	−4,53385152	−4,57941	−4,56179	−4,58817	−4,54343	−4,57864	−4,58391	−4,50991	−4,56956	−4,55348	−4,56982
f25	−4,6877	std	0,112839192	0,101474	0,093973	0,125046	0,120682	0,116052	0,128083	0,189494	0,089165	0,136816	0,103228
		min	−4,68765818	−4,68766	−4,68766	−4,68766	−4,68766	−4,68766	−4,68766	−4,68766	−4,68766	−4,68766	−4,68766
		mean	−8,46602237	−8,63814	−8,51872	−8,51652	−8,56995	−8,63945	**−10,4029**	−8,7038	−8,36627	−8,5621	−8,3416
f26	−9,6602	std	0,596666899	0,518618	0,649331	0,658733	0,604426	0,567071	0,45518	0,596524	0,655977	0,553302	0,717263
		min	−9,32950179	−9,4015	−9,41113	−9,27467	−9,21475	−9,43097	−9,43933	−9,55342	−9,47404	−9,37027	−9,62254
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f27	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**	**−1,03163**
f28	−1,03163	std	0	0	0	0	0	0	0	0	0	0	0
		min	−1,03162845	−1,03163	−1,03163	−1,03163	−1,03163	−1,03163	−1,03163	−1,03163	−1,03163	−1,03163	−1,03163
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f29	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f30	0	std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
		mean	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**	**−186,731**
f31	−186,73	std	8,52651E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14	2,84E−14
		min	−186,730909	−186,731	−186,731	−186,731	−186,731	−186,731	−186,731	−186,731	−186,731	−186,731	−186,731
		mean	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
f32	3	std	0	2,98E−15	2,98E−15	2,52E−15	2,52E−15	1,78E−15	1,92E−15	2,45E−15	4,44E−16	2,52E−15	1,78E−15
		min	3	3	3	3	3	3	3	3	3	3	3
		mean	**0,000309537**	0,000307	0,000307	0,000307	0,000307	0,000307	0,000338	0,000307	0,000307	0,000307	0,000307
f33	0,00031	std	1,10428E−05	4,58E−19	4,33E−19	4,33E−19	4,37E−19	6,04E−19	0,000164	6,15E−19	9,92E−19	5,27E−19	3,87E−19
		min	0,000307486	0,000307	0,000307	0,000307	0,000307	0,000307	0,000307	0,000307	0,000307	0,000307	0,000307
		mean	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**	**−10,1532**
f34	−10,15	std	0	1,78E−15	1,78E−15	1,78E−15	1,78E−15	1,78E−15	1,78E−15	1,78E−15	1,78E−15	1,78E−15	1,78E−15
		min	−10,1531997	−10,1532	−10,1532	−10,1532	−10,1532	−10,1532	−10,1532	−10,1532	−10,1532	−10,1532	−10,1532
		mean	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**	**−10,4029**
f35	−10,4	std	0	0	0	0	0	0	0	0	0	0	0
		min	−10,4029406	−10,4029	−10,4029	−10,4029	−10,4029	−10,4029	−10,4029	−10,4029	−10,4029	−10,4029	−10,4029
		mean	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**	**−10,5364**
f36	−10,53	std	0	8,88E−15	1,93E−14	2,59E−14	2,59E−14	2,55E−14	2,55E−14	8,88E−15	8,88E−15	1,93E−14	8,88E−15
		min	−10,5364098	−10,5364	−10,5364	−10,5364	−10,5364	−10,5364	−10,5364	−10,5364	−10,5364	−10,5364	−10,5364
		mean	**0,00232**	0,0096	0,002707	0,013838	0,002803	0,007344	0,011271	0,002803	0,007096	0,005944	0,005814
f37	0	std	0,00232835	0,028659	0,002429	0,033317	0,002575	0,023577	0,032432	0,002393	0,023964	0,017484	0,017653
		min	0,000000299	1,2E−07	1,51E−06	5,38E−06	1,68E−06	2,31E−06	4,47E−07	2,12E−06	7,18E−07	4,07E−07	1,84E−07
		mean	0,000232286	0,000251	0,000141	0,00016	**0,000101**	0,000145	0,000133	0,000132	0,000161	0,000137	0,000169
f38	0	std	0,000149149	0,000147	0,000136	0,000164	0,000112	0,000141	0,000131	0,000145	0,000164	0,000132	0,000158
		min	4,05E−08	6,9E−07	1,26E−09	6,66E−11	3,16E−10	1,01E−07	6,62E−10	8,06E−10	1,14E−10	2,29E−08	2,55E−09
		mean	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**	**−3,86278**
f39	−3,86	std	2,22045E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15	2,66E−15
		min	−3,86278215	−3,86278	−3,86278	−3,86278	−3,86278	−3,86278	−3,86278	−3,86278	−3,86278	−3,86278	−3,86278
		mean	−3,28263265	−3,28661	−3,25879	−3,27866	**−3,29455**	−3,29058	−3,28263	−3,28263	−3,26674	−3,28263	−3,28263
f40	−3,32	std	0,056194291	0,054627	0,05947	0,057445	0,050419	0,052715	0,056194	0,056194	0,05947	0,056194	0,056194
		min	−3,32236801	−3,32237	−3,32237	−3,32237	−3,32237	−3,32237	−3,32237	−3,32237	−3,32237	−3,32237	−3,32237
		mean	0,017147472	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f41	0	std	0,010950619	0	0	0	0	0	0	0	0	0	0
		min	1,11E−16	0	0	0	0	0	0	0	0	0	0
		mean	0,099554831	**8,88E−16**	**8,88E−16**	**8,88E−16**	**8,88E−16**	**8,88E−16**	**8,88E−16**	**8,88E−16**	**8,88E−16**	4,44E−15	**8,88E−16**
f42	0	std	0,374586285	9,86E−32	9,86E−32	9,86E−32	9,86E−32	9,86E−32	9,86E−32	9,86E−32	9,86E−32	1,58E−30	9,86E−32
		min	1,51E−14	8,88E−16	8,88E−16	8,88E−16	8,88E−16	8,88E−16	8,88E−16	8,88E−16	8,88E−16	4,44E−15	8,88E−16
		mean	12,84505515	**1,8E−31**	2,36E−31	2,61E−31	2,6E−31	5,9E−31	2,89E−31	3,45E−31	5,13E−31	3,18E−31	3,26E−31
f43	0	std	19,41523547	1,28E−31	1,77E−31	2,89E−31	2,34E−31	9,72E−31	2,39E−31	3,72E−31	1,45E−30	2,4E−31	3,37E−31
		min	1,14E−31	2,6E−32	5,7E−32	3,64E−32	5,18E−32	6,6E−32	4,15E−32	5,18E−32	4,8E−32	4,15E−32	5,18E−32
		mean	2,13E−32	2,13E−32	2,19E−32	**1,72E−32**	1,97E−32	2,33E−32	1,75E−32	2,33E−32	1,94E−32	2,02E−32	1,82E−32
f44	0	std	8,57465E−33	7,61E−33	8,86E−33	7,94E−33	8,35E−33	1,08E−32	8,69E−33	8,43E−33	9,69E−33	1,23E−32	9,27E−33
		min	6,43E−33	6,43E−33	6,43E−33	6,43E−33	6,43E−33	2,73E−33	1,5E−33	8,9E−33	1,5E−33	3,96E−33	2,73E−33

(*continued on next page*)

Table 3 (*continued*).

F	Func Min	Statics	VS	CVS1	CVS2	CVS3	CVS4	CVS5	CVS6	CVS7	CVS8	CVS9	CVS10
f45	−1,08	mean	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**	**−1,08094**
		std	4,44089E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16	6,66E−16
		min	−1,08093844	−1,08094	−1,08094	−1,08094	−1,08094	−1,08094	−1,08094	−1,08094	−1,08094	−1,08094	−1,08094
f46	−1,5	mean	−1,48216588	**−1,5**	−1,48133	−1,48217	−1,48217	−1,48217	−1,48217	**−10,4029**	−1,48217	−1,48217	−1,46433
		std	0,096035513	4,76E−15	0,100554	0,096036	0,096036	0,096036	0,096036	4,82E−15	0,096036	0,096036	0,133453
		min	−1,49999922	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5
f47	NA	mean	−0,73471622	−0,87646	−0,80656	−0,79804	−0,84621	−0,82155	**−0,89052**	−0,75225	−0,84403	−0,76935	−0,72198
		std	0,406550166	0,367307	0,354821	0,372764	0,370617	0,364567	0,41428	0,413779	0,379988	0,392993	0,321314
		min	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5	−1,5
f48	0	mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		std	0	0	0	0	0	0	0	0	0	0	0
		min	0	0	0	0	0	0	0	0	0	0	0
f49	0	mean	0,003026356	0,000112	0,003107	1,53E−05	0,000848	0,000289	**2,72E−07**	6,47E−06	0,002065	0,001895	9,56E−05
		std	0,011334942	0,0006	0,010723	8,21E−05	0,004563	0,001559	1,25E−06	3,12E−05	0,011122	0,010206	0,000514
		min	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29	5,05E−29
f50	0	mean	4,405111857	10,47737	5,259899	3,473492	5,969255	4,296374	7,50296	3,798208	3,27651	**2,867465**	16,63537
		std	14,70546042	22,32809	10,87775	6,127262	12,87284	12,72118	7,265171	8,749848	6,57688	5,212867	45,28746
		min	2,21E−12	8,12E−11	7,28E−12	1,26E−11	5,2E−12	3,07E−11	1,55E−11	6,45E−12	8,22E−12	2,73E−11	8,22E−11
wn			24	29	27	**38**	29	28	30	28	27	26	27

The null hypothesis H0 for this test is: "There is no difference between the medians of the solutions produced by the algorithms CVS3 and VS for the same benchmark problem". The results of the third test were given in Table 4. In the table, there are both the mean, standard derivation and the best solutions of the 30 runs and p-values, the sum of positive ranks (R+) and the sum of negative ranks (R−) that computed for the Wilcoxon Signed-Rank Test. The results of the Wilcoxon Signed-Rank test evaluated as follows: for a pairwise comparison, if the p-value > 0.05 then there is no significant difference between the two compared algorithms such a result indicated with a "=" sign in the w column of the table. On the other hand if the p-value $<= 0.05$ then there is a significant difference between the two algorithms and if the R−> R+ this means CVS3 algorithm shown better performance than the VS algorithm otherwise the VS algorithm shown better performance than the CVS3 algorithm. These two results were also depicted with "+" and "−" signs in the w column of the table, respectively.

The results given in Table 4 can be evaluated from two angles first when the *Maxt* = 10 000 and the second when Maxt = 100 000. For Maxt = 10 000, both of the results of the statistical values (mean, std and best values) and the Wilcoxon Signed-Rank showed the superiority of the CVS3 algorithm over the VS algorithm. At this maximum NOI, CVS3 gets the best mean values of the objective function values for thirty-seven benchmark functions while the VS gets the best results for twenty-six functions. It is also seen that CVS3 outperforms the VS in terms of the results of the Wilcoxon Signed-Rank test; CVS3 shows a significant difference from VS algorithm for twenty-five functions while the VS algorithm shows better results for only five functions. For the other twenty functions, there is no difference between the performances of the both of the algorithms. For *Maxt* = 100 000, it is seen that the performance of the VS algorithms obviously improved. However, when these results compared with the CVS3 algorithm (Maxt = 10 000), it can be seen that the both of the algorithms show nearly the same performances. CVS3 algorithm shows better performance than VS algorithm in terms of the means of the objective function values(CVS3: 37, VS: 34) while the results of the Wilcoxon Signed-Rank test shows that the VS algorithm gets better results for only two more benchmark functions (+ : 11, −: 13, = : 26) than the CVS3 algorithm. As a result, it can be said that using the chaos effect as explained in the proposed two methods improve the performance of the VS algorithm and help to reduce the required maximum NOI that is very important for solving the real-world optimization problems.

So as to present the improvement in the performance of the VS algorithm in terms of the algorithm processing time we performed a final test. In this test, we run the CVS3 and the VS algorithms for Maxt = 1000 and Maxt = 10 000 under the same conditions for solving the 50 benchmark functions and recorded the computation time for all the functions. The other parameters were used as given in the first test. The results of this test were given in Fig. 7.

In the figure that was drawn for the Maxt = 1000, it can be seen that both of the algorithms completed the iterations in very close time periods. As seen from the F26 marked on the figure, computed time for the VS was 0.1654 s while it was 0.1828 s for the CVS3. The difference between these two-time values is 0.0174 s. On the other hand, as Maxt increased to 10 000 the difference was also increased to 0.257 s. Although it can be said that the proposed methods increase the computation time of the VS algorithm slightly, this increment may be ignored by considering the performance improvements provided by these methods. As stated in the previous section, the CVS3 (*Maxt* = 10 000) was shown nearly the same performance as VS (*Maxt* = 100 000). In this respect, it is clear that the computation time of the VS (*Maxt* = 100 000) will be very much longer than the CVS3 (*Maxt* = 10 000). Thus, it can be concluded that the proposed methods help the algorithm to obtain feasible solutions in an acceptable computation time by improving the searching characteristics of the VS algorithm. And, this makes the proposed CVS3 algorithm as a good candidate for solving the real-world optimization problems with its simple formulations, better exploration and exploitation abilities, and tolerable computation time performance.

4. The solution of the inverse kinematics problem of a six DOF serial robot manipulator with offset wrist by the proposed CVS3 algorithm

Inverse kinematics problem is a basic problem to be solved for the serial robot manipulators. While this problem can be solved analytically for some kind of serial manipulators, generally, the solution of this problem cannot be obtained by fully analytical methods especially, for the manipulators those have offset wrists [41]. In this section we solved the inverse kinematics problem of a 6-DOF serial robot manipulator with offset wrist; this manipulator is composed of six revolute joints and has highly nonlinear inverse kinematics equations that cannot be

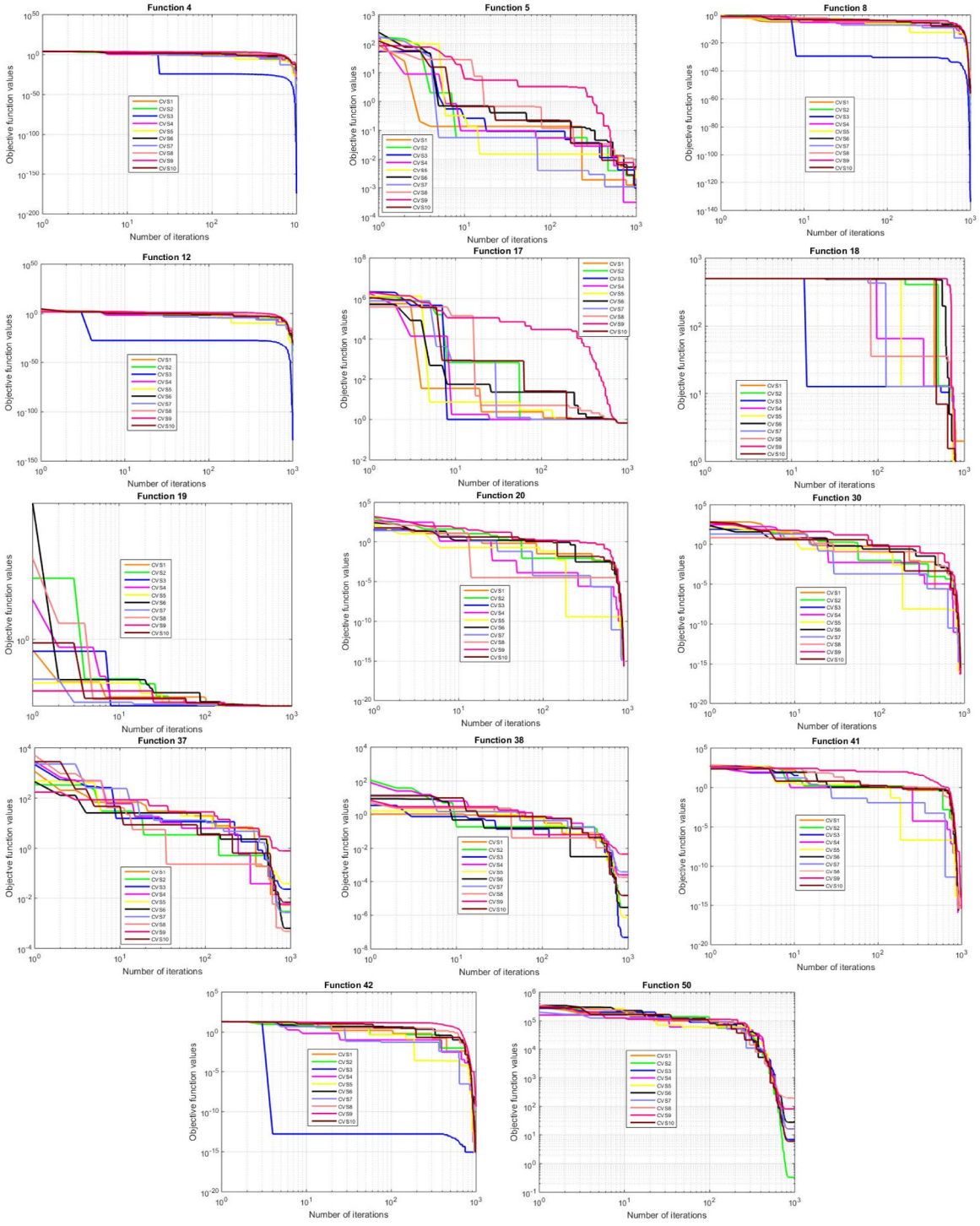

Fig. 4. Convergence graphics of the ten CVS algorithms for solving the selected benchmark functions.

solved analytically. The body structure and the coordinate system replacements for the selected mechanism are given in Fig. 8 [41].

In order to obtain forward and inverse kinematic equations of a serial robot mechanism, DH method [41] can be used. DH method is based on the DH parameters of the serial robot mechanism and these parameters of the selected mechanism are given in Table 5 [41].

Direct kinematics of the mechanism can be obtained by using the homogeneous transformation matrices defined for each joint. These matrices can be written by using the DH parameters of the mechanism and the general homogeneous transformation matrix formula given in Eq. (15) [41].

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15} $$

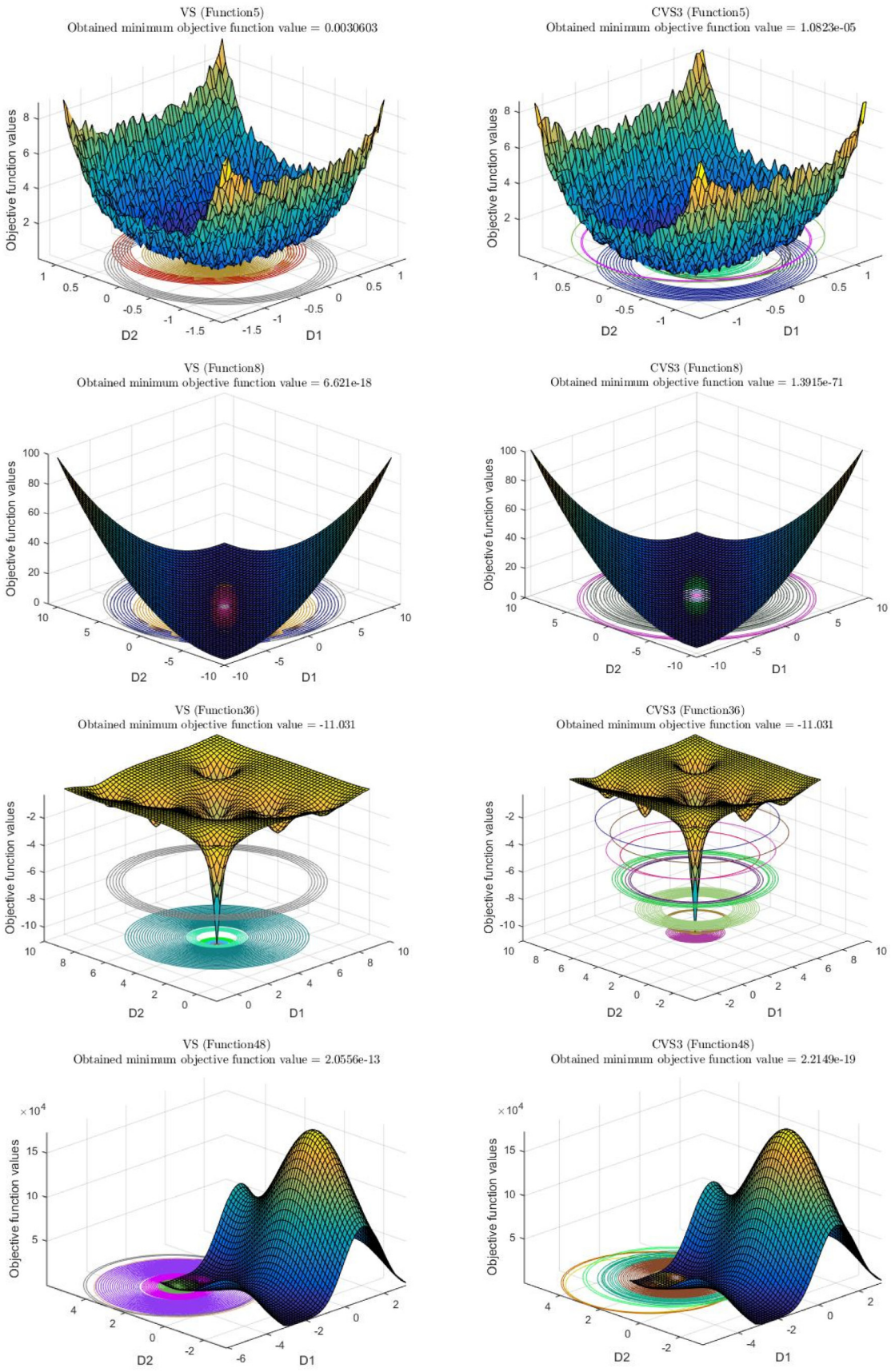

Fig. 5. Effects of the proposed methods on searching characteristic of the VS algorithm.

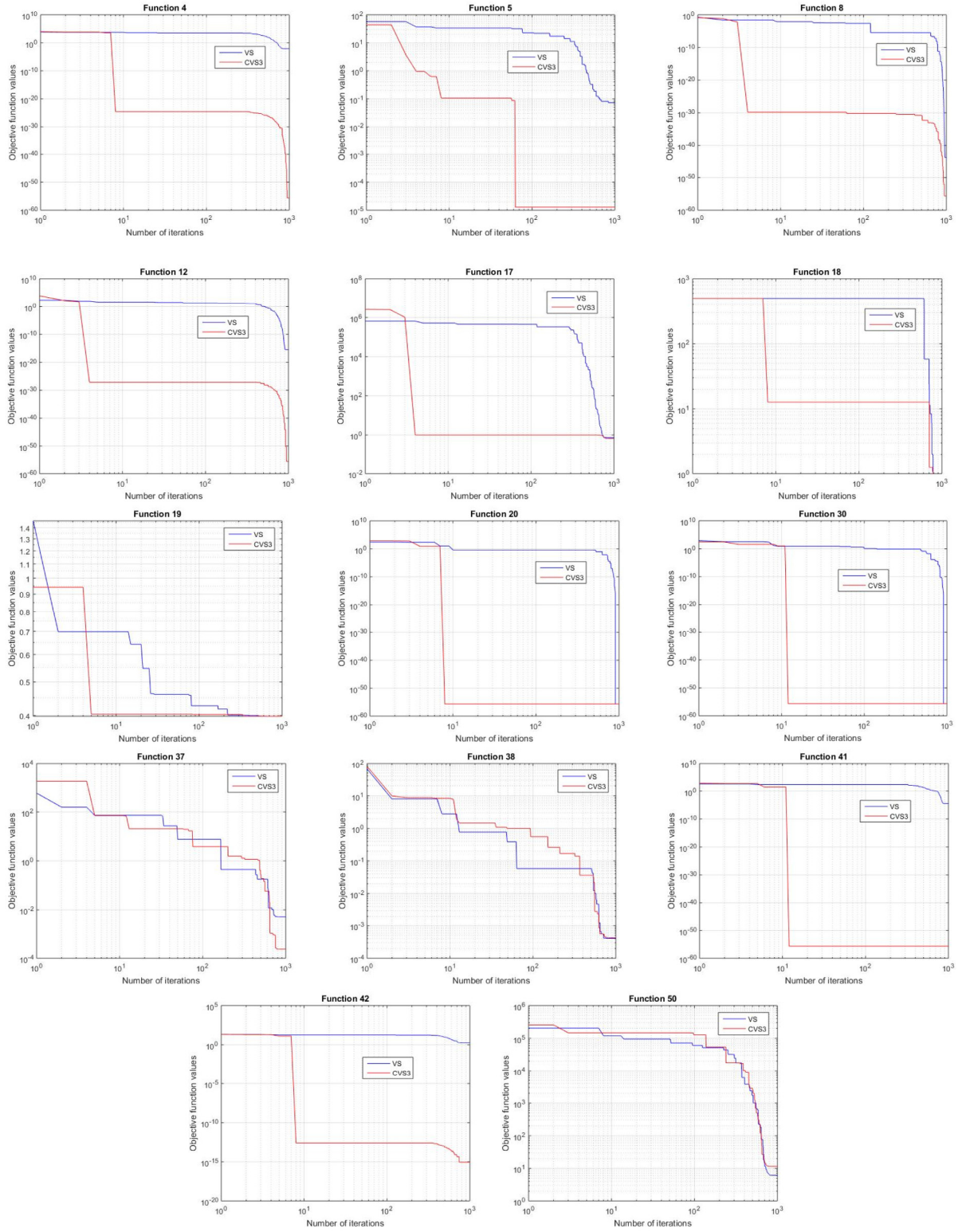

Fig. 6. Graphs of the convergence behaviors of the VS and CVS3 algorithms for the selected benchmark functions.

where ${}^{i-1}_{i}T$ is the homogeneous transformation matrix between $(i-1)$th and ith joints, and c and s indicates cosine and sine functions, respectively. Homogeneous transformation matrices of the selected mechanism are obtained by substituting each row of Table 5, in Eq. (15) as following;

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & h_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

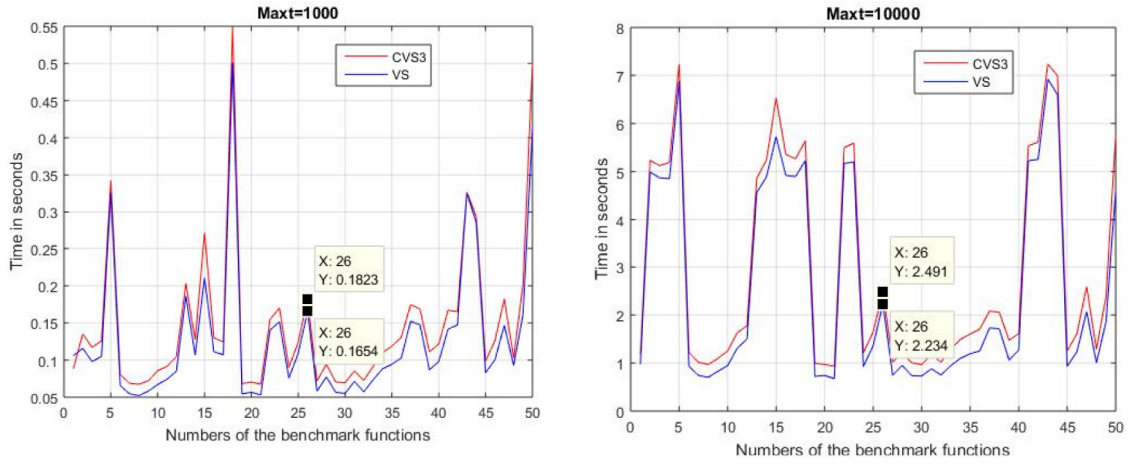

Fig. 7. Computation time values of the CVS3 and the VS algorithms computed for the 50 benchmark functions.

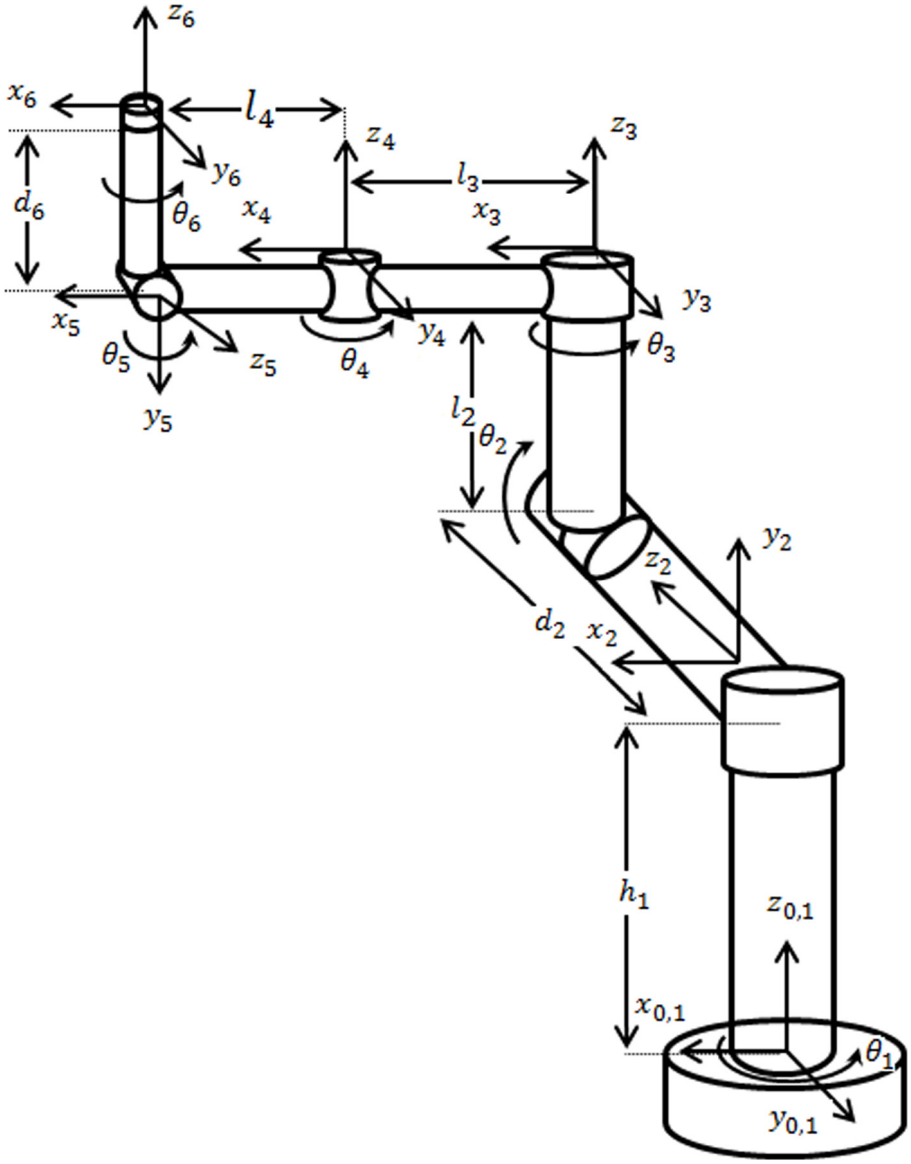

Fig. 8. 6-DOF serial robot manipulator with offset wrist [41].

Table 4
Statistical results of the VS and ten CVS3 algorithms in terms of the maximum NOI for the benchmark functions.

f	*VS* (*Maxt* = 10 000)				*CVS3* (*Maxt* = 10 000)			*VS* (*Maxt* = 100 000)			*VS* (*Maxt* = 10 000) *vs CVS3* (*Maxt* = 10 000)				*VS* (*Maxt* = 100 000) *vs CVS3* (*Maxt* = 10 000)			
	fmin	mean	std	best	mean	std	best	mean	std	best	*p*-value	R+	R−	w	*p*-value	R+	R−	w
1	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
2	0	1,6	1,42876846	0	**0**	0	0	0,4	0,6214555	0	3,23E−05	0	253	+	0,001953	0	55	+
3	0	2,19E−36	3,64E−36	1,93E−38	**0**	0	0	1,24E−120	4,14E−120	2,25E−126	1,73E−06	0	465	+	1,73E−06	0	465	+
4	0	2,59E−15	6,54E−15	3,70E−21	**0**	0	0	2,26E−54	7,07E−54	2,42E−58	1,73E−06	0	465	+	1,73E−06	0	465	+
5	0	0,00587276	0,00322025	0,001625	**0,000137**	9,16E−05	1,88E−05	0,00052706	0,0002572	6,83E−05	1,73E−06	0	465	+	1,92E−06	1	464	+
6	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
7	−1	**−1**	0	−1	**−1**	0	−1	**−1**	0	−1	1	0	0	=	1	0	0	=
8	0	1,93E−119	1,05E−118	2,09E−160	**0**	0	0	**0**	0	0	1,73E−06	0	465	+	1	0	0	=
9	0	**3,25E−16**	8,51E−16	1,11E−24	1,09E−15	4,84E−15	6,36E−22	1,48E−31	2,38E−31	0	0,926255	237	228	=	1,73E−06	465	0	−
10	−50	**−50**	0	−50	**−50**	3,29E−14	−50	**−50**	4,66E−14	−50	7,42E−07	0	465	+	0,027807	187	66	−
11	−210	**−210**	0	−210	**−210**	6,20E−13	−210	**−210**	5,01E−13	−210	2,47E−06	0	378	+	0,000294	213	18	−
12	0	3,10E−51	9,05E−51	3,28E−59	**0**	0	0	1,19E−153	6,49E−153	6,49E−178	1,73E−06	0	465	+	1,73E−06	0	465	+
13	0	0,00702422	0,00121163	0,004002	**0**	0	0	0,00020274	2,98E−05	0,000147569	1,73E−06	0	465	+	1,73E−06	0	465	+
14	0	1,55E−09	5,23E−09	1,71E−15	**3,74E−211**	0	0	6,13E−29	3,17E−28	1,30E−44	1,73E−06	0	465	+	1,73E−06	0	465	+
15	0	2,46E−11	4,81E−11	5,10E−13	**0**	0	0	3,70E−46	1,07E−45	2,53E−51	1,73E−06	0	465	+	1,73E−06	0	465	+
16	0	34,061437	30,2424699	20,01081	25,753	0,208435	25,31584	**20,4409536**	27,506181	6,690464643	0,289477	284	181	=	0,014795	351	114	−
17	0	0,6680625	0,00727598	0,666667	**0,666667**	3,06E−07	0,666667	**0,666667**	1,10E−15	0,666666667	1,02E−05	18	447	+	1,73E−06	465	0	−
18	0,998	**0,99800384**	4,52E−16	9,98E−01	**0,99800384**	1,17E−16	9,98E−01	**0,99800384**	4,12E−17	0,998003838	4,32E−08	0	465	+	1	0	0	=
19	0,398	**0,39788736**	1,13E−16	0,397887	**0,39788736**	0	0,397887	**0,39788736**	0	0,397887358	4,32E−08	0	465	+	1	0	0	=
20	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
21	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
22	0	70,3434074	21,4513161	32,83357	**0**	0	0	63,3455565	17,168301	23,87900729	1,73E−06	0	465	+	1,73E−06	0	465	+
23	−12569,5	−10555,234	609,390195	−11819,4	−10738,2	523,9617	−11523,2	**−10957,97**	508,29771	−11898,33061	0,338856	186	279	=	0,12209	289	146	=
24	−1,8013	**−1,8013034**	6,78E−16	−1,80E+00	**−1,8013034**	9,03E−16	−1,80E+00	**−1,8013034**	9,03E−16	−1,80130341	4,32E−08	0	465	+	1	0	0	=
25	−4,6877	−4,5338515	0,11476821	−4,69E+00	−4,58817	0,127184	−4,69E+00	**−4,6045066**	0,0760107	−4,687658179	0,171139	166	299	=	0,895697	108,5	101,5	=
26	−9,6602	−8,4660224	0,60686707	−9,3295	−8,51652	0,669994	−9,27467	**−8,8428149**	0,4807649	−9,578541716	0,643517	210	255	=	0,028486	339	126	−
27	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
28	−1,03163	**−1,0316285**	0	−1,03E+00	**−1,03163**	6,52E−16	−1,03E+00	**−1,03163**	6,71E−16	−1,031628453	4,32E−08	0	465	+	1	0	0	=
29	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
30	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
31	−186,73	**−186,7309**	8,67E−14	−186,731	**−186,7309**	3,69E−14	−186,731	**−186,7309**	3,66E−14	−186,7309088	4,32E−08	0	465	+	0,0625	15	0	=
32	3	**3**	0	3	**3**	1,40E−15	3	**3**	2,02E−15	3	4,32E−08	0	465	+	0,000488	78	0	−
33	0,00031	**0,00030954**	1,12E−05	3,07E−04	0,000307	5,57E−19	3,07E−04	0,00030749	1,87E−19	0,000307486	1,98E−07	0	465	+	5,57E−05	221	10	−
34	−10,15	**−10,1532**	0	−10,1532	**−10,1532**	6,22E−15	−10,1532	**−10,1532**	6,85E−15	−10,15319968	4,32E−08	465	0	−	1	0	0	=
35	−10,4	**−10,4029**	0	−10,4029	**−10,4029**	8,73E−16	−10,4029	**−10,40294**	1,51E−15	−10,40294057	4,32E−08	465	0	−	1	0	0	=
36	−10,53	**−10,53641**	0	−10,5364	**−10,53641**	9,33E−16	−10,5364	**−10,53641**	1,65E−15	−10,53640982	4,32E−08	465	0	−	1	0	0	=
37	0	**0,00232043**	0,00236815	2,99E−07	0,013838	0,033887	5,38E−06	0,00616177	0,0177615	2,45E−06	0,042767	331	134	−	0,703564	251	214	=
38	0	0,00023229	0,0001517	4,05E−08	0,00016	0,000167	6,66E−11	**5,21E−05**	9,44E−05	4,83E−11	0,085896	149	316	=	0,00439	371	94	−
39	−3,86	**−3,8627821**	2,26E−15	−3,86278	**−3,8627821**	2,61E−15	−3,86278	**−3,8627821**	2,68E−15	−3,862782148	4,32E−08	465	0	−	1	0	0	=
40	−3,32	−3,2826326	0,05715495	−3,32237	−3,27866	0,058427	−3,32237	**−3,2945533**	0,0512804	−3,322368011	0,111728	156	309	=	0,258118	80	40	=
41	0	0,01714747	0,01113782	1,11E−16	**0**	0	0	0,02549282	0,0208377	0	1,71E−06	0	465	+	5,60E−06	0	378	+
42	0	0,09955483	0,38098993	1,51E−14	**8,88E−16**	0	8,88E−16	1,37E−14	3,91E−15	7,99E−15	1,31E−06	0	465	+	6,63E−07	0	465	+
43	0	12,8450552	19,7471439	1,14E−31	**2,61E−31**	2,94E−31	3,64E−32	0,02764493	0,0813707	1,57E−32	3,86E−06	8	457	+	0,057024	325	140	=
44	0	2,13E−32	8,72E−33	6,43E−33	1,72E−32	8,08E−33	6,43E−33	**1,50E−33**	0	1,50E−33	0,071873	145	320	=	1,70E−06	465	0	−
45	−1,08	**−1,0809384**	4,52E−16	−1,08094	**−1,0809384**	4,88E−16	−1,08094	**−1,0809384**	4,52E−16	−1,080938442	4,32E−08	0	465	+	1	0	0	=
46	−1,5	**−1,4821659**	0,09767726	−1,50E+00	**−1,4821659**	0,097677	−1,50E+00	−1,4802659	0,1080838	−1,499999223	3,16E−06	29	436	+	1	1	2	=
47	NA	−0,7347162	0,41350024	−0,27494	−0,79804	0,379136	−0,27866	**−1,1545667**	0,344427	−0,531	0,673255	212	253	=	0,000959	348	58	−
48	0	**0**	0	0	**0**	0	0	**0**	0	0	1	0	0	=	1	0	0	=
49	0	0,00302636	0,01152872	5,05E−29	1,53E−05	8,35E−05	5,05E−29	**1,91E−25**	3,08E−25	5,05E−29	0,991793	232	233	=	0,001181	324	54	−
50	0	4,40511186	14,956854	2,21E−12	3,473492	6,232009	1,26E−11	**0,4564302**	1,8629579	2,02E−27	0,158855	301	164	=	1,73E−06	465	0	−
wn		26			**37**			34			+ : 25, − : 5, = : 20				+ : 11, − : 13, = : 26			

$$
{}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & -d_2 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{2}_{3}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ -s\theta_3 & -c\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{3}_{4}T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & l_3 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}
$$

$$
{}^{4}_{5}T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & l_4 \\ 0 & 0 & 1 & 0 \\ -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{5}_{6}T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & -1 & -d_6 \\ -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Forward kinematics of the mechanism can be obtained by multiplying the homogeneous transformation matrices as follows [41];

$$
{}^{0}_{6}T = {}^{0}_{1}T^{1}_{2}T^{2}_{3}T^{3}_{4}T^{4}_{5}T^{5}_{6}T \tag{17}
$$

where ${}^{0}_{6}T$ is the homogeneous transformation matrix that defines the position and the rotation of the end-effector of the mechanism according to the 0th coordinate system. Therefore, this matrix can be used to find the position and the rotation of the end-effector by using the given joint variables according to the base coordinate system of the mechanism.

$$
{}^{0}_{6}T = \begin{bmatrix} & & & P_x \\ & R_{3\times3} & & P_y \\ & & & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}
$$

where, $\begin{bmatrix} P_x & P_y & P_z \end{bmatrix}^T$ represent the position of the end-effector of the mechanism for the x, y and z axes in the 3D Cartesian coordinate system and $R_{3\times3}$ defines the 3×3 rotation matrix of the end-effector according to the base coordinate system. As explained above, the forward kinematics of a mechanism can be easily obtained by using the joint variables of the mechanism. On the other hand, inverse kinematics that is used to find the joint variables when the position of the end-effector is given are highly nonlinear to solve analytically as they obtained backwardly from ${}^{0}_{6}T$ to ${}^{0}_{1}T$. It is also should be noted that, even if the inverse kinematics problem of a mechanism can be solved analytically, there may be more than one solutions for one position of the end-effector. Therefore, in the literature, there are lots of studies that propose to solve this problem by numerical techniques or by using optimization algorithms. In this study, we solve the inverse kinematics of the 6-DOF serial robot mechanism with offset wrist by both of the proposed CVS3 and the classical VS algorithms in order to test the performance of CVS3 on a real-world optimization problem.

4.1. Problem definition and the objective function

Inverse kinematics problem is used to find the joint variables when the position of the end-effector of the mechanism is given. In order to solve the inverse kinematics problem by an optimization algorithm, it is needed to define the problem as an optimization problem. Let the position of the end-effector of the manipulator is defined as follows;

Table 5
DH parameters of the 6-DOF serial robot manipulator [41].

i	θ_i	α_{i-1} (radyan)	a_{i-1} (unit)	d_i (unit)
1	θ_1	0	0	h_1
2	θ_2	$\pi/2$	0	d_2
3	θ_3	$-\pi/2$	0	l_2
4	θ_4	0	l_3	0
5	θ_5	$-\pi/2$	l_4	0
6	θ_6	$\pi/2$	0	d_6

$$
P_{desired} = \begin{bmatrix} P_x & P_y & P_z \end{bmatrix} \tag{19}
$$

where $P_{desired}$ is the end-effector position that can be reached by solving the inverse kinematics problem. The solution of the problem requires finding six joint variables $(\theta_1, \theta_2, \ldots, \theta_6)$ to move the end-effector to $P_{desired}$. These variables can be used to define the structure of the candidate solutions for the VS and CVS3 algorithms.

$$
C_s = \begin{bmatrix} \theta_{i1}, & \theta_{i2}, \ldots, \theta_{i6} \\ \vdots & \ldots & \vdots \\ \theta_{n1}, & \theta_{n2}, \ldots, \theta_{n6} \end{bmatrix} \tag{20}
$$

where, C_s is the candidate solution matrix that includes one solution in its one row and $i = 1, 2, \ldots, n$ is the number of the candidate solutions. The objective function for the solution of the inverse kinematics problem can be simply defined as the position error between the $P_{desired}$ and the position of the end-effector ($P_{candidate} = \begin{bmatrix} P_{cx} & P_{cy} & P_{cz} \end{bmatrix}$) in the 3D Cartesian coordinate system that found by substituting a candidate solution in the forward kinematics equations of the mechanism. However, inverse kinematics equation generally, is solved not only for a single end-effector position but also for a group of the positions that compose a trajectory that can be used to move the end-effector from a start point to a goal point. Therefore, it is important to find the solutions of the inverse kinematics problem for all the end-effector positions on a trajectory by providing minimum joint movements in order to obtain a smooth move for the end-effector. This can be provided by finding the solution for an end-effector position by taking the joint variables that found for the previous position of the end-effector into account. Therefore, we defined the objective function for both minimizing the error between the desired and the proposed positions of the end-effector and also for minimizing the sum of the joint movements required to move the end-effector from one point of the trajectory to the next one as follows;

$$
fitness = \left(err \times sum\left(abs\left(C_{sj} - F_{sj-1}\right)\right) + norm\left(P_{desired_j} - P_{candidate_j}\right)\right)^2 \tag{21}
$$

where *fitness* is the objective function value of a candidate solution for jth end-effector position on the trajectory. *err* is a constant used to balance the effect of the two portions of the objective function on the *fitness*. *sum* and *abs* are the vector summation and absolute value functions, respectively. C_{sj} is a candidate solution for the jth end-effector position while F_{sj-1} is the best solution obtained for the $(j-1)$th end-effector position. It should be noted that F_{sj-1} for the first point of the trajectory is defined as $F_{s0} = [0\ 0\ 0\ 0\ 0\ 0]$ which defines the zero position of the mechanism.

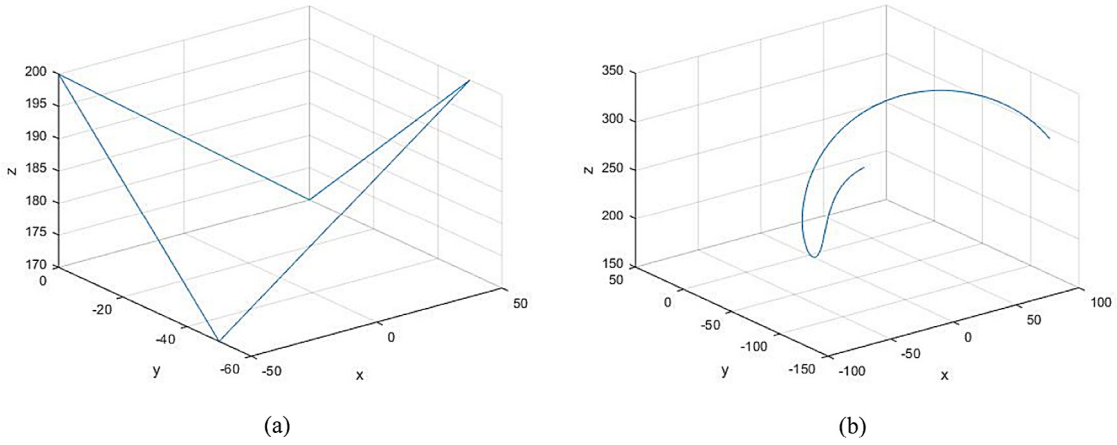

Fig. 9. Linear (a) and curve (b) shaped trajectories in the 3D Cartesian coordinate system.

Table 6
The structural parameters of the mechanism and the parameters that used for the CVS3 and VS algorithms for both of the trajectories.

Algorithms	*population size*	*Maxt*	$N_{ce_{max}}$	$N_{ce_{min}}$	a_0	x	*err*
VS	50	100	***	***	1	0.1	100
CVS3	50	100	0.5	$1E-21$	1	0.1	100
Mechanism structural parameters (mm)		$h_1 = 200,\ d_2 = 120,\ l_2 = 50,\ l_3 = 50,\ l_4 = 50,\ d_6 = 40$					

4.2. Experiments for solving the inverse kinematics problem

In this section, we used two trajectories for testing the performance of the CVS3 and the VS algorithms for solving the inverse kinematics problem of the 6-DOF serial robot manipulator with offset wrist. One of them is a linear-shaped trajectory while the other is a curve-shaped trajectory (Fig. 9a and b). Both of the trajectories are composed of one-hundred numbers of points in the 3D Cartesian coordinate system. The experiments were conducted for both of the trajectories by using both of the algorithms the VS and the CVS3 with the parameters given in Table 6.

As the main purpose of this study is reducing the maximum NOI required to solve real-world problems by the VS algorithm, in this experiments we defined the *Maxt* to be only 100 for both of the algorithms. Objective function values, the position errors between the points of the desired and the resultant trajectories that found by the algorithms and the computation times were recorded. These experiments were performed by a computer equipped with 2.7 GHz CPU and 8 GB RAM. The obtained best objective function values were drawn for the linear-shaped and the curve-shape trajectories and given in Figs. 10a and 10b, respectively.

According to Fig. 10, it can be seen that CVS3 obtained smaller objective function values for both of the trajectories. In order to compare the resultant trajectories found by the algorithms with the desired trajectories; all the trajectories were drawn together in Fig. 11a and b.

From Fig. 11a and b, two results can be concluded, firstly it is seen that the proposed objective function (Eq. (21)) for the solutions of the inverse kinematics problem can be effectively applied for the serial robot mechanisms. And secondly, from both of the figures, it is seen that the CVS3 algorithm obtained nearly same results with the desired trajectories while VS algorithm can produce erroneous results for some points, especially for the curve shaped trajectory. Graphs of the position errors can show more clearly the performance of the algorithms in terms of the following desired trajectories (Fig. 12).

Fig. 12 shows that the CVS3 algorithm has achieved near-zero errors for the curve-shaped trajectory and obtained smaller errors than 1 mm only for a few points of the linear-shaped trajectory. On the other hand, it is seen that the VS algorithm get worse results than the CVS3 in terms of the both of the number and the amplitude of the position errors for both of the trajectories. Since this study aims to reduce the maximum NOI for the VS algorithm and so also the algorithm computation time, the computation times of the algorithms for solving the inverse kinematics problem were given in Fig. 13 as a final evaluation.

According to Fig. 13a and b, it can be seen that both of the algorithms reached the maximum NOI at smaller time values (between 0.2 and 0.37 s) for both of the trajectories. Therefore, it can be said that the proposed methods did not cause a significant time delay for the VS algorithm while improving its performance.

5. Discussion of the results

In this section, we presented a discussion for evaluating the overall results of the proposed methods on the searching characteristics of the VS algorithm. Firstly, it is needed to show if the chaos-based proposed methods improved the performance of the VS algorithm without consideration of the differences between the chaos maps. In this respect, the results given in Table 3 show that all the proposed CVS algorithms performed better than the VS algorithm for all the four functions groups. According to the results, all the CVS algorithms obtained the best performance at least for twenty-six benchmark functions while the VS algorithm obtained the best results for twenty-four benchmark functions. Thus, it is clear that the proposed methods improved the performance of the VS algorithm. And, this result is in parallel with the literature about using chaos theory with the meta-heuristics algorithms such as the studies given in [28,31,35–39]. In the second, in this study, the best chaos map was determined according to the results given in the third section. And, it was found that the CVS3 obtained by Gauss–Mouse map shown the best performance. In this point, a more detailed analysis is required to present the superiority of the Gauss–Mouse map over the other chaos maps

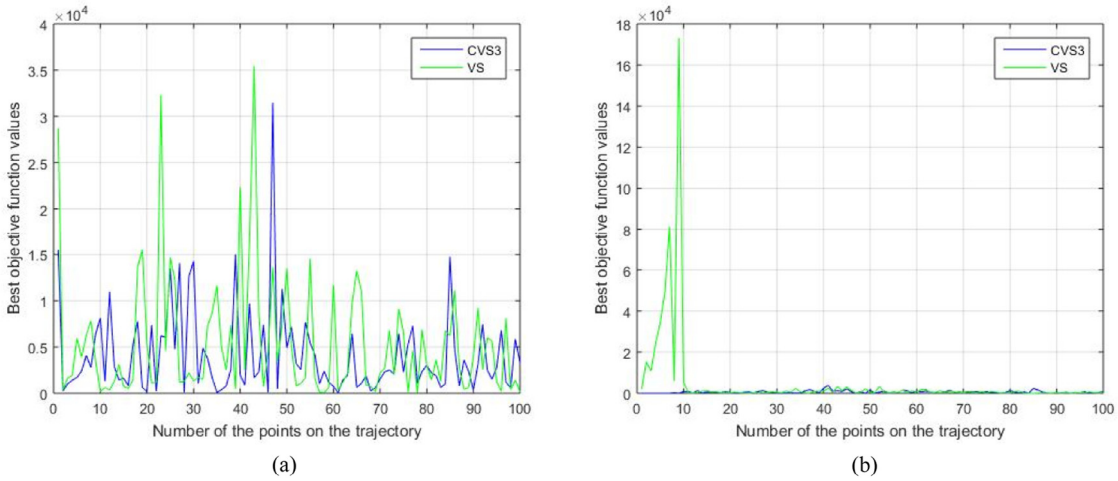

Fig. 10. Best objective function values obtained for (a) Linear-shaped and (b) Curve-shaped trajectories.

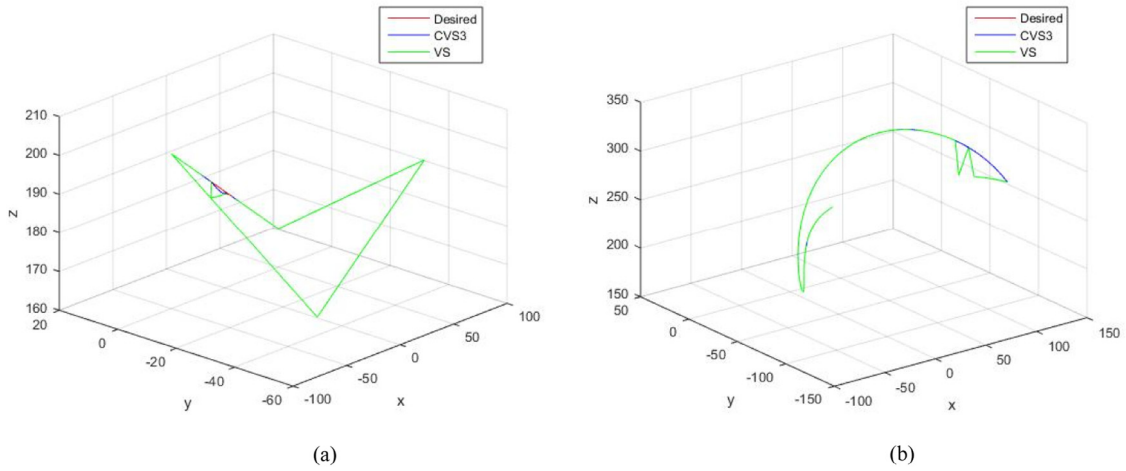

Fig. 11. The resultant trajectories (a) Linear-shaped and (b) Curve-shaped trajectories.

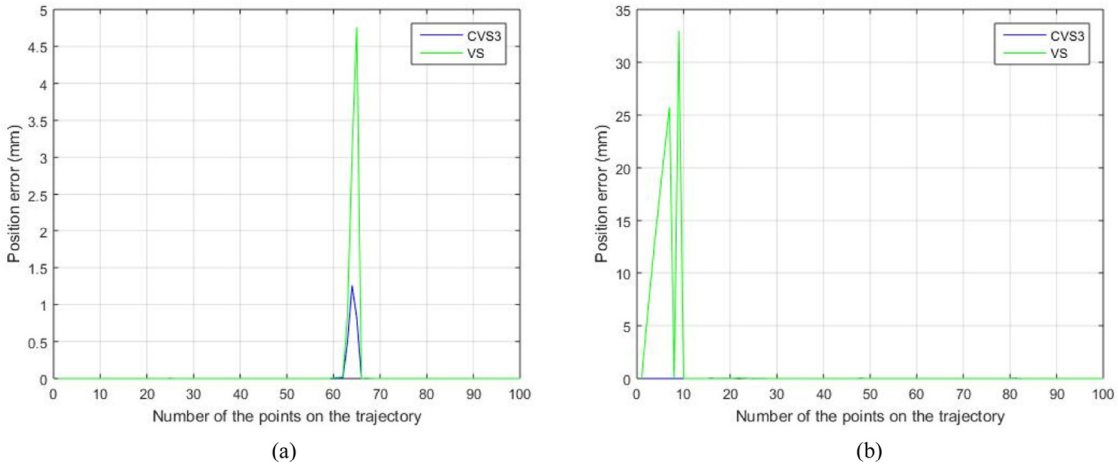

Fig. 12. Position errors between the resultant trajectories and the desired trajectories for both of the algorithms (a) Linear-shaped trajectory and (b) Curve-shaped trajectory.

in terms of the proposed methods. For this aim, we can look at the two studies from the literature, the first one is the study by Sheikholeslami and Kaveh [32]. In that study, the authors reviewed the chaos embedded meta-heuristic optimization algorithms. They divided the studies conducted for improving the performance of the meta-heuristic algorithms using chaos in the literature into two types. The first type is about the studies that use the chaos instead of the random number generators of the algorithms while the second type includes the algorithms that incorporate the chaotic search into their searching characteristics to improve

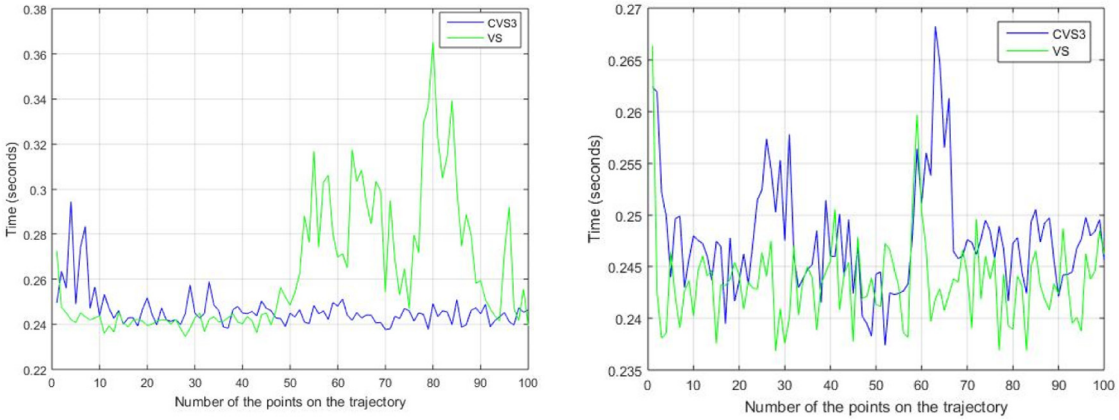

Fig. 13. Computation times of the algorithms for both of the trajectories (a) Linear-shaped and (b) Curve-shaped trajectories.

their performance. The results of that study indicate the positive effects of using the chaos with the meta-heuristic algorithms on their performances. On the other hand, the study also points out the difficulty in saying which chaotic map performs the best for any algorithm. Thus, in this study, it is also not easy to clearly present all the reasons behind the success of the Gause/mouse map over the other chaos maps. However, some important points about this subject can be explained as follows; firstly, the two methods proposed in this study try to improve the exploration and exploitation capabilities of the VS algorithm by adding the chaos effect to the searching behaviors of the algorithm. Hence, we can say that this study belongs to the second group defined by Sheikholeslami and Kaveh [32]. After this point, we can continue the evaluation with the work by Mozaffari et al. [19]. In their study, the authors investigated the effect of using the chaos maps instead of the conventional stochastic random procedure of the evolutionary algorithms and presented a very detailed analysis of this subject. The results of the study were shown that; in terms of using the chaotic crossover and mutation operators, the Gauss–Mouse and logistic maps had shown the best performance for solving the considered 50D benchmark problems [19]. Moreover, in terms of the robustness metric, the authors stated that the circle, Gauss–Mouse and sinusoidal maps can afford much more promising outcomes [19]. The authors also expressed that the logistic, sinusoidal, and Gauss–Mouse chaotic maps were both able to effectively improve the performance of the evolutionary algorithms and can generate random deterministic chaotic sequences with a low computational budget [19]. As explained in the second section of this study, the proposed two methods provide adding the chaos effect to the searching characteristics of the VS algorithm in similar manners as explained in the study given in [19]. And, the obtained results indicate the superior performance of the Gauss–Mouse map over the other chaos maps. This is also in parallel with the study by Mozaffari et al. [19]. Thus, according to the results obtained in this study, the CVS3 algorithm obtained by the Gauss–Mouse was selected as the best CVS algorithm to solve the inverse kinematics problem of the 6-DOF serial robot manipulator with offset wrist. In addition, the experiments performed for solving the inverse kinematics problem shown that there is a significant difference in favor of the CVS3 against the VS algorithm. As a result, it can be said that the proposed methods help to improve the performance of the VS algorithm and made it be more useful in solving real-world optimization problems.

6. Conclusion

In order to reduce the maximum NOI of the VS algorithm, in this study ten chaos maps were applied to the algorithm through two proposed methods and ten chaos based VS algorithms were obtained. The best CVS algorithm was determined by solving fifty benchmark functions and evaluating the results in terms of some statistical values and the Wilcoxon Signed-Rank Test. According to results, it was found that the CVS3 algorithm that obtained by composing the VS algorithm with the Gauss–Mouse chaos map was the best CVS algorithm. And, it was shown that the CVS3 algorithm outperforms the classical VS algorithm even when it uses ten times less the maximum NOI than the VS algorithm. Additionally, four two-dimensional benchmark functions were solved with both the VS and the CVS3 algorithms in order to visually show the differences between searching characteristics of the two algorithms. And, a comparison was performed about algorithm processing time. Moreover, inverse kinematics problem of a 6-DOF serial robot manipulator with offset wrist was selected as a real world optimization problem and solved with the CVS3 and the VS algorithms for two different types of trajectories. The results showed that the CVS3 algorithm shown better performance in terms of the objective function values and the position errors while maintaining nearly the same computation time with the VS algorithm.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106074.

CRediT authorship contribution statement

Metin Toz: Investigation, Visualization, Writing - original draft, Writing - review & editing.

References

[1] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.

[2] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science (80-.) (1983) http://dx.doi.org/10.1126/science.220.4598.671.

[3] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Inf. Sci. (Ny). 179 (2009) 2232–2248, http://dx.doi.org/10.1016/j.ins.2009.03.004.

[4] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–72, http://dx.doi.org/10.1038/scientificamerican0792-66.

[5] J.R. Koza, R. Poli, Genetic programming, in: Search Methodol. Introd. Tutorials Optim. Decis. Support Tech., Springer US, 2005, pp. 127–164, http://dx.doi.org/10.1007/0-387-28356-0_5.

[6] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713, http://dx.doi.org/10.1109/TEVC.2008.919004.

[7] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. ICNN'95 - Int. Conf. Neural Networks, IEEE, 1995, pp. 1942–1948.

[8] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/J.ADVENGSOFT.2013.12.007.

[9] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (2006) 28–39, http://dx.doi.org/10.1109/MCI.2006.329691.

[10] F. Glover, Tabu search—Part I, ORSA J. Comput. 1 (1989) 190–206, http://dx.doi.org/10.1287/ijoc.1.3.190.

[11] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (1997) 1097–1100, http://dx.doi.org/10.1016/S0305-0548(97)00031-2.

[12] B. Doğan, T. Ölmez, A new metaheuristic for numerical function optimization: Vortex search algorithm, Inf. Sci. (Ny). 293 (2015) 125–145, http://dx.doi.org/10.1016/j.ins.2014.08.053.

[13] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. (Ny). 237 (2013) 82–117, http://dx.doi.org/10.1016/j.ins.2013.02.041.

[14] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471, http://dx.doi.org/10.1007/s10898-007-9149-x.

[15] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Adv. Eng. Softw. 95 (2016) 51–67, http://dx.doi.org/10.1016/J.ADVENGSOFT.2016.01.008.

[16] X.S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Inspired Comput. 2 (2010) 78–84, http://dx.doi.org/10.1504/IJBIC.2010.032124.

[17] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, O. Maciel-Castillo, I. Aranguren, F. Fausto, An improved simulated annealing algorithm based on ancient metallurgy techniques, Appl. Soft Comput. 84 (2019) 105761, http://dx.doi.org/10.1016/J.ASOC.2019.105761.

[18] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: A literature review, Swarm Evol. Comput. 2 (2012) 1–14, http://dx.doi.org/10.1016/j.swevo.2011.11.003.

[19] A. Mozaffari, M. Emami, A. Fathi, A comprehensive investigation into the performance, robustness, scalability and convergence of chaos-enhanced evolutionary algorithms with boundary constraints, Artif. Intell. Rev. 52 (2019) 2319–2380, http://dx.doi.org/10.1007/s10462-018-9616-4.

[20] U. Ozkaya, L. Seyfi, A comparative study on parameters of leaf-shaped patch antenna using hybrid artificial intelligence network models, Neural Comput. Appl. 29 (2016) 35–45, http://dx.doi.org/10.1007/s00521-016-2620-1.

[21] E. García, I. Amaya, R. Correa, Estimation of thermal properties of a solid sample during a microwave heating process, Appl. Therm. Eng. 129 (2018) 587–595, http://dx.doi.org/10.1016/j.applthermaleng.2017.10.037.

[22] B. Doğan, T. Ölmez, Vortex search algorithm for the analog active filter component selection problem, AEU - Int. J. Electron. Commun. 69 (2015) 1243–1253, http://dx.doi.org/10.1016/j.aeue.2015.05.005.

[23] W. Ali, M.A. Qyyum, K. Qadeer, M. Lee, Energy optimization for single mixed refrigerant natural gas liquefaction process using the metaheuristic vortex search algorithm, Appl. Therm. Eng. 129 (2018) 782–791, http://dx.doi.org/10.1016/j.applthermaleng.2017.10.078.

[24] M. Saka, I. Eke, S.S. Tezcan, M.C. Taplamacioglu, Economic load dispatch using vortex search algorithm, in: 2017 4th Int. Conf. Electr. Electron. Eng., IEEE, Ankara, 2017, pp. 77–81.

[25] M. Rajeswari, J. Amudhavel, P. Dhavachelvan, Vortex search algorithm for solving set covering problem in wireless sensor network, Adv. Appl. Math. Sci. 17 (2017) 95–111.

[26] O. Aydin, S.S. Tezcan, I. Eke, M.C. Taplamacioglu, Solving the optimal power flow quadratic cost functions using vortex search algorithm, IFAC-PapersOnLine 50 (2017) 239–244, http://dx.doi.org/10.1016/j.ifacol.2017.08.040.

[27] B. Do, T. Ölmez, Fuzzy clustering of ECG beats using a new metaheuristic approach, in: 2nd Int. Work. Bioinforma. Biomed. Eng., Granada, 2014, pp. 54–65.

[28] X. Li, P. Niu, J. Liu, Combustion optimization of a boiler based on the chaos and Lévy flight vortex search algorithm, Appl. Math. Model. 58 (2018) 3–18, http://dx.doi.org/10.1016/j.apm.2018.01.043.

[29] Y. Huang, G. Hou, X. Cheng, B. Feng, L. Gao, M. Xiao, A new vortex search algorithm with gradient-based approximation for optimization of the fore part of KCS container ship, J. Mar. Sci. Technol. 22 (2017) 403–413, http://dx.doi.org/10.1007/s00773-016-0419-5.

[30] H. Sajedi, S.F. Razavi, MVSA: multiple vortex search algortihm, in: 17th IEEE Int. Symp. Comput. Intell. Informatics, Budapest, Hungary, 2016, pp. 169–174.

[31] B. Alatas, E. Akin, A.B. Ozer, Chaos embedded particle swarm optimization algorithms, Chaos Solitons Fractals 40 (2009) 1715–1734, http://dx.doi.org/10.1016/J.CHAOS.2007.09.063.

[32] R. Sheikholeslami, A. Kaveh, A survey of chaos embedded metaheuristic algorithms, Iran Univ. Sci. Technol. 3 (2013) 617–633, http://ijoce.iust.ac.ir/article-1-153-en.html&sw=A+Survey+of+Chaos+Embedded+Meta-Heuristic. (Accessed 6 July 2018).

[33] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao, H. Wang, Chaotic Krill Herd algorithm, Inf. Sci. (Ny). 274 (2014) 17–34, http://dx.doi.org/10.1016/J.INS.2014.02.123.

[34] R. Tang, S. Fong, R.K. Wong, K.K.L. Wong, Dynamic group optimization algorithm with embedded chaos, IEEE Access 6 (2018) 22728–22743, http://dx.doi.org/10.1109/ACCESS.2017.2724073.

[35] S. Saha, V. Mukherjee, A novel chaos-integrated symbiotic organisms search algorithm for global optimization, Soft Comput. 22 (2017) 1–20, http://dx.doi.org/10.1007/s00500-017-2597-4.

[36] M. Metlicka, D. Davendra, Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems, Swarm Evol. Comput. 25 (2015) 15–28, http://dx.doi.org/10.1016/j.swevo.2015.03.002.

[37] S.K. Nie, Y.J. Wang, S. Xiao, Z. Liu, An adaptive chaos particle swarm optimization for tuning parameters of PID controller, Optim. Control Appl. Methods 38 (2017) 1091–1102, http://dx.doi.org/10.1002/oca.2314.

[38] S. Hinojosa, D. Oliva, E. Cuevas, G. Pajares, O. Avalos, J. Gálvez, Improving multi-criterion optimization with chaos: a novel Multi-Objective Chaotic Crow Search Algorithm, Neural Comput. Appl. 29 (2018) 319–335, http://dx.doi.org/10.1007/s00521-017-3251-x.

[39] C. Rim, S. Piao, G. Li, U. Pak, A niching chaos optimization algorithm for multimodal optimization, Soft Comput. 22 (2018) 621–633, http://dx.doi.org/10.1007/s00500-016-2360-2.

[40] Y. Shen, Improved chaos genetic algorithm based state of charge determination for lithium batteries in electric vehicles, Energy 152 (2018) 576–585, http://dx.doi.org/10.1016/J.ENERGY.2018.03.174.

[41] S. Kucuk, Z. Bingul, Inverse kinematics solutions for industrial robot manipulators with offset wrists, Appl. Math. Model. 38 (2014) 1983–1999, http://dx.doi.org/10.1016/J.APM.2013.10.014.

[42] S. Kucuk, Modelling and Off-Line Programming of Industrial Robots, Kocaeli University, 2004.

[43] S. Mirjalili, A.H. Gandomi, Chaotic gravitational constants for the gravitational search algorithm, Appl. Soft Comput. 53 (2017) 407–419, http://dx.doi.org/10.1016/J.ASOC.2017.01.008.