Computing and Informatics, Vol. 38, 2019, 1444–1478, doi: 10.31577/cai_2019_5_1444

CHAOTIC ELECTION ALGORITHM

Hojjat Emami

Computer Engineering Department, University of Bonab
Bonab, East Azerbaijan, Iran
e-mail: emami@ubonab.ac.ir

Abstract. A novel Chaotic Election Algorithm (CEA) is presented for numerical function optimization. CEA is a powerful enhancement of election algorithm. The election algorithm is a socio-politically inspired strategy that mimics the behavior of candidates and voters in presidential election process. In election algorithm, individuals are organized as electoral parties. Advertising campaign forms the basis of the algorithm in which individuals interact or compete with one other using three operators: positive advertisement, negative advertisement and coalition. Advertising campaign hopefully causes the individuals converge to the global optimum point in solution space. However, election algorithm suffers from a fundamental challenge: it gets stuck at local optima due to the inability of advertising campaign in searching solution space. CEA enhances the election algorithm through modifying party formation step, introducing chaotic positive advertisement and migration operator. By chaotic positive advertisement, CEA exploits the entire solution space, what increases the probability of obtaining global optimum point. By migration, CEA increases the diversity of the population and prevents early convergence of the individuals. The proposed CEA algorithm is tested on 28 well-known standard boundary-constrained test functions, and the results are verified by a comparative study with several well-known meta-heuristics. The results demonstrate that CEA is able to provide significant improvement over canonical election algorithm and other comparable algorithms.

Keywords: Optimization, meta-heuristic, election algorithm, Chaotic Election Algorithm (CEA)

Mathematics Subject Classification 2010: 68T01, 68T20, 68T05, 68W25

1 INTRODUCTION

Optimization is the process of finding the best solution from among the set of all feasible solutions subject to a given set of constraints. An optimization problem can be represented as a minimization (maximization) model with the goal to obtain a point x^* from a solution space $S \in R^n$, where objective function $f : S \rightarrow R$ is minimized, i.e. $f(x^*) \leq f(x)$ for all $x \in S$. In recent years, several meta-heuristics have been presented to solve optimization problems. A bibliography of recently proposed meta-heuristics is given in [1], and several surveys are given in [2, 3, 4, 5, 6, 7]. Generally speaking, meta-heuristics can be classified into three main categories [53]: evolutionary, swarm intelligence and physics-based algorithms.

Evolutionary meta-heuristics are mainly inspired by the concepts of natural biological evolution, in which the fittest individuals can survive and the weak must die [8]. In natural evolution survival is achieved through reproduction. Evolutionary algorithms begin their optimization process with a randomly generated population of individuals, where any individual is a candidate solution for the given problem. For each generation, individuals compete with each other to reproduce offspring. The best-fit individuals have the best chance to reproduce. Offspring are generated by the combination and mutation of the individuals in the previous generation. The offspring iteratively update over the course of generations until an optimal solution is reached. Some of the well-known evolutionary algorithms are Genetic Algorithm (GA) [9], Differential Evolution (DE) [35], Biogeography-Based Optimizer (BBO) [42] and Backtracking Search Optimization Algorithm (BSA) [10].

The second main branch of meta-heuristics is swarm intelligence algorithms. These algorithms are inspired by natural or non-natural phenomena and mostly mimic the social behavior of swarms and social organisms [53, 8]. For example, Artificial Bee Colony (ABC) [11] is a nature inspired algorithm, which models intelligent behavior of honey bees in nature. Another example is election algorithm [12], a non-natural inspired algorithm, which simulates candidates' behavior in a presidential election process. Swarm intelligence based algorithms are multi-agent models. These algorithms model the intelligent behaviors of agents and their local interaction with the environment and neighboring agents to explore solution space and reach global optima. Some of the well-known swarm intelligence algorithms include: Particle Swarm Optimization (PSO) [13], Ant Colony Optimization (ACO) [37], ABC [11], Election algorithm [12], Firefly Algorithm (FA) [44], Grey Wolf Optimizer (GWO) [53] and Salp Swarm Algorithm (SSA) [55].

The third class of meta-heuristics is physics-based methods, which almost mimic the physical processes of nature. For example, Big-Bang Big-Crunch (BB-BC) [28] is inspired by the evolution of universe; and Gravitational Search Algorithm (GSA) [43] is developed based on gravity law. Some other well-known algorithms that fall into the category of physics-based meta-heuristics include: Intelligent Water Drops (IWD) [15], Charged System Search (CSS) [16], Black Hole (BH) [17] and Magnetic Optimization Algorithm (MOA) [18]. For a survey of physics-based algorithms see [19].

Meta-heuristics are widely used in various scientific and engineering applications because they have shown good performance in solving large-scale, complex non-linear and non-differentiable problems. The applications range from data mining [20], image processing [21] and social network analysis [22] in computer science domain, at one end of the spectrum, to air traffic control [23], airfoil design [55], optical buffer design [53] in industrial field, at the other side of spectrum. However, according to the famous "No Free Lunch" theory [24], there is no meta-heuristic best suited for solving all optimization problems. A particular meta-heuristic may show promising results on a set of problems, but the same algorithm may show poor results on a different set of problems. On the other hand, meta-heuristics achieved encouraging results on optimization problems but their performance far from the ideal. According to this issue and the NFL theory, it is obvious that there is still a room for introducing new meta-heuristics or improving existing meta-heuristics.

As an element of research in this field, this paper presents a new Chaotic Election Algorithm, denoted as CEA. The CEA enhances the canonical election algorithm threefold:

1. increasing the speed of party formation step employing random initialization method,
2. introducing migration operator to enhance the diversity of population and preventing early convergence of the algorithm, and
3. introducing chaotic positive advertisement operator to searching efficiently the entire solution space.

The CEA algorithm is tested on 28 test problems and compared with several well-known meta-heuristics. The experimental results show that the proposed algorithm outperforms counterpart meta-heuristics for several benchmark test functions.

The rest of the paper is organized as follows. Section 2 presents related work, with the focus on chaotic swarm optimization algorithms. Section 3 presents the canonical election algorithm. Section 4 outlines the proposed Chaotic Election Algorithm (CEA). In Section 5, the proposed algorithm is tested on numerical optimization benchmark problems and the simulation results are compared with several well-known algorithms. Finally, Section 6 presents a conclusion of this work and suggests some directions for future work.

2 RELATED WORK

Most of the meta-heuristic algorithms suffer from stagnation in local optima and low convergence rate. With the development of the nonlinear dynamics, chaos theory has been widely used in various applications [29]. One of the major applications is the introduction of chaos concept into the optimization meta-heuristics. Chaos mechanism is one of the best methods to improve the performance of evolutionary algorithms in terms of both local optima avoidance and convergence speed [32]. Due to the ergodicity and randomness nature, chaos has several advantages that include self-organization, evolution, easy implementation and high ability to avoid being trapped in local optima [34, 41]. Due to these properties, simultaneous use of chaos and optimization algorithms improves the performance of algorithms. Up to now, the chaos theory has been successfully combined with several meta-heuristic optimization methods [41]. Table 1 lists some familiar meta-heuristics and their improved ones by incorporated chaos. It is important to notice that Table 1 is not aiming to summarize a comprehensive survey of such chaotic combination, but to show that utilizing chaotic mechanisms indeed empowers the algorithm to possess better performance. This issue highlights that there is an interesting room to combine other meta-heuristics with chaotic mechanism to improve their performance.

Algorithm	Reference	
	Canonical Version	Chaotic Version
Differential Evolution	[35]	[36]
Ant Colony Optimization	[37]	[38]
Artificial Bee Colony Algorithm	[39]	[31]
Imperialist Competitive Algorithm	[40]	[41]
Biogeography-Based Optimization	[42]	[32]
Gravitational Search Algorithm	[43]	[30]
Bat Swarm Optimization	[44]	[45]
Cuckoo Search Algorithm	[46]	[47]
Firefly Algorithm	[48]	[49]
Particle Swarm Optimization	[50]	[51]
Krill Herd Algorithm	[52]	[29]
Grey Wolf Optimizer	[53]	[34, 54]
Salp Swarm Algorithm	[55]	[56]

Table 1. Some meta-heuristics and their corresponding chaotic meta-heuristics

Jia et al. [36] proposed DECLS algorithm to enhance the search ability of Differential Evolution (DE). DECLS explores a huge search space in the early run phases to avoid premature convergence, and exploiting a small region in the later run phases to refine the final solutions. Cai et al. [38] proposed Chaotic Ant Swarm Optimization (CASO) algorithm for solving the economic dispatch problems of thermal generators in power systems. CASO combines the chaotic and swarm-based search capability of ants in searching the global optimum solution.

Alatas [31] proposed Chaotic ABC (CABC) algorithm that adopts chaotic maps for parameter adaptation to prevent the ABC to get stuck on local optima and to improve its convergence speed. This is done by using of chaotic number generators each time a random number is needed by the canonical ABC algorithm.

Talatahari et al. [41] proposed a Chaotic Imperialist Competitive Algorithm (CICA). They used different chaotic maps to improve the assimilation phase of the algorithm. The results on four benchmark problems show the benefits of using chaotic maps in assimilation phase. Saremi et al. [32] investigated the effectiveness of ten different chaotic maps in solving the entrapment in local optima and slow convergence speed problems of the BBO algorithm. They used chaotic maps to define selection, emigration, and mutation probabilities. The experiments show that the chaotic maps are able to improve the performance of BBO.

Gao et al. [30] proposed Chaotic Gravitation Search Algorithms (CGSA) to alleviate the slow convergence and local optima trapping problems of GSA algorithm. The big problem in the canonical Bat Swarm Optimization (BSO) is the premature convergence into local optima. To alleviate this issue, Rezaee [45] presented the CBSO algorithm, which is a chaotic-based bat swarm optimization algorithm. In CBSO, the loudness is updated via multiplying a linearly decreasing function by chaotic map functions.

Wang et al. [47] proposed Chaotic Cuckoo Search (CCS) that embeds chaotic mechanisms into Cuckoo Search (CS) algorithm. In CCS, twelve chaotic maps are applied to tune the step size of the cuckoos used in the original CS algorithm. The experiments on optimization benchmark problems show that the performance of CCS is much better than canonical CS algorithm.

Gandomi et al. [49] proposed Chaotic Firefly Algorithm (CFA) algorithm that incorporated chaos into FA so as to increase its global search mobility. They used twelve different chaotic maps to tune the attractive movement of the fireflies in the algorithm. The experiments show that CFA outperforms the canonical FA.

Alatas et al. [51] proposed twelve different Chaos Embedded Particle Swarm Optimization Algorithms (CEPSOAs) that use chaotic maps for parameter adaptation. CEPSOAs use chaotic number generators each time a random number is needed by the canonical PSO algorithm. The results on benchmark problems show that CEPSOAs increased the solution quality and improved the global searching capability by escaping the local optimum points.

Yaghoobi and Mojallali proposed an Improved Chaotic Krill Herd (ICKH) algorithm used for PID controller design [57]. The main idea of the ICKH is to combine chaos theory and Krill Herd (KH) algorithm to improve the search efficiency.

Yu et al. [34] incorporated chaotic local search mechanism to enhance the search dynamics of GWO algorithm and accelerating it convergence speed. They investigated twelve different kinds of chaotic maps to identify the influence of chaotic search capability on GWO. The results show that chaotic empowers GWO to achieve better performance in terms of solution quality and convergence speed. In another work, Kohli and Arora [54] proposed CGWO algorithm that uses different chaotic maps to regulate the key parameter "a" of GWO algorithm, with the aim of accelerating its convergence speed. The results show the superiority of CGWO when compared to GWO algorithm.

In order to boost the performance of the canonical SSA, Sayed et al. [56] proposed Chaotic Salp Swarm Algorithm (CSSA) that is a hybrid solution based on SSA algorithm and chaos theory. They evaluated ten chaotic maps and found that logistic chaotic map is the optimal map of the used ten maps. The simulation results on optimization benchmarks and feature selection problem reveal the superiority of CSSA algorithm when compared to canonical SSA and some other counterparts.

After this short review, and from the experimental studies presented in the above-mentioned literature, it is obvious that utilizing chaotic mechanisms indeed empowers the algorithm to get better results. This issue highlights that there is an interesting room to combine other meta-heuristics with chaotic mechanism to improve their performance.

3 ELECTION ALGORITHM

3.1 General Aspects

The election algorithm simulates the socio-political process of presidential election in real world [12]. It is a multi-agent algorithm, in which agents are called "persons". There are two types of persons: candidates and voters. Some of the best persons are selected to be the candidates and the remaining are the voters. Initially, all the voters are divided among the candidates based on their similarity in opinions and ideas. Candidates together with their voters form some political parties.

Once initial parties are formed, the candidates start their advertising campaign. Candidates to advertise themselves employ two kinds of advertisements: positive advertisement and negative advertisement. In positive advertisement, candidates convey their agendas and ideas to the voters and attempt to attract the voters towards themselves. In negative advertisement, candidates attempt to increase their own popularity and decrease the popularity of other candidates. Any candidate that is not able to succeed in negative advertisement and cannot increase his popularity will be eliminated. The candidates that have similar opinions can unite and form a new party which is a combination of these parties. This process is a simple model of coalition which is pursued by some candidates in real-world elections. The election algorithm iteratively applies positive advertisement, negative advertisement and coalition on population until termination conditions are satisfied. Once the algorithm stops, the candidate who attained the majority of votes will be announced as the winner. The winner candidate is equal to the best solution found for the given optimization problem.

3.2 Working Principle

Figure 1 shows the working principle of the election algorithm. The algorithm starts with an initial population. Each individual in the population is called a person. For a problem with x_1, x_2, $\dots$, $x_{N_{var}}$ variables, the initial population consists of N_{pop} persons. Each person P_i is an $1 \times N_{var}$ array of variables values and is defined as

$$P_i = [x_1, x_2, \dots, x_{N_{var}}]. \tag{1}$$

The eligibility of a person P_i is found by evaluation of the eligibility function $\mathbf{E}$ at the variables x_1, x_2, $\dots$, $x_{N_{var}}$ considering objective function of the problem.

The eligibility function is defined as follows:

$$E(P_i) = E(x_1, x_2, \ldots, x_{N_{var}}). \tag{2}$$

The persons are divided to several political parties. To fulfill this aim, from the total population, N_c of the most popular persons (the persons with best eligibility values) are selected to be candidates, and the remaining N_v persons will be the voters, each of which belongs to a candidate. The voters are divided among candidates based on their eligibility distance. Voter v_k is considered as a supporter of candidate c_i, if the following predicate holds.

$$P_i = \{v_k : |E_{v_k} - E_{c_i}| < |E_{v_k} - E_{c_j}| \quad \forall \ 1 \leq j \leq N_c\} \tag{3}$$

where P_i is the i^{th} party and N_c is the number of initial candidates. E_{c_i} and E_{v_k} present the eligibility of candidate c_i and voter v_k, respectively. In the party formation process, each voter is assigned to exactly one party. After dividing the voters among candidates and forming the initial parties, the candidates start advertising campaign. The advertising campaign consists of three main phases: positive advertisement, negative advertisement and coalition.

The positive advertisement is modeled by conveying some variables of the candidate to its voters inside a party. To do this task, in each party, N_s variables of the target candidate are randomly selected and replaced with the selected variables of the voters. N_s is computed as follows:

$$N_s = \lceil X_s \times S_c \rceil \tag{4}$$

where S_c is the number of candidate's variables and X_s is the selection rate. The selected variables N_s are weighted with a coefficient ω and then embedded in voters. The new value for the i^{th} variable of a voter after positive advertisement is given by:

$$x_{i_{new}} = \omega . x_{i_{old}}, \quad \text{where} \quad \omega = \frac{1}{|E_{c_i} - E_{v_k}| + 1}. \tag{5}$$

In negative advertisement, candidates try to attract voters of weak candidates toward themselves. A party is weak if its candidate to be the weakest compared to other parties' candidates. To model the negative advertisement, first, a number of voters from the weakest party are selected. Then, a race is taking place among powerful parties to possess these voters. To select the weakest voters from the weakest party, the eligibility distance between the voters, and the weakest candidate is computed, and then 5 % of the farthest voters are selected. The distances between selected voters and the powerful candidates are computed, and the voters are assigned to the closest candidates.

In coalition phase, several candidates join together and form a new party. Among the candidates that wish to collate, a candidate is picked up at random to be the leader candidate and the remaining are considered as the followers. In coalition, all of the follower candidates and their voters become the voters of the leader one.

Until termination conditions are not satisfied, the advertising campaign operators are iteratively applied to update the population. Finally, the update process stops and the candidate with the majority of votes is announced as the winner. The winner is equal to the best solution found for the optimization problem.

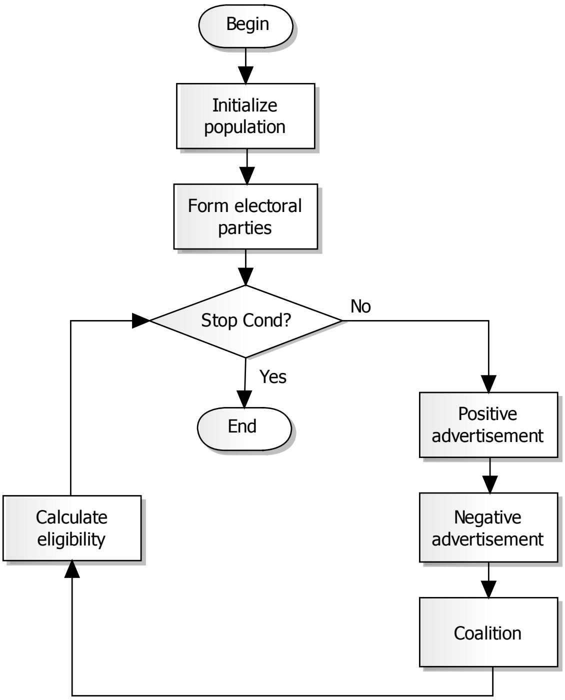


Figure 1. The working principle of the election algorithm

4 CHAOTIC ELECTION ALGORITHM

The advertising campaign is the core operator in the election algorithm, which causes the individuals converge to an optimal point in the search space. However, advertising campaign suffers from three challenges:

1. computing the Euclidean distance in the creation of initial parties and the negative advertisement steps that decrease the speed of the algorithm,
2. getting stuck at local optima,
3. inefficiency of positive advertisement phase.

In advertising campaign, after several iterations, diversity in the population may decrease. As a result, the candidates and their voters cannot explore the entire

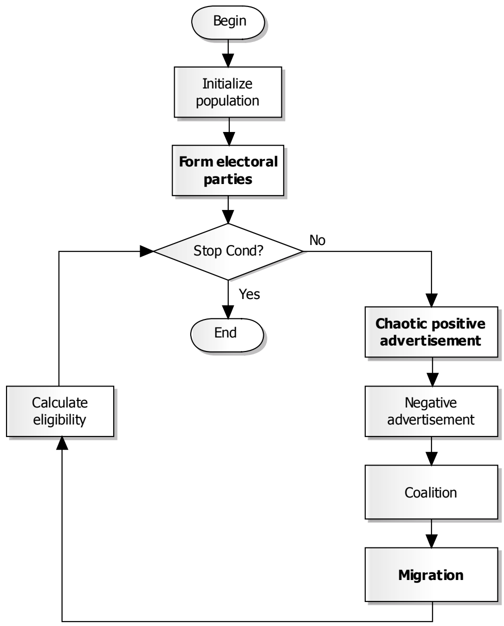


Figure 2. The working principle of the CEA algorithm

solution space and get stuck at local optima. To alleviate these issues, we proposed a Chaotic Election Algorithm, denoted as CEA. Figure 2 shows the flowchart of the CEA algorithm. The CEA enhances the election algorithm threefold:

1. increasing the speed of electoral party formation step utilizing random initialization method,
2. introducing migration operator, and
3. improving the positive advertisement using chaotic maps.

In the following, these enhancements are described.

4.1 Electoral Party Formation

As mentioned above, one drawback of the election algorithm is the computation of Euclidean distance for creating the initial electoral parties that decreases the speed of the algorithm. To alleviate this issue, we substitute the computation of Euclidean distance with a random initialization process. By this way, the voters are divided among candidates based on their eligibility, in which the initial number of voters of a candidate is proportionate to its eligibility. To identify the voters of a candidate c_i, first, its normalized eligibility is computed as

$$ne_{c_i} = \left| \frac{e_{c_i} - \max(I)}{\sum k \in N_c e_{c_k} - \max(I)} \right| \quad \text{where} \quad I = \{e_{c_j} | j \in N_c\} \tag{6}$$

where e_{c_i} is the eligibility of candidate c_i, ne_{c_i} is the normalized eligibility of candidate c_i, and N_c is the initial number of candidates. The initial number of voters of candidate c_i is computed as

$$N_{v_{c_i}} = \lceil ne_{c_i} \times N_v \rceil . \tag{7}$$

N_v is the number of all voters.

Then we randomly select $N_{v_{c_i}}$ of the voters and give them to candidate c_i. The voters along with their candidate c_i form an electoral party P_i in the solution space.

4.2 Migration

We introduced migration operator to help the election algorithm maintain diversity in the population and improve its optimization and search capability. Migration keeps the election algorithm away from converging too fast before exploring the entire solution space. The motivation to introducing the migration operator comes from the fact that in some real-world elections, some individuals can travel from other countries to the target country and vote to their favourite candidate. The travellers are referred as migrants, which can increase the popularity of some candidates. To model migration, some new voters are randomly generated on different areas of the solution space. Here, the new generated voters referred as migrants. The number of migrants at every generation of the algorithm is given by:

$$M = \lceil \mu \times N_{pop} \rceil \tag{8}$$

where M is the number of new migrants, μ is the migration coefficient, and N_{pop} is the population size. In the implementations, the proper value for μ is determined empirically. The migration in every generation of the algorithm adds M new individuals to the population. This causes two issues:

1. excessive growth of the population and
2. increasing the computational time of the algorithm.

To alleviate these issues, we eliminate M of the weakest individuals from the population at every generation of the algorithm. To do this, first all of the individuals in the population are sorted based on their eligibility in ascending order and then M of the inferior individuals (the individuals with lowest eligibility) are removed.

4.3 Chaotic Positive Advertisement

In the election algorithm, positive advertisement is realized through transferring some randomly selected variables from a candidate to its voters. The information only transfers towards voters and the candidate remains without change. Two weaknesses may exist in this way. First, the information exchange (social learning) is one-directional, in which some variables of candidates convey towards voters. As a result, the candidates and their voters cannot explore the entire solution space and the convergence speed decreases. Second, the voters who are affected and their variables are all chosen randomly. As a result, voters with higher eligibility, which may guide the population towards global optimums are not utilized. To overcome these issues and improve the exploration and exploitation ability of canonical EA, we proposed a new chaotic positive advertisement. Chaos is a special kind of dynamic behavior of non-linear systems [41]. Due to the high ability to avoid being trapped in local optima and easy implementation, chaos has raised enormous interest in optimization theory [41, 57]. The application of chaotic maps instead of random variables in the positive advertisement phase is a powerful mechanism to increase diversity of the population and improve the CEA's performance in preventing premature convergence to local optima. Let $v_k(t)$ denote the position of voter k in the search space at iteration t, and $c_i(t)$ denote the position of candidate i at iteration t. The position of voter v_k at iteration $t+1$ is computed as

$$v_k\,(t+1) = v_k(t) + A + B \tag{9}$$

where t indicates the current iteration, A and B are coefficient vectors, which are calculated as

$$A = \omega \times r \times V_1, \tag{10}$$

$$B = \omega \times r \times \tan(\theta) \times V_2 \tag{11}$$

where r is the chaotic variable generated based on a chaotic map, V_1 is a vector where its starting point is the previous position of the voter v_k and its direction is toward the candidate position c_i, and V_2 is a unit vector which is perpendicular to V_1. It is important to notice that $V_1.V_2 = 0$. ω is the distance between voter v_k and candidate c_i, which is computed as

$$\omega = \left| c_i(t) - v_k(t) \right|. \tag{12}$$

By term A, the candidate c_i attracts voter v_k towards itself with no deviation (point l_1 in Figure 3). In order to increase the searching around the candidate c_i, some deviations are added to locate the final position of the voter v_k in its movement toward candidate c_i (point l_2 in Figure 3). By this way, different points around the candidate c_i are explored. $\theta \in U(-\lambda, +\lambda)$ is a random number with uniform distribution regenerated every iteration. λ adjusts the deviation of voter v_k from its original direction. In our implementation, $\lambda = \pi/4$ is used that resulted in good convergence of individuals to the global optimum.

Different chaotic maps can be used to generate chaotic variables. In our implementations, we used logistic map [41] to generate chaotic variable r. The reason to this choice is that CEA have shown better performance when logistic map have been used in compared to the other chaotic maps. The logistic map shows good chaotic properties, it displays better randomness than other maps, and it can navigate the algorithm to the points that have been distributed in search space as much as possible [30, 27].

Logistic map is defined as

$$r_{k+1} = ar_k(1 - r_k) \tag{13}$$

where r_k represents the k^{rmth} number in the chaotic sequence, and k means the index of the chaotic sequence. $r \in (0, 1)$ under the conditions that the initial $r_0 \in (0, 1)$ and that $r_0 \notin \{0.0, 0.25, 0.5, 0.75, 1\}$. In the experiments $a = 4$ is used. In the current study, 1-dimension, non-invertible logistic map is used to produce chaotic sequences.

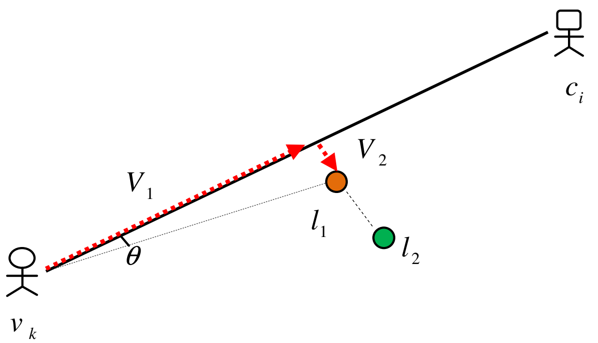


Figure 3. Attracting of voter v_k toward candidate c_i in the chaotic positive advertisement

Due to the non-repetition and ergodicity property of chaotic variables and non-repetition nature of chaos, the newly proposed chaotic positive advertisement carries out overall searches at higher speed than the standard positive advertisement, which is based on the random-based searches. The incorporation of the chaotic positive advertisement in the CEA has two advantages: (i) improving the information exchange between candidates and voters, and (ii) searching efficiently the entire solution space to find a global optimum point. Based on the simulation results presented in the next section, CEA is faster when compared with the canonical EA.

5 EXPERIMENTS

The proposed CEA algorithm is tested on 28 benchmark functions. The CEA algorithm is compared with several top-performing meta-heuristics in solving real-parameter optimization problems, including Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [60], Self-adaptive Differential Evolution (SaDE) algorithm [61], adaptive Differential Evolution (JDE) algorithm [62], PSO2011 [50], Election algorithm Emami2015, Socio Evolution & Learning Optimization (SELO) algorithm [63], and Chaotic Salp Swarm Algorithm (CSSA) [56]. CMA-ES, SaDE, and JDE algorithms are the most successful optimization algorithms. In the competitions at different CEC conferences, these algorithm and their variants possess top positions when compared to other best performing algorithms. PSO2011 [50] is an advanced version of the standard PSO, which incorporates many improvements of PSO that have been identified by years of studies. SELO is a novel meta-heuristic inspired by the social learning behavior of humans organized as families in a societal setup. The reason behind the selection of SELO as a comparative algorithm is that it is a socio-inspired strategy (similar to CEA, which is a socio-politically inspired strategy), and it outperformed other socio-inspired algorithms. CSSA is a chaotic version of SSA algorithm and achieved encouraging results [56].

5.1 Benchmark Functions

Twenty eight well-known benchmark functions are used in the experiment. These are continuous, unbiased optimization problems and have different degrees of complexity and multi-modality. This set of problems has different kinds of properties such as unimodal, multimodal, separable and non-separable. These problems are single objective optimization problems taken from various sources including CEC2005 [58], CEC2013 [64], CEC2015 [59] and recently published papers. The benchmark functions can be classified into four groups:

Group I: F1-F10 are unimodal functions. These functions are used to assess the fast-converging performance of CEA and comparative algorithms.

Group II: F11-F20 are multimodal functions. These functions have many local optima points and are considered to evaluate the ability of algorithms to avoid local optima. Details about hybrid benchmarks are given in [11, 39].

Group III: F21-F24 are shifted and rotated multimodal functions whose base functions belong to Group II functions. These functions are enough complex and used to test the search capability of algorithms.

Group IV: F25-F28 are hybrid multimodal functions whose base functions belong to Group I, II and III functions. This set of functions is more complex than other ones and used to test the performance of algorithms in finding the global optimum of problems consisting of different subcomponents with different properties. Details about hybrid benchmarks are given in [58, 59].

We test the benchmark functions in 30 and 50 dimensions to draw empirical conclusion on the performance of the algorithms. Tables 2, 3, 4 and 5 list the characteristics of benchmark functions used in the tests. These functions are introduced in evolutionary computation share tasks and utilized by many researchers [11, 63, 10, 53].

We have chosen these benchmarks to be able to fairly compare our results to those of the counterpart algorithms.

5.2 Parameter Setting

The initial population sizes of all algorithms were 100 and the maximum number of function evaluations was 100 000. The other specific parameters for the algorithms are given in Table 6, as provided by their authors. We used five predefined criteria to terminate the algorithms' searching process that include:

- if the algorithm failed to find a better solution than the existing solution during the last 100 000 function evaluations,
- if the number of function evaluations reaches 1 000 000,
- if the maximum number of iterations (2 000 000 iterations) was reached,
- if the value of the objective function is less than 10^{-16},
- if the fitness value reaches below a predefined maximum error, the function evaluation is terminated.

All algorithms were programmed in MATLAB R2017a on a Personal Computer Intel Pentium 4 with the 3 GHz and 2 GB RAM. The operating system of the computer is Windows 7.

Problem	Name	Type	Range	Minimum	Definition
f_1	Cigar	S	[0, 10]	0	$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$
f_2	Discus	S	[0, 10]	0	$f_2(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$
f_3	DixonPrice	N	[-10, 10]	0	$f_3(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
f_4	Powell	N	[-4, 5]	0	$f_4(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i^{\ 2}\right) + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$
f_5	Rosenbrock	N	[-30, 30]	0	$F_5(x) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
f_6	Schwefel_1_2	N	[-100, 100]	0	$f_6(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$
f_7	Schwefel_2_22	N	[-10, 10]	0	$f_7(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
f_8	Sphere	S	[-100, 100]	0	$f_8(x) = \sum_{i=1}^{n} x_i^{\ 2}$
f_9	Sumsquares	S	[-10, 10]	0	$f_9(x) = \sum_{i=1}^{n} ix_i^{\ 2}$
f_{10}	Zakharov	N	[-5, 10]	0	$f_{10}(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i^{\ 2}\right) + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$

Table 2. Unimodal benchmark problems. Range: limits of search space, N: non-separabple, S: separable.

Problem	Name	Type	Range	Minimum	Definition
f_{11}	Ackley	N	[-32, 32]	0	$f_{11}(x) = -20\exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
f_{12}	Alpine	N	[0, 10]	0	$f_{12} = \sum_{i=1}^{n}\lvert x_i sin(x_i) + 0.1x_i\rvert$
F_{13}	Griewank	N	[-600, 600]	0	$F_{13}(x) = \frac{1}{4000}\left(\sum_{i=1}^{n} x_i^2\right) - \left(\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)$
f_{14}	Leavy	N	[-10, 10]	0	$f_{14}(x) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_n - 1)^2\left[1 + \sin^2(2\pi w_n + 1)\right]$ $w_i = 1 + \frac{x_i - 1}{4}, \quad i = 1, 2, ..., n$
f_{15}	Penalized	N	[50, 50]	0	$f_{15}(y) = \frac{\pi}{n} \times \left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} +$ $\sum_{i=1}^{n} u(x_i, a, k, m)$ $(y_i = 1 + \frac{1}{4}(x_i + 1), \quad u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases} \quad (a = 10, k = 100, m = 4)$
f_{16}	Penalized2	N	[-50, 50]	0	$f_{16}(x) = 0.1 \times \left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} +$ $\sum_{i=1}^{n} u(x_i, a, k, m)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases} \quad (a = 5, k = 100, m = 4)$
f_{17}	Periodic	N	[-10, 10]	0.9	$f_{17}(\mathbf{x}) = 1 + \sum_{i=1}^{n} sin^2(x_i) - 0.1e^{(\sum_{i=1}^{n} x_i^2)}$
f_{18}	Rastrigin	S	[-5.12, 5.12]	0	$f_{18}(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
f_{19}	Schwefel	S	[-500, 500]	0	$f_{19}(x) = 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$
f_{20}	Shubert	N	[-10, 10]	-186.7309	$f_{20}(\mathbf{x}) = \prod_{i=1}^{n}\left(\sum_{j=1}^{5} cos((j+1)x_i + j)\right)$

Table 3. Multimodal benchmark problems. Range: limits of search space, N: non-separable, S: separable.

Problem	Name	Type	Range	Minimum	Definition
f_{21}	Shifted Sphere Function	S	[-100, 100]	-450	$f_{21}(z)=\sum_{i=1}^{n} z_i^2 + f\_bias,\quad z=x-o$
f_{22}	Shifted Schwefel	N	[-100, 100]	-450	$f_{22}(z)=\sum_{i=1}^{n}(\sum_{j=1}^{i} z_j)^2 + f\_bias,\quad z=x-o$
f_{23}	Shifted Rosenbrock's	N	[-100, 100]	390	$f_{23}(z)=\sum_{i=1}^{n-1}\left((z_i^2-z_{i+1})^2+(z_i-1)^2\right)+f\_bias,\quad z=x-o+1$
f_{24}	Shifted rotated Rastrigin's	N	[-5, 5]	-330	$f_{24}(z)=\sum_{i=1}^{n}\left(z_i^2-10\cos(2\pi z_i)+10\right)+f\_bias,\quad z=(x-o)*M$

Table 4. Shifted and rotated benchmark problems. Range: limits of search space, N: non-separable, S: separable.

5.3 Results

In experiments, the algorithms ran for 30 times for all test functions, each time using a different initial population. We test the benchmark functions in 30 and 50 dimensions to draw empirical conclusion on the performance and scalability of the algorithms. The statistical results are reported in Tables 7–14. In these tables, *min* and *mean* are respectively the minimum and the mean function values obtained by the algorithms over 30 simulation runs. *Std* indicates the standard deviation of the results, and *Succ* indicates the number of success trials over 30 simulation runs. *Succ* is defined as

$$Succ = \bigcup_{i=1}^{30} N_{Succ}\,|_{\varepsilon}\,. \tag{14}$$

where N_{Succ} denotes the number of successful trials, in which the solution is found on ε. In simulations, an algorithm found global optimum when it converges into ε tolerance and it is defined as

$$|f_{\cos t}(T_i) - f_{\cos t}(T^*)| \le \varepsilon \tag{15}$$

where $f_{\cos t}(T_i)$ denotes the cost function value in i^{th} iteration and $f_{\cos t}(T^*)$ indicates the global optimum of the test function.

In Tables 7–14, in order to make comparison clear, the values below 10^{-16} are assumed to be 0. In Tables 7–14, symbol "n" presents the dimension of the problems. As shown in Tables 7–14, for 30-dimension problems, the CEA algorithm performed best on 26 benchmark functions. The second and third ranks belong to JDE and SADE with 24 and 22 successes, respectively. The election algorithm, CSSA, SELO, PSO2011 and CMA-ES performed best on 22, 21, 20, 20 and 19 benchmark functions, respectively. For 50-dimension problems, the CEA algorithm performed best on 20 benchmark functions and takes the first rank. The second and third ranks belong to JDE and SADE with 18 and 16 successes, respectively. The election algorithm, CSSA, SELO, PSO2011 and CMA-ES performed best on 15, 13, 14, 9 and 13 benchmark functions, respectively. From numerical simulations,

Problem	Name	Type	Range	Minimum	Definition
f_{25}	Hybrid composition function	S	[-5, 5]	120	$F_1, F_2 =$ *Rastrigin's Function* $F_3, F_4 =$ *Weierstrass Function* $F_5, F_6 =$ *Griewank's Function* $F_7, F_8 =$ *Ackley's Function* $F_9, F_{10} =$ *Sphere Function* $[\sigma_1, \sigma_2, ..., \sigma_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]$
f_{26}	Rotated hybrid comp. Fn 1	N	[-5, 5]	120	*rotated version of* f_{25} : $F_1, F_2 =$ *Rastrigin's Function* $F_3, F_4 =$ *Weierstrass Function* $F_5, F_6 =$ *Griewank's Function* $F_7, F_8 =$ *Ackley's Function* $F_9, F_{10} =$ *Sphere Function* $[\sigma_1, \sigma_2, ..., \sigma_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]$
f_{27}	Rotated hybrid comp. Fn 2	N	[-5, 5]	310	$F_1, F_2 =$ *Ackley's Function* $F_3, F_4 =$ *Rastrigin's Function* $F_5, F_6 =$ *Sphere Function* $F_7, F_8 =$ *Weierstrass Function* $F_9, F_{10} =$ *Griewank's Function* $[\sigma_1, \sigma_2, ..., \sigma_{10}] = [1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [2*5/32, 5/32, 2*1, 1, 2*5/100, 5/100, 2*10, 10, 2*5/60, 5/60]$
f_{28}	Rotated hybrid comp. Fn 4	N	[-5, 5]	-330	$F_1 =$ *Weierstrass Function* $F_2 =$ *Rotated Expanded Scaffer's Function* $F_3 =$ *F8F2 Function* $F_4 =$ *Ackley's Function* $F_5 =$ *Rastrigin's Function* $F_6 =$ *Griewank's Function* $F_7 =$ *Non-Continuous Expanded Scaffer's Function* $F_8 =$ *Non-Continuous Rastrigin'sFunction* $F_9 =$ *High Conditioned Elliptic Function* $F_{10} =$ *Sphere Function with Noise in Fitness* $[\sigma_1, \sigma_2, ..., \sigma_{10}] = [1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [2*5/32, 5/32, 2*1, 1, 2*5/100, 5/100, 2*10, 10, 2*5/60, 5/60]$

Table 5. Hybrid benchmark problems. Range: limits of search space, N: non-separable, S: separable.

it is obvious that all algorithms have almost consistent behavior on all benchmark functions. The solution quality and convergence accuracy obtained on most test functions using the CEA in 30 independent simulation runs are almost exceeding or matching the best performance obtained by other algorithms. This testifies that the CEA has better stability behavior on most test functions rather than other algorithms.

CEA outperforms all compared algorithms on the unimodal benchmark functions in terms of the statistical test. The performance of CEA in solving multimodal benchmark problems is superior, and it generates best results in terms of *min* and *mean* values in solving 30 and 50 dimension benchmark functions. The worst results belong to CSSA, SELO and PSO2011 in solving 30 and 50 dimension multimodal benchmark functions. When solving shifted and rotated benchmark functions, CEA generally performs very well in 30 dimension benchmark functions, however, it does

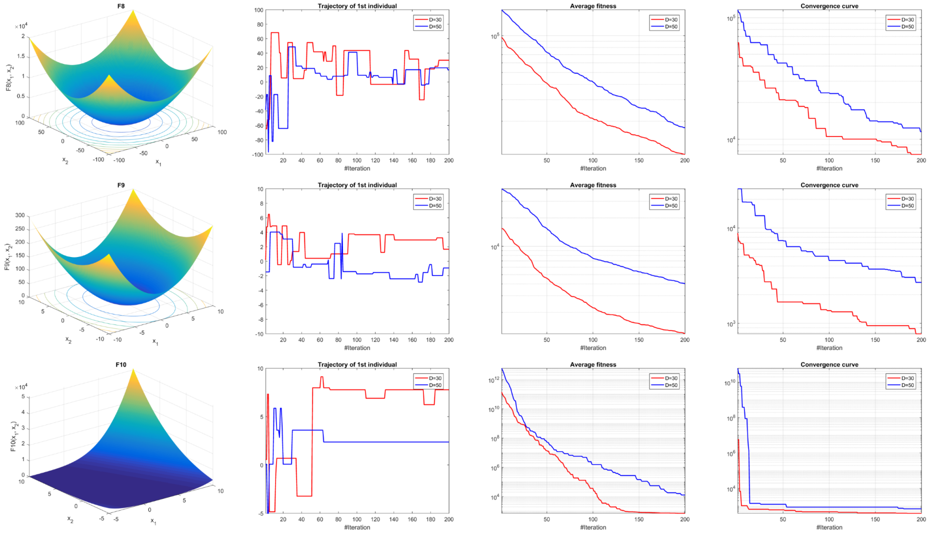
Algorithm	Control parameters
CMA-ES	$\sigma = 0.25$, $\mu = \left\lfloor \dfrac{4 + \lfloor 2.\log(N) \rfloor}{2} \right\rfloor$
SADE	$F \sim N(0.5, 0.3)$ $CR \sim n(CR_m, 0.1)$ $c = 0.1$ $p = 0.05$
JDE	$f_{initial} = 0.5$, $CR_{initial} = 0.90$, $\tau_1 = 0.1$, $\tau_2 = 0.1$
PSO2011	$C_1 = 1.80$ $C_2 = 1.80$ ω=0.5+(1-rand)
Election algorithm	$N_C = 0.7 \times P_{size}$, $N_V = P_{size} - N_C$, Coalition rate=0.2, X_s =0.30
SELO	$P = 2$, $O = 3$, $r_p = 0.999$, $r_k = 0.1$ $follow\_prob\_factor\_ownparent = 0.999$ $follow\_prob\_factor\_otherkids = 0.9991$ $r = 0.95000$ *to* 0.99995
CSSA	$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2}$, c_2, $c_3 \in [0,1]$
CEA	$N_C = 0.7 \times P_{size}$, $N_V = P_{size} - N_C$, Coalition rate=0.2, X_s =0.30 $\mu = 0.10$

Table 6. Control parameters of the algorithms used in the tests

not perform well in 50 dimension functions. SADE performs very well in solving 50 dimension shifted and rotated benchmark functions. CEA is not as competitive in solving 50 dimension hybrid functions as it does in unimodal and multimodal benchmarks. However, a careful investigation on the *mean* values shows that the performance of CEA is encouraging. From simulation results we can see that CEA performs very well in 30-dimension hybrid functions, JDE and SADE perform well in 50 dimension hybrid functions, and PSO2011 and election algorithm report the worst results in 30 and 50 dimension hybrid functions when compared with other algorithms. The *mean* and *min* values of CMA-EA, SELO, and CSSA in solving both 30 and 50 dimension functions are close to each other and show moderate results.

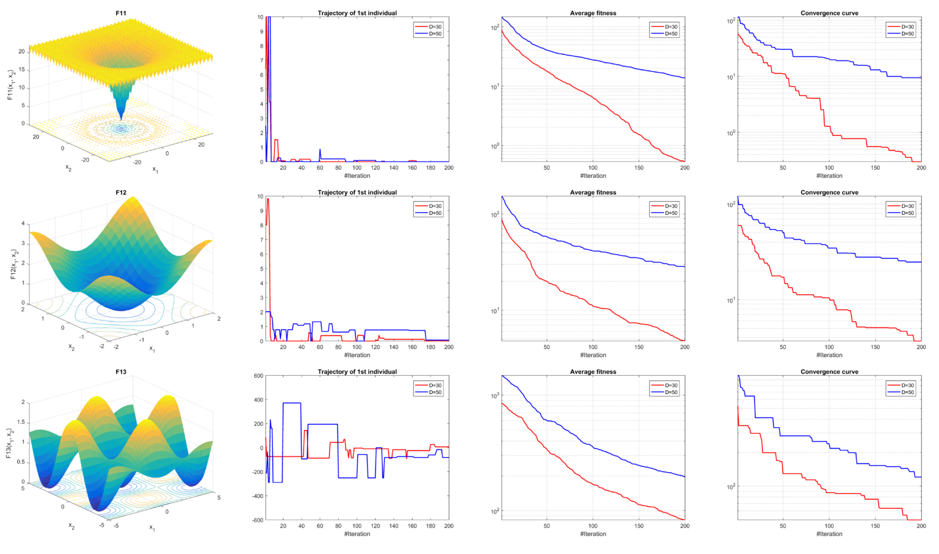
Table 15 presents the multi-problem based pairwise statistical comparison results on 30-dimension benchmark functions using the averages of the global minimum values obtained through 30 simulation runs of CEA and the comparison algorithms, based on the Wilcoxon Signed-Rank Test [26]. Table 16 presents the multi-problem based pairwise comparison results for 50-dimension benchmark functions. Multi-problem based pairwise comparisons identify which algorithm is statistically more successful in a test that includes several benchmark problems [26]. The results show that CEA was statistically more successful than other algorithms in solving the benchmark functions with a statistical significance value $\alpha = 0.05$.

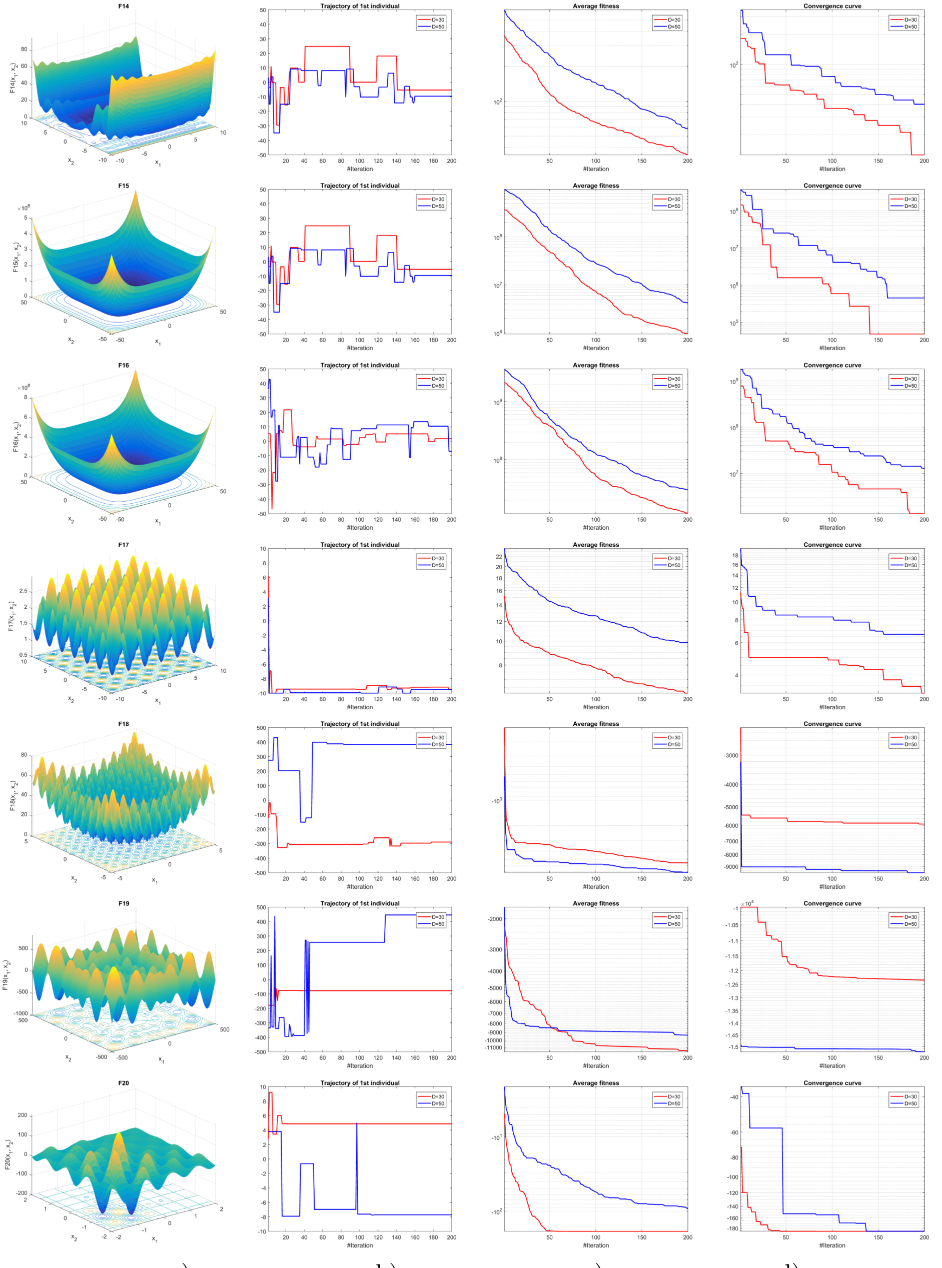
In order to observe the convergence behavior of the CEA algorithm, the convergence curve, the average fitness, and the trajectory of the first individual in its first dimension are illustrated in Figures 4, 5, 6 and 7. It should be noted that for greater clarity of plots the behavior of algorithms is shown only for 200 iterations. The second column of Figures 4, 5, 6 and 7 depicts the trajectory of the first individual in the population, in which changes of the first person in its first dimension



a) b) c) d)

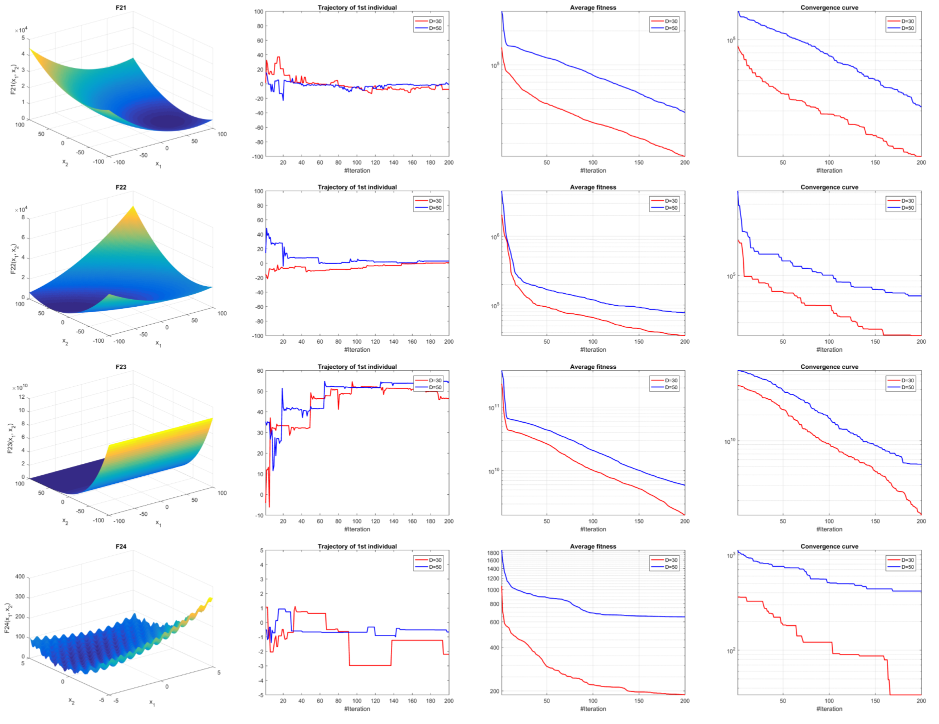
Figure 4. Results on unimodal problems, a) Graphical representations of benchmark problem, b) trajectory of the first individual in the first dimension, c) the average fitness of individuals, and d) the convergence curve of CEA algorithm





a) b) c) d)

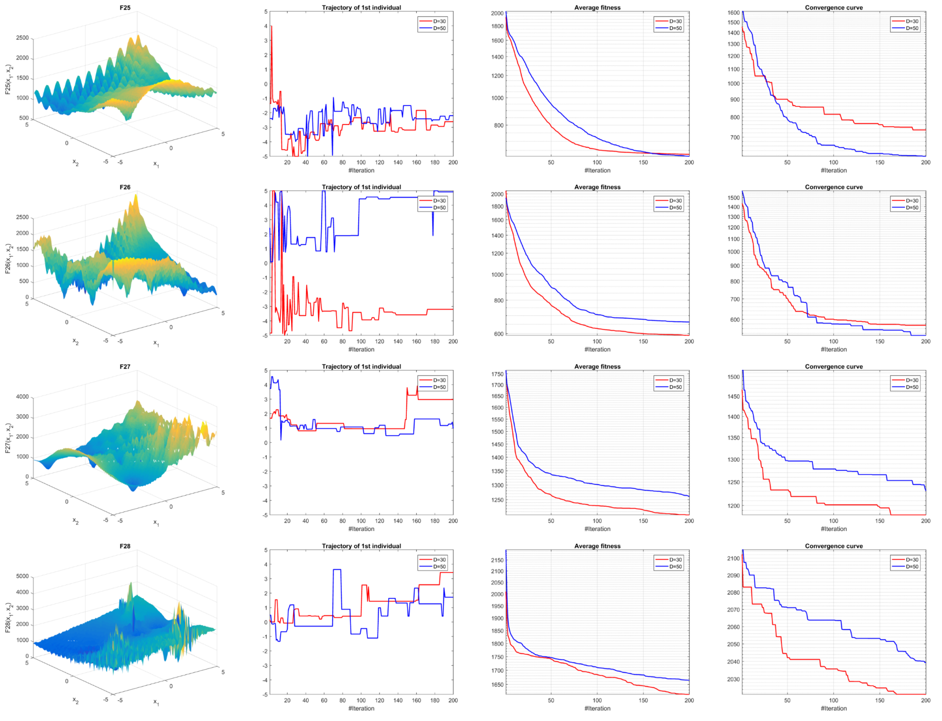
Figure 5. Results on multimodal problems, a) Graphical representations of benchmark problem, b) trajectory of the first individual in the first dimension, c) the average fitness of individuals, and d) the convergence curve of CEA algorithm



a) b) c) d)

Figure 6. Results on shifted and rotated problems, a) Graphical representations of benchmark problem, b) trajectory of the first individual in the first dimension, c) the average fitness of individuals, and d) the convergence curve of CEA algorithm

can be observed. It can be seen that there are abrupt changes in the initial steps of iterations. These abrupt changes are decreased gradually during the search process. This behavior guarantees that a population-based algorithm eventually converges to a point in search space [53, 56]. The third column of Figures 4, 5, 6 and 7 shows the average fitness that individuals obtain over 200 iterations. It can be observed that average fitness values are decreased gradually. From this behavior, it can be concluded that the fitness of individuals in the population improves through iterations. This is due to a proper balance between exploration and exploitation power of the CEA algorithm. The forth column of Figures 4, 5, 6 and 7 shows the convergence curve of CEA algorithm. It can be seen that the proposed CEA algorithm converges with a steady speed. This behavior shows the superiority of the CEA algorithm in terms of the stability and the performance. To sum up, the results verify the performance of the CEA algorithm in solving various benchmark problems compared to the counterpart algorithms. It can be concluded that the proposed CEA is an efficient algorithm for numerical function optimization.



a) b) c) d)

Figure 7. Results on hybrid problems, a) Graphical representations of benchmark problem, b) trajectory of the first individual in the first dimension, c) the average fitness of individuals, and d) the convergence curve of CEA algorithm

From the results we can see that time consumption of JDE costs least time on most test functions. CSSA costs the most time and it is in the last rank. Although CEA reported slightly more run time than JDE and SADE, their run times are comparable on most benchmarks. In contrast, CEA reports run times less than election algorithm, CMA-EA, CSSA, PSO2011 and SELO on most benchmark functions. While CEA reports more run time than JDE and SADE (on most benchmarks), its results are much better in terms of solution quality and finding global optima. On most test functions, CEA obtained the global optimum earlier before the total function evaluations. This is the reason that the time consumption of CEA is much better than the election algorithm. This justifies that the CEA is a powerful and robust extension of the election algorithm.

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_1	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	2.015	3.187	5.447	5.409	4.348	6.47	7.005	4.257
f_2	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000022	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000846	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000011	0.0000000000000000
	Succ	100	100	100	100	100	100	0	100
	Time	1.751	1.221	3.418	4.266	6.632	3.400	7.204	6.833
f_3	Min	0.6666669588917279	0.6666666666666670	0.6666666666666670	0.6666666666666720	0.6429261382035644	0.9541730938494050	0.6666666835995361	0.635574103060858
	Mean	0.6666743346755102	0.6666666666666670	0.6666666666666670	0.6666666666666750	0.6484196536514408	0.9737369841168760	0.6666667989603425	0.635970446657034
	Std	0.0000079162957191	0.0000000000000002	0.0000000000000000	0.0000000000000022	0.0000019450349497	0.0054869670667257	0.0000002248904081	0.000000250446178
	Succ	0	0	0	0	0	0	0	0
	Time	22.477	23.689	13.206	26.225	29.256	33.127	41.1351	32.836
f_4	Min	0.00018812227398726	0.0000000000000000	0.0000077644005742	0.0000095067504097	0.0000000001209500	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.00153977762312255	0.0000000000000000	0.0000110725465874	0.0000130718912008	0.0000002775466712	0.0000000000000000	0.0000000000000636	0.0000000000000000
	Std	0.00167292734740937	0.0000000000000000	0.0000034063714767	0.0000014288348929	0.0000000561469137	0.0000000000000000	0.0000000000000235	0.0000000000000000
	Succ	0	100	0	0	0	100	43.33	100
	Time	55.916	18.692	21.171	57.007	55.370	16.237	47.098	40.159
f_5	Min	0.0000000000000000	0.0000000000000000	0.0001448955835246	0.0042535368984501	0.0037741791472449	0.0000000000000000	7.1440236889946900	0.0000000000000000
	Mean	0.3986623855035210	1.0630996944802500	1.2137377447007000	2.6757043114269700	22.4020614789213527	0.0000000000000000	8.8138860361911850	0.0000000000004077
	Std	1.2164328621946200	1.7930895051734300	1.8518519388285700	12.3490058210004000	7.4158314789297055	0.0000000000000000	0.06482678290314312	0.0000000000000370
	Succ	30	20	0	0	0	100	0	33.33
	Time	82.485	19.278	45.607	43.064	37.181	13.101	34.785	30.113
f_6	Min	0.5023592840073707	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000007	0.0000000000000000	0.0000000000000000
	Mean	1.7915045718386977	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000020	0.0000000000000009	0.0000000000000000	0.0000000000000000
	Std	1.9585800691758875	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000002	0.0000000000000001	0.0000000000000000	0.0000000000000000
	Succ	0	100	100	100	83.33	0	100	100
	Time	135.024	13.679	109.551	103.764	27.673	12.688	24.315	19.163
f_7	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	18.063	3.113	8.335	٧.513	6.178	4.374	14.8644	5.832
f_8	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	7.322	3.215	4.920	6.131	9.157	1.871	16.187	9.004
f_9	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	0	100
	Time	8.670	2.098	6.383	7.229	3.577	2.115	13.70	3.180
f_{10}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	100.868	10.165	21.488	18.083	13.213	7.351	16.1914	13.210

Table 7. Results of unimodal benchmark functions, $n = 30$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_1	Min	0.0000000000000000	0.0000000000000000	0.0000000000000013	0.0000000307641078	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000034	0.0000000939418142	0.0000000000000000	0.0000000000000000	0.0000000000000638	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000015	0.0000000753481345	0.0000000000000000	0.0000000000000000	0.0000000000000076	0.0000000000000000
	Succ	100	100	0	0	100	100	80	100
	Time	16.749	3.433	21.516	23.638	15.348	14.471	27.201	15.451
f_2	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000001656881	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000003494273	0.0000000000009314	0.0000000000000000	0.0000000000000281	0.0000000000000000
	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000002131836	0.0000000000006654	0.0000000000000000	0.0000000000000090	0.0000000000000000
	Succ	100	100	100	0	76.67	100	83.33	100
	Time	17.135	3.032	19.731	11.3352	12.346	11.400	31.616	12.007
f_3	Min	0.6666666712080868	0.6666666666666665	0.6666666666666665	0.6666670230472646	0.6666666666840324	0.8514339096029764	0.6666668308727451	0.6380227724721356
	Mean	0.6666667405927262	0.6666666666666665	0.6666666666666665	0.6666768415722471	0.6666667195171128	1.0654716657959429	0.6666668873145076	0.6347368218366791
	Std	0.0000000729527354	0.0000000000000000	0.0000000000000000	0.0000120526677103	1.2908660020147678	0.3121960564812137	0.0000000459120280	0.0001051699387589
	Succ	0	0	0	0	0	0	0	0
	Time	25.493	26.783	18.052	20.417	24. 522	41.092	47.542	23.510
f_4	Min	0.0026415268539887	0.0000000000000000	0.0000006977481945	0.0484732633771529	0.0007936812645283	0.0000992123421870	0.0006040263968728	0.0000013599298381
	Mean	0.0031817121734536	0.0000000003809886	0.0000006977481945	0.0642270033828177	0.0018456262696528	0.0001194704095890	0.0006618000863430	0.0000019177635714
	Std	0.0004317292482951	0.0000000000300807	0.0000000000000000	0.0101406354957466	0.0006990594195686	0.0000256898252446	0.0000460410613896	0.0000003640723976
	Succ	0	50	0	0	0	0	0	0
	Time	64.994	28.158	58.800	53.746	58.740	33.306	59.220	54.001
f_5	Min	0.1618363509246701	0.0000000000000000	0.3475981703340429	10.1962821712599410	0.0799738768514800	0.0000002980396757	0.0006544015026803	0.0000000000000000
	Mean	64.2459213107620140	0.0000031909510204	62.6637216523635490	10.3574003379587990	23.9314903263880900	0.0797391612124302	0.0006069684030456	0.0011324151702647
	Std	110.9023700345396900	0.0000063608639448	51.4806879277399430	0.1816473515536368	36.3906215416394900	0.0594635542752125	0.0000115736444337	0.0019614006101603
	Succ	0	0	0	0	0	0	0	20
	Time	91.766	20.545	51.487	43.668	41.702	35.021	49.658	39.849
f_6	Min	0.0960251162413680	0.0000000000000000	0.0000000000000000	0.0764595673918435	0.0000005525442833	0.0000000023703930	0.0000000000000000	0.0000000000000000
	Mean	0.2519480590461070	0.0000000000000000	0.0000000000000000	0.0990401508526359	0.0000046955977214	0.0000006205994047	0.0000000000000651	0.0000178225610856
	Std	0.1393738482498040	0.0000000000000000	0.0000000000000000	0.0179672572699185	0.0000055169568637	0.0000009040824291	0.0000000000000068	0.0000356451221712
	Succ	0	100	100	0	0	0	80	56.67
	Time	251.166	34.821	163.252	102.394	51.794	51.721	69.605	49.383
f_7	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	31.257	7.013	12.227	13.728	10. 634	4.374	21.449	10.530
f_8	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0. 0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000019808005301	0.0000004349891225	0.0000000014838244	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000007641013227	0.0000005546172377	0.0000000012109497	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	76.67	76.67	80	100	100
	Time	20.286	4.972	8.481	14.138	11.183	16.205	19.281	11.196
f_9	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	21.634	3.188	10.850	9.110	8.201	6.462	50.794	27.818
f_{10}	Min	624.5630179244759600	0.0000000000000000	0.0000000000000000	0.0453695392950734	0.0000000000000000	0.0004971972185198	0.0000000000000000	0.0000000000000000
	Mean	963.9116925353591800	0.0000000000000000	0.0000000000000000	0.0946941941778695	0.0000000000003423	0.0017045282408875	0.0000000000000000	0.0000000000000000
	Std	201.1185662003226200	0.0000000000000000	0.0000000000000000	0.0359721171318677	0.0000000000003651	0.0011890090919424	0.0000000000000000	0.0000000000000000
	Succ	0	100	100	0	80	0	0	100
	Time	202.417	22.730	69.996	23.198	22.748	45.951	19.699	19.240

Table 8. Results of unimodal benchmark functions, $n = 50$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_{11}	Min	0.0000000000000027	0.0000000000030836	0.0000000000000027	0.0000000000000080	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000027	0.0000000004992996	0.0000000000000027	1.5214322973725000	0.0000000000000000	0.0000000000000000	0.0000008834796247	0.0000000000000000
	Std	0.0000000000000000	0.0000000002573112	0.0000000000000000	0.6617570384662600	0.0000000000000000	0.0000000000000000	0.0000005120149712	0.0000000000000000
	Succ	0	0	0	0	100	100	0	100
	Time	30.287	25.136	10.428	23.039	9.250	5.674	28.616	9.222
f_{12}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000247	0.0000000000005810	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000015	0.0000000000000339	0.0000000000000000
	Succ	100	100	100	100	100	73.34	36.67	100
	Time	15.604	6.140	56.738	29.248	26.877	27.328	51.705	23.751
F_{13}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000274060	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0226359326967139	0.0068943694819713	0.0000000000000000	0.0000000000000000	0.0000140271987100	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0283874287215679	0.0080565201649587	0.0000000000000000	0.0000000000000000	0.0000041779936673	0.0000000000000000
	Succ	100	100	36.66	50	100	100	0	100
	Time	2.647	6.914	25.858	23.895	9.1919	7.940	55.4761	7.16962
f_{14}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0020185116261490	0.0000000000000000	0.0000000000001254	0.0000000000000000	0.0000000000000012	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0077448684015362	0.0000000000000000	0.00000000000001503	0.0000000000000000	0.0000000000000001	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	83.34	100	93.34	100	100
	Time	۵.062	6.692	7.886	9.441	4.685	9.492	26.811	4.116
f_{15}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000043	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000015	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	90	100	100	100	100
	Time	5.851	9.492	15.992	19.600	8.441	18.075	48.514	8.411
f_{16}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0003662455278628	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	0.0020060093719584	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	33.33	100	100	100	100	100	100	100
	Time	6.158	1.969	4.544	12.507	11.715	9.537	31.668	12.324
f_{17}	Min	7.5330185062452992	1.0000000000000002	1.000000000000004	1.0000000000000002	1.0000000000489453	0.9000000000001315	1.0000000000074536	0.9000000000000000
	Mean	8.0447358655943226	1.0000000000000004	1.1305510725002295	1.0000000000000042	1.0000000000591651	1.0000000000055178	1.0000000000102574	0.9200000000175261
	Std	0.1630535915023040	0.0000000000000002	0.0107698043735300	0.0000000000000011	0.0000000000067066	0.0000000000001234	0.0000000000022537	0.0399999999912369
	Succ	0	0	0	0	0	0	0	43.33
	Time	76.081	24.002	29.688	48.170	36.198	73.124	48.5810	23.909
f_{18}	Min	29.8487565993415000	0.0000000000000000	0.0000000000000000	12.9344677422129000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	95.9799861204982000	0.0000000000000000	0.8622978494808570	25.6367602258676000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Std	56.6919245985100000	0.0000000000000000	0.9323785263847000	8.2943512684216700	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	0	100	26.67	0	100	100	100	100
	Time	27.40	3.635	13.594	16.023	7.988	3.941	15.812	7.187
f_{19}	Min	-8340.0386911070600000	-12569.4866181730000000	-12569.4866181730000000	-8912.8855854978200000	-12569.4866181730140000	-0.0325083488969540	-8542.3106048357640000	-12569.4866181730140000
	Mean	-6835.1836730901400000	-12304.9743375341000000	-12549.7468957373000000	-7684.6104757783800000	-12036.1119932232430000	-0.3402784042291390	-7796.5971553930576000	-12166.6227102694010000
	Std	750.7338055436110000	221.4322514436480000	44.8939348779747000	745.3954005014180000	923.8319498809704600	3.2212919091274600	658.0863939861569600	805.7278158072265300
	Succ	0	0	0	0	0	0	0	0
	Time	6.174	10.315	34.383	30.427	15.5106	17.419	20.797	11.143
f_{20}	Min	-186.7309088310240000	-186.7309088310240000	-186.7309088310240000	-186.7309088310240000	-186.7309088310240000	-186.7363874875390000	-186.7309088310240000	-186.7309088310240000
	Mean	-81.5609772893002000	-186.7309088310240000	-186.7309088310239000	-186.7309073569880000	-186.7309088310239200	-186.7153981691330000	-186.7309088310238700	-186.7309088310239000
	Std	66.4508342743478000	0.0000000000000388	0.0000000000000377	0.0000046401472660	0.0000000000000027826	0.0190762312882078	0.0000000000000180	0.0000000000000000393
	Succ	13.34	63.34	66.67	30	60	0	66.67	70
	Time	25.225	8.213	27.109	19.770	31.2819	25.147	69.338	26.335

Table 9. Results of multimodal benchmark functions, $n = 30$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_{11}	Min	0.0000000000000027	0.0000000000000027	0.0000000000000027	0.0000000000004000	0.0000000000000000	0.0000000000000000	0.0000007266174480	0.0000000000000000
	Mean	0.0000000000000027	0.0000000000000027	0.0000000000000027	1.7266174422716470	0.0000000000000000	0.0000000000000000	0.0000051230641084	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.946515355 044351	0.0000000000000000	0.0000000000000000	0.0000001550003205	0.0000000000000000
	Succ	0	0	0	0	100	100	0	100
	Time	37.402	27.267	25.373	37.110	11.041	9.704	95.117	10.984
f_{12}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000001566247	0.0000000528283220	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000375115	0.0000000004796087	0.0000000000000000
	Succ	100	100	100	100	100	50	30	100
	Time	33.568	8.010	53.258	41.066	32.267	38.115	57.338	25.170
F_{13}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000051235828	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0009845014593703	0.0068943694819713	0.0000000000000000	0.0000000000000000	0.0018490161387324	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.00079649632056923	0.0080565201649587	0.0000000000000000	0.0000000000000000	0.0032025787311004	0.0000000000000000
	Succ	100	100	86.67	50	100	100	0	100
	Time	17.862	5.355	24.6358	22.533	12.665	9.385	65.850	11.004
f_{14}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0007913556231509	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.4733646979405994	0.3133488764575432	0.0000000000000000	0.0000089528250416	0.0013102112517912	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.3741896258335322	0.1060956269188929	0.0000000000000000	0.0000000633060329	0.0005322648501157	0.0000000000000000
	Succ	100	100	30	20	100	40	0	100
	Time	20.321	6.702	29.764	28.003	9.344	39.127	43.561	9.760
f_{15}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000018	0.0000000000000000	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.1278728062391630	0.0000000000000000	0.0000642528164196	0.0000000000000003	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.2772792346028400	0.0000000000000000	0.0000151668177694	0.0000000000000002	0.0000000000000000
	Succ	100	100	100	16.66	100	50	80	100
	Time	20.551	4.0768	8.2708	39.555	9.565	34.087	51.196	9.402
f_{16}	Min	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0234368584375104	0.0000000000000000
	Mean	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0043949463343535	0.0000000000000000	0.0000000000000000	0.9453148964751007	0.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0054747064090174	0.0000000000000000	0.0000000000000000	0.1531241868389090	0.0000000000000000
	Succ	100	100	100	16.66	100	100	0	100
	Time	18.993	6.201	9.203	16.137	16.017	17.106	35.648	15.297
f_{17}	Min	14.0566700961880930	1.0000000000000004	1.0000000002066967	1.3795783622590090	1.0000000000369984	1.0000000000057578	0.9000000000000000	0.9000000000000000
	Mean	14.3884549392425620	1.0000000000000011	1.0000006030324166	1.4181834652710708	1.0000000000449019	1.0000000000097171	0.9750000000078725	0.9600000000000000
	Std	0.3017442249329344	0.0000000000000009	0.0000000374858280	0.0254098620222145	0.0000000000053155	0.0000000000025909	0.0433012701937671	0.0489897948556636
	Succ	0	0	0	0	0	0	23.33	50
	Time	252.480	56.117	108.077	62.924	61.730	59.648	65.159	57.322
f_{18}	Min	15.9193449134927500	0.0000000000000000	0.0000000000000000	65.6672473922326960	0.0000000000000000	0.0000000000000000	0.0000000000084371	0.00000000000000000
	Mean	224.9814655516387900	0.0000000000000000	0.2653902873322949	93.0285078955684380	0.9958240635327371	0.0000000000000000	0.0000000000140816	0.00000000000009949
	Std	121.2874373727220700	0.0000000000000000	0.4247444940971572	15.9890829391755500	1. 4026979638646450	0.0000000000000000	0.0000000000016255	0.00000000000070354
	Succ	0	100	43.33	0	20	100	0	70
	Time	191.468	4.922	16.738	55.0321	15.110	14.814	37.8123	15.817
f_{19}	Min	-Inf	-20949.1443632022420000	-20949.1443636216720000	-14342.5691261116770000	-20949.1443636216720000	-19830.1777064995950000	-13337.3783150455510000	-20949.1443630272190000
	Mean	-Inf	-20949.1443620160350000	-20949.1443636216720000	-13770.2818275667050000	-17612.3302345816880000	-18832.9353757876050000	-13019.9077731192970000	-20949.1443612843000000
	Std	NaN	0.0000011287140530	0.0000000000000000	340.9143617460632100	2737.4387810085359000	881.1129403390057200	265.0087423288549100	0.0000029487969428
	Succ	0	0	0	0	0	0	0	0
	Time	-	20.422	22.435s	20.899	21.750	25.132	58.4874	18.454
f_{20}	Min	-186.7261812733892400	-186.7309088310239800	-186.7309088310239800	-186.7309088308585700	-186.7309088309020800	-186.7309088310239800	-186.7309088310240000	-186.7309088310240000
	Mean	-186.6819266547511200	186.7309088310239800	-186.7309088310239200	-186.7309083581201300	-186.7309086707775200	-186.7309088310239500	-186.7309088310237800	-186.7309065904519200
	Std	0.0483046109651153	0.0000000000000284	0.0000000000000318	0.0000008659591989	0.0000003137304075	0.0000000000000376	0.0000000000000651	0.0000007440998569
	Succ	0	0	0	0	0	0	46.67	53.33
	Time	275.1913	29.203	37.847	22.3088	34.366	41.954	36.791	29.075

Table 10. Results of multimodal benchmark functions, $n = 50$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_{21}	Min	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000
	Mean	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	100	100	100
	Time	83.144	11.166	39.155	38.136	43.544	37.250	63.972	44.088
f_{22}	Min	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000
	Mean	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-441.8724485886073130	-450.0000000000000000	-450.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	12.0594053705881000	0.0000000000000000	0.0000000000000000
	Succ	100	100	100	100	100	50	100	100
	Time	21.701	31.067	65.375	59.178	37.922	62.197	57.359	33.430
f_{23}	Min	390.0000000000000000	390.0000000000000000	390.0000000000000000	390.0000000032170240	390.0000000000000000	411.2548416301633000	390.0000000000000000	390.0000000000000000
	Mean	390.5315438816460000	390.0000000000001100	390.0000000000011900	398.2000147834449103	392.5883754233423700	414.6731243841997500	413.2364746591623500	390.3081451255193200
	Std	2.50007412834795167	0.0000000000000568	0.0000000000018808	16.2217059831472681	3.1053936255249658	2.4370774444214609	29.7345673904860000	0.47905185297883029
	Succ	26.67	50	30	0	26.67	0	30	40
	Time	59.432	28.733	57.149	77.319	62.439	102.734	116.892	60.739
f_{24}	Min	-187.9511478720692100	-328.0100818858134100	-326.0201637716268100	-327.7654123597851270	-312.8425436863587900	-283.7415542110191800	-311.789711627220061	-310.7873572616098800
	Mean	-170.8132987651011100	-326.7663830644468100	-324.0302481763498000	-323.9710235600048743	-294.7985918252179500	-157.3753555505669500	-297.620443705256010	-305.2082580888827000
	Std	10.1551136331903620	1.2924897287523103	1.2185689457394560	4.00000513791587136	12.2729097988707100	37.1250046662017720	32. 4838827283423496	7.3223833415277388
	Succ	0	0	0	0	0	0	0	0
	Time	193.0771	79.196	81.979	86.221	67.836	86.720	98.316	70.469

Table 11. Results of shifted and rotated benchmark functions, $n = 30$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_{21}	Min	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000
	Mean	-450.0000000000000000	-449.9999999999999400	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000
	Std	0.0000000000000000	0.0000000000000508	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
	Succ	100	50	100	100	100	100	100	**100**
	Time	134.515	34.837	40.705	44.803	46.157	46.330	75.084	47.028
f_{22}	Min	-450.0000000000000000	-449.9999999999996600	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-449.2047023577194400	-450.0000000000000000
	Mean	-450.0000000000000000	-449.9999999999995500	-450.0000000000000000	-450.0000000000000000	-450.0000000000000000	-449.9923244886244900	-446.9325426695196500	-450.0000000000000000
	Std	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0044382733553957	3.3286174731900031	0.0000000000000000
	Succ	100	0	100	100	100	50	0	**100**
	Time	153.189	36.635	65.375	110.178	83. 923	70.521	102.853	67.147
f_{23}	Min	923.3122301231220500	390.2651577762746900	390.0433169643209200	390.0000000032170240	390.1349684072680000	406.1913933149286300	390.0000000774123800	390.0000682074815500
	Mean	6004.1402501951288000	390.2651577762746900	425.3330809945244400	398.2000147834449103	395.7864367244400000	458.9883748242529000	390.8192072680831200	391.7998112680409700
	Std	5594.7482946023929000	0.0000000000000000	34.9323508668616260	16.2217059831472681	7.3178607152460000	27.3512160883286800	1.5914652405331551	1.7962980906277657
	Succ	0	0	0	0	0	0	0	0
	Time	195.009	152.979	116.520	144.319	66.613	95.125	120.633	60.548
f_{24}	Min	-262.0000858641211600	-290.2016528297256200	-223.5396434040115400	-215.5799000531476400	-239.1246610103919700	-224.5346074989227500	-288.4640698301597000	-292.7043735974306200
	Mean	-236.2931389112091000	-281.3981485447598100	-207.2224344191213800	-194.4871539557644800	-230.0882309073202200	-206.8741753507984000	-235.2322820353835500	-271.0274687832123800
	Std	15.4167083463846360	5.3345602587061283	14.3714838213930720	20.3456873003582500	5.2380481185273311	16.5798203539693250	28.3769846149553120	13.3149257435571790
	Succ	0	0	0	0	0	0	0	0
	Time	176.420	120.380	101.025	89.199	115.902	132.213	246.338	107.349

Table 12. Results of shifted and rotated benchmark functions, $n = 50$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_{25}	Min	234.2514754322145003	120.0000000000000000	120.0000000000000000	120.0000000000000000	120.0000000000000000	120.0000000000000000	120.0000000000000000	120.0000000000000000
	Mean	455.0851123744125896	161.5045411507748800	231.1247852578964788	415.4627658246898124	251.2678634156410020	301.8803634401870700	247.1679437402640000	220.4566396417482100
	Std	167.1554863214578955	40.8123547862124458	152.1047684571247851	146.2245789214568549	157.3938282341687970	182.2785372619515600	163.8506305541810000	134.5789554563156700
	Succ	0	33.34	10	16.67	40	10	20	40
	Time	158.217	195.247	156.294	202.541	175.519	168.153	315.110	170.467
f_{26}	Min	120.0000000000000000	221.9559596030777900	213.8866174069701500	178.1500148357924853	238.3285405884752200	216.4603102871090000	138.1239330780676900	230.4697363886152700
	Mean	221.6635496595235100	228.3093455824848900	216.0670953318125700	230.0043894117463852	253.7391338926102300	328.8991571751680000	215.0891175978302400	250.0590278771736600
	Std	40.4793070004110830	4.4197947468679075	2.7755155824340010	16.66157851467941340	17.0097061443176365	169.3910734871141100	110.4370379004781702	17.1082798514738560
	Succ	20	0	0	0	0	0	0	0
	Time	747.772	120.440	113.098	156.571	129.621	106.391	226.638	113.071
f_{27}	Min	310.0000000000000000	310.0000000000000000	310.0000000000000000	310.0000000000000000	310.0000000000000000	310.0000000000000000	310.0000000000000000	310.0000000000000000
	Mean	855.5086440559429100	310.0000000000000000	810.0000154320209700	539.9127648531200066	310.0000000000000000	760.1441063457674500	921.4554355013506200	310.0000000000000000
	Std	166.7753968101395050	0.0000000000000000	120.4615734891640092	139.4157431958731500	0.0000000000000000	237.9472146523420480	203.5667647189327747	0.0000000000000000
	Succ	20	100	30	20	100	20	16.67	100
	Time	400.744	180.943	250.709	248.043	195.573	174.396	328.6237	180.042
f_{28}	Min	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000	460.0000000000000000
	Mean	639.1997430436929300	496.3517246697841250	486.0000000301876248	470.93881475621897413	460.0000000000000000	501.1324892514786200	610.3068519876720800	460.0000000000000000
	Std	115.4619998211036105	51.1203578914876000	34.6021574698525521	45.02178931580041978	0.0000000000000000	130.3485147618954797	150.2745692476307200	0.0000000000000000
	Succ	33.34	56.67	63.34	20	100	50	23.34	100
	Time	251.472	95.099	183.511	163.150	159.221	145.307	215.507	158.921

Table 13. Results of hybrid benchmark functions, $n = 30$

Problem	Statistics	CMA-ES	JDE	SADE	PSO2011	Election algorithm	SELO	CSSA	CEA
f_{25}	Min	396.6452204913950900	320.000000000000000	421.0505168707150000	439.9360251174790600	443.0833104620444400	330.5889241782028900	358.1087818436275300	365.7198427286517100
	Mean	480.3266429050074700	422.4652412915387500	496.0041304732940300	487.3270263606730700	515.7121551751708900	482.1177848356405800	374.5817626615117000	638.1043872977845700
	Std	91.3460815719953100	100.3165747092249400	43.2914263451065300	39.9836893568482590	41.9448125432753680	75.7644303287188450	15.6650035501275000	200.0575903312088700
	Succ	0	0	0	0	0	0	0	0
	Time	491.786	298.295	319.425	307.517	350.546	282.164	361.275	356.520
f_{26}	Min	120.0000000000000000	217.1900815567871900	199.9382210829309100	278.2154218390603000	190.8414908573395500	184.0157761153815800	149.1922042052661900	192.7732803163116300
	Mean	196.7110237013338900	252.6683601225867900	223.9699491162265000	327.0597704682400000	206.0035312273601000	198.5561626310519000	230.9178687302762700	200.0198512342537400
	Std	57.1921651574837780	32.3475224810199489	35.5713239484792610	52.5087126644775070	9.0692872306153571	12.6271827756316280	50.3369274848248230	6.4160931595084012
	Succ	10	0	0	0	0	0	0	0
	Time	931.186	185.062	159.9185	154.903	209.122	171.186	328.580	198.559
f_{27}	Min	850.4272847542918000	896.3547537249504600	873.7341022604559800	866.9770245102914700	851.3122051250039700	850.7138279453589600	855.0006511545613000	852.2913959787829300
	Mean	987.1374598307908200	905.2151711585196400	888.0783375694557000	894.4624097928066200	916.0776838272911400	915.4467335913683400	910.1927459268479100	912.8355132527331100
	Std	157.9085076379647732	4.5521655742118154	12.6807771772059020	16.3342488132452140	39.9225774311974000	55.2168825407607000	187.0135958016884677	29.1706442006170550
	Succ	0	0	0	0	0	0	0	0
	Time	582.924	240.531	290.079	285.772	331.805	317.105	343.0724	316.988
f_{28}	Min	460.1038377611588400	460.0000000000033000	460.0000000000001100	460.0000000000034100	460.1446074981029000	460.0000000000070500	460.0000000000005630	460.0000000000006900
	Mean	533.3414369696136000	460.0000000000033500	482.3554625773845000	507.3288440371169400	608.3316183481684000	664.4400697095233000	660.6319018127858300	556.5771419380732800
	Std	50.9414908921000604	0.0000000000000284	19.9666695102418400	126.3417283082863500	111.4992852066428700	109.4638592242058600	201.7638036255714800	88.3115525305439280
	Succ	0	0	0	0	0	0	0	0
	Time	322.232	115.933	203.169	187.480	183.390	173.537	239.971	175.477

Table 14. Results of hybrid benchmark functions, $n = 50$

Comparison	T+	T-	*p*-value	Winner
CEA vs. CMA-ES	39	6	0.02852	CEA
CEA vs. JDE	7	3	-*	CEA
CEA vs. SADE	11	10	-	CEA
CEA vs. PSO2011	30	15	0.2040	CEA
CEA vs. Election algorithm	11	4	-	CEA
CEA vs. SELO	17	4	-	CEA
CEA vs. CSSA	13	8	-	CEA

* Symbol "-" means that it is not possible to calculate an accurate *p*-value

Table 15. Multi-problem based statistical pairwise comparison of CEA and comparison algorithms using two-sided Wilcoxon Signed-Rank test ($\alpha = 0.05$), $n = 30$

Comparison	T+	T-	*p*-value	Winner
CEA vs. CMA-ES	101	19	0.0114	CEA
CEA vs. JDE	33	12	0.01928	CEA
CEA vs. SADE	52	3	0.00298	CEA
CEA vs. PSO2011	103	2	0.00028	CEA
CEA vs. Election algorithm	57	21	0.08726	CEA
CEA vs. SELO	53	25	0.0466	CEA
CEA vs. CSSA	77	28	0.06876	CEA

Table 16. Multi-problem based statistical pairwise comparison of CEA and comparison algorithms using two-sided Wilcoxon Signed-Rank test ($\alpha = 0.05$), $n = 50$

6 CONCLUSION

This paper presents a Chaotic Election Algorithm (CEA) to improve the original election algorithm. The CEA enhances the election algorithm threefold:

1. modifying party formation phase,
2. introducing migration operator, and
3. introducing a chaotic positive advertisement.

CEA by modifying the party formation phase through eliminating the Euclidean distance computation from the process increases the speed of the algorithm. With the migration operator, diversity in the population is maintained, what keeps the CEA away from converging too fast before exploring the entire solution space. With the new chaotic positive advertisement, the information exchanges between candidates and voters efficiently and improves the algorithm's search ability. To show the performance of the CEA algorithm, it is evaluated on 28 optimization benchmarks and compared with CMA-EA, JDE, SADE, PSO2011, SELO, CSSA and election algorithm. The results show that the proposed CEA algorithm outperforms the canonical election algorithm and other comparable counterparts in terms of solution quality and convergence speed. There remain several points to improve our research. First, the CEA will be trapped in local optimums on few functions, which can be seen from simulation results on some benchmark functions. We can combine the CEA with some local search strategies or other meta-heuristics to further enhance its optimization ability. Second, we can apply the proposed CEA algorithm to solve more practical optimization problems to accurately identify its weaknesses and merits. Third, in some specific engineering applications, some components of the algorithm can be modified in order to improve the performance of the algorithm.

REFERENCES

[1] Sotoudeh-Anvari, A.—Hafezalkotob, A.: A Bibliography of Metaheuristics – Review from 2009 to 2015. International Journal of Knowledge-Based and Intelligent Engineering Systems, Vol. 22, 2018, No. 1, pp. 83–95, doi: 10.3233/KES-180376.

[2] Gogna, A.—Tayal, A.: Metaheuristics: Review and Application. Journal of Experimental and Theoretical Artificial Intelligence, Vol. 25, 2013, No. 4, pp. 503–526, doi: 10.1080/0952813X.2013.782347.

[3] Boussaïd, I.—Lepagnot, J.—Siarry, P.: A Survey on Optimization Metaheuristics. Information Sciences, Vol. 237, 2013, pp. 82–117, doi: 10.1016/j.ins.2013.02.041.

[4] Hussain, K.—Mohd Salleh, M. N.—Cheng, S.—Shi, Y.: Metaheuristic Research: A Comprehensive Survey. Artificial Intelligence Review, Vol. 52, 2019, pp. 2191–2233, doi: 10.1007/s10462-017-9605-z.

[5] Beheshti, Z.—Shamsuddin, S. M.: A Review of Population-Based Meta-Heuristic Algorithms. International Journal of Advances in Soft Computing and Its Applications, Vol. 5, 2013, No. 1, pp. 1–35.

[6] Alsalibi, B.—Venkat, I.—Subramanian, K. G.—Lufti, S. L.—De Wilde, P.: The Impact of Bio-Inspired Approaches Toward the Advancement of Face Recognition. ACM Computing Surveys (CSUR), Vol. 48, 2015, No. 1, Art. No. 5, 31 pp., doi: 10.1145/2791121.

[7] José-García, A.—Gómez-Flores, W.: Automatic Clustering Using Nature-Inspired Metaheuristics: A Survey. Applied Soft Computing, Vol. 41, 2016, pp. 192–213, doi: 10.1016/j.asoc.2015.12.001.

[8] Engelbrecht, A. P.: Computational Intelligence: An Introduction. 2nd Edition. Wiley, New York, 2007.

[9] Haupt, R. L.—Haupt, S. E.: Practical Genetic Algorithms. 2nd Edition. Wiley, New York, 2004, doi: 10.1002/0471671746.

[10] Civicioglu, P.: Backtracking Search Optimization Algorithm for Numerical Optimization Problems. Applied Mathematics and Computation, Vol. 219, 2013, No. 15, pp. 8121–8144, doi: 10.1016/j.amc.2013.02.017.

[11] Karaboga, D.—Akay, B.: A Comparative Study of Artificial Bee Colony Algorithm. Applied Mathematics and Computation, Vol. 214, 2009, No. 1, pp. 108–132, doi: 10.1016/j.amc.2009.03.090.

[12] Emami, H.—Derakhshan, F.: Election Algorithm: A New Socio-Politically Inspired Strategy. AI Communications, Vol. 28, 2015, pp. 591–603, doi: 10.3233/AIC-140652.

[13] Kennedy, J.—Eberhart, R.: Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks (ICNN '95), Perth, WA, Australia, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[14] Ghaemi, M.—Feizi-Derakhshi, M.-R.: Forest Optimization Algorithm. Expert Systems with Applications, Vol. 41, 2014, No. 15, pp. 6676–6687, doi: 10.1016/j.eswa.2014.05.009.

[15] Shah-Hosseini, H.: Problem Solving by Intelligent Water Drops. 2007 IEEE Congress on Evolutionary Computation (CEC 2007), 2007, pp. 3226–3231, doi: 10.1109/CEC.2007.4424885.

[16] Kaveh, A.—Talatahari, S.: A Novel Heuristic Optimization Method: Charged System Search. Acta Mechanica, Vol. 213, 2010, pp. 267–289, doi: 10.1007/s00707-009-0270-4.

[17] Hatamlou, A.: Black Hole: A New Heuristic Optimization Approach for Data Clustering. Information Sciences, Vol. 222, 2013, pp. 175–184, doi: 10.1016/j.ins.2012.08.023.

[18] Tayarani, M.—Akbarzadeh, M.: Magnetic-Inspired Optimization Algorithms: Operators and Structures. Swarm and Evolutionary Computation, Vol. 19, 2014, pp. 82–101, doi: 10.1016/j.swevo.2014.06.004.

[19] Biswas, A.—Mishra, K. K.—Tiwari, S.—Misra, A. K.: Physics-Inspired Optimization Algorithms: A Survey. Journal of Optimization, Vol. 2013, 2013, Art. No. 438152, 16 pp., doi: 10.1155/2013/438152.

[20] Emami, H.—Derakhshan, F.: Integrating Fuzzy K-Means, Particle Swarm Optimization and Imperialist Competitive Algorithm for Data Clustering. Arabian Journal for Science and Engineering, Vol. 40, 2015, pp. 3545–3554, doi: 10.1007/s13369-015-1826-3.

[21] Lokhande, N. M.—Pujeri, R. V.: Novel Image Segmentation Using Particle Swarm Optimization. Proceedings of the 2018 8th International Conference on Biomedical Engineering and Technology (ICBET 2018), ACM, 2018, pp. 46–50, doi: 10.1145/3208955.3208962.

[22] Guerrero, M.—Montoya, F. G.—Baños, R.—Alcayde, A.—Gil, C.: Adaptive Community Detection in Complex Networks Using Genetic Algorithms. Neurocomputing, Vol. 266, 2017, pp. 101–113, doi: 10.1016/j.neucom.2017.05.029.

[23] Emami, H.—Derakhshan, F.: A New Method for Conflict Detection and Resolution in Air Traffic Management. Twenty-Sixth AAAI Conference on Artificial Intelligence @ Semantic Cities, 22–26 July 2012, Toronto, Canada, AAAI Press, pp. 37–40.

[24] Wolpert, D.—Macready, W. G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation, Vol. 1, 1997, No. 1, pp. 67–82, doi: 10.1109/4235.585893.

[25] Vesterstrøm, J.—Thomsen, R.: A Comparative Study of Differential Evolution Particle Swarm Optimization and Evolutionary Algorithms on Numerical Benchmark Problems. Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004), IEEE, Vol. 2, 2004, pp. 1980–1987, doi: 10.1109/CEC.2004.1331139.

[26] Derrac, J.—García, S.—Molina, D.—Herrera, F.: A Practical Tutorial on the Use of Non-Parametric Statistical Tests as a Methodology for Comparing Evolu-

tionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation, Vol. 1, 2011, No. 1, pp. 3–18, doi: 10.1016/j.swevo.2011.02.002.

[27] Liu, B.—Wang, L.—Jin, Y.-H.—Tang, F.—Huang, D.-X.: Improved Particle Swarm Optimization Combined with Chaos. Chaos, Solitons and Fractals, Vol. 25, 2005, No. 5, pp. 1261–1271, doi: 10.1016/j.chaos.2004.11.095.

[28] Erol, O.—Eksin, I.: A New Optimization Method: Big Bang – Big Crunch. Advances in Engineering Software, Vol. 37, 2006, No. 2, pp. 106–111, doi: 10.1016/j.advengsoft.2005.04.005.

[29] Wang, G.-G.—Guo, L.—Gandomi, A. H.—Hao, G.-S.—Wang, H.: Chaotic Krill Herd Algorithm. Information Sciences, Vol. 274, 2014, pp. 17–34, doi: 10.1016/j.ins.2014.02.123.

[30] Gao, S.—Vairappan, C.—Wang, Y.—Cao, Q.—Tang, Z.: Gravitational Search Algorithm Combined with Chaos for Unconstrained Numerical Optimization. Applied Mathematics and Computation, Vol. 231, 2014, pp. 48–62, doi: 10.1016/j.amc.2013.12.175.

[31] Alatas, B.: Chaotic Bee Colony Algorithms for Global Numerical Optimization. Expert Systems with Applications, Vol. 37, 2010, No. 8, 2010, pp. 5682–5687, doi: 10.1016/j.eswa.2010.02.042.

[32] Saremi, S.—Mirjalili, S.—Lewis, A.: Biogeography-Based Optimisation with Chaos. Neural Computing and Applications, Vol. 25, 2014, pp. 1077–1097, doi: 10.1007/s00521-014-1597-x.

[33] Talatahari, S.—Kaveh, A.—Sheikholeslami, R.: Chaotic Imperialist Competitive Algorithm for Optimum Design of Truss Structures. Structural and Multidisciplinary Optimization, Vol. 46, 2012, No. 3, pp. 355–367, doi: 10.1007/s00158-011-0754-4.

[34] Yu, H.—Yu, Y.—Liu, Y.—Wang, Y.—Gao, S.: Chaotic Grey Wolf Optimization. Proceedings of the 2016 IEEE International Conference on Progress in Informatics and Computing (PIC 2016), 2017, pp. 108–113, doi: 10.1109/PIC.2016.7949476.

[35] Storn, R.—Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, Vol. 11, 1997, pp. 341–359, doi: 10.1023/A:1008202821328.

[36] Jia, D.—Zheng, G.—Khan, M. K.: An Effective Memetic Differential Evolution Algorithm Based on Chaotic Local Search. Information Sciences, Vol. 181, 2011, No. 15, pp. 3175–3187, doi: 10.1016/j.ins.2011.03.018.

[37] Dorigo, M.—Birattari, M.—Stützle, T.: Ant Colony Optimization. IEEE Computational Intelligence Magazine, Vol. 1, 2006, No. 4, pp. 28–39, doi: 10.1109/MCI.2006.329691.

[38] Cai, J.—Ma, X.—Li, L.—Yang, Y.—Peng, H.—Wang, X.: Chaotic Ant Swarm Optimization to Economic Dispatch. Electric Power Systems Research, Vol. 77, 2007, No. 10, pp. 1373–1380, doi: 10.1016/j.epsr.2006.10.006.

[39] Karaboga, D.—Basturk, B.: A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. Journal of Global Optimization, Vol. 39, 2007, No. 3, pp. 459–471, doi: 10.1007/s10898-007-9149-x.

[40] Atashpaz-Gargari, E.—Lucas, C.: Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition. 2007 IEEE Congress on Evolutionary Computation (CEC 2007), 2007, pp. 4661–4667, doi: 10.1109/CEC.2007.4425083.

[41] Talatahari, S.—Farahmand Azar, B.—Sheikholeslami, R.—Gandomi, A. H.: Imperialist Competitive Algorithm Combined with Chaos for Global Optimization. Communications in Nonlinear Science and Numerical Simulation, Vol. 17, 2012, No. 3, pp. 1312–1319, doi: 10.1016/j.cnsns.2011.08.021.

[42] Simon, D.: Biogeography-Based Optimization. IEEE Transactions on Evolutionary Computation, Vol. 12, 2008, No. 6, pp. 702–713, doi: 10.1109/TEVC.2008.919004.

[43] Rashedi, E.—Nezamabadi-pour, H.—Saryazdi, S.: GSA: A Gravitational Search Algorithm. Information Sciences, Vol. 179, No. 13, 2009, pp. 2232–2248, doi: 10.1016/j.ins.2009.03.004.

[44] Yang, X.-S.: A New Meta-Heuristic Bat-Inspired Algorithm. In: González, J. R., Pelta, D. A., Cruz, C., Terrazas, G., Krasnogor, N. (Eds.): Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer, Berlin, Heidelberg, Studies in Computational Intelligence, Vol. 284, 2010, pp. 65–74, doi: 10.1007/978-3-642-12538-6_6.

[45] Rezaee Jordehi, A.: Chaotic Bat Swarm Optimisation (CBSO). Applied Soft Computing, Vol. 26, 2015, pp. 523–530, doi: 10.1016/j.asoc.2014.10.010.

[46] Yang, X.-S.—Deb, S.: Engineering Optimisation by Cuckoo Search. International Journal of Mathematical Modelling and Numerical Optimisation (IJMMNO), Vol. 1, 2010, No. 4, pp. 330–343, doi: 10.1504/IJMMNO.2010.035430.

[47] Wang, G.-G.—Deb, S.—Gandomi, A. H.—Zhang, Z.—Alavi, A. H.: Chaotic Cuckoo Search. Soft Computing, Vol. 20, 2016, pp. 3349–3362, doi: 10.1007/s00500-015-1726-1.

[48] Yang, X.-S.: Firefly Algorithm, Stochastic Test Functions and Design Optimisation. International Journal of Bio-Inspired Computation, Vol. 2, 2010, No. 2, pp. 78–84, doi: 10.1504/IJBIC.2010.032124.

[49] Gandomi, A. H.—Yang, X.-S.—Talatahari, S.—Alavi, A. H.: Firefly Algorithm with Chaos. Communications in Nonlinear Science and Numerical Simulation, Vol. 18, 2013, No. 1, pp. 89–98, doi: 10.1016/j.cnsns.2012.06.009.

[50] Thangaraj, R.—Pant, M.—Abraham, A.—Bouvry, P.: Particle Swarm Optimization: Hybridization Perspectives and Experimental Illustrations. Applied Mathematics and Computation, Vol. 217, 2011, No. 12, pp. 5208–5226, doi: 10.1016/j.amc.2010.12.053.

[51] Alatas, B.—Akin, E.—Ozer, A. B.: An Adaptive Chaos Embedded Particle Swarm Optimization Algorithm. Chaos, Solitons and Fractals, Vol. 40, 2009, No. 4, pp. 1715–1734, doi: 10.1016/j.chaos.2007.09.063.

[52] Gandomi, A. H.—Alavi, A. H.: Krill Herd: A New Bio-Inspired Optimization Algorithm. Communications in Nonlinear Science and Numerical Simulation, Vol. 17, 2012, No. 12, pp. 4831–4845, doi: 10.1016/j.cnsns.2012.05.010.

[53] Mirjalili, S.—Mirjalili, S. M.—Lewis, A.: Grey Wolf Optimizer. Advances in Engineering Software, Vol. 69, 2014, pp. 46–61, doi: 10.1016/j.advengsoft.2013.12.007.

[54] Kohli, M.—Arora, S.: Chaotic Grey Wolf Optimization Algorithm for Constrained Optimization Problems. Journal of Computational Design and Engineering, Vol. 5, 2018, No. 4, pp. 458–472, doi: 10.1016/j.jcde.2017.02.005.

[55] Mirjalili, S.—Gandomi, A. H.—Mirjalili, S. Z.—Saremi, S.—Faris, H.—Mirjalili, S. M.: Salp Swarm Algorithm: A Bio-Inspired Optimizer for Engineering Design Problems. Advances in Engineering Software, Vol. 114, 2017, pp. 163–191, doi: 10.1016/j.advengsoft.2017.07.002.

[56] Sayed, G. I.—Khoriba, G.—Haggag, M. H.: A Novel Chaotic Salp Swarm Algorithm for Global Optimization and Feature Selection. Applied Intelligence, Vol. 48, 2018, No. 10, pp. 3462–3481, doi: 10.1007/s10489-018-1158-6.

[57] Yaghoobi, S.—Mojallali, H.: Tuning of a PID Controller Using Improved Chaotic Krill Herd Algorithm. Optik, Vol. 127, 2016, No. 11, pp. 4803–4807, doi: 10.1016/j.ijleo.2016.01.055.

[58] Suganthan, P. N.—Hansen, N.—Liang, J. J.— Deb, K.—Chen, Y. P.—Auger, A.—Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical Report. Nanyang Technological University, Singapore, 2005, `http://www.ntu.edu.sg/home/EPNSugan`.

[59] Chen, Q.—Liu, B.—Zhang, Q.—Liang, J.: Evaluation Criteria for CEC 2015 Special Session and Competition on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization. Singapore, 2015. `http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/SharedDocuments/CEC-2015/ExpensiveProblems/CEC15-singleobjectiveoptimizationcompetition-expensive.pdf`.

[60] Igel, C.—Hansen, N.—Roth, S.: Covariance Matrix Adaptation for Multi-Objective Optimization. Evolutionary Computation, Vol. 15, 2007, No. 1, pp. 1–28, doi: 10.1162/evco.2007.15.1.1.

[61] Qin, A. K.—Suganthan, P. N.: Self-Adaptive Differential Evolution Algorithm for Numerical Optimization. 2005 IEEE Congress on Evolutionary Computation, Vol. 2, 2005, pp. 1785–1791, doi: 10.1109/CEC.2005.1554904.

[62] Brest, J.—Greiner, J.—Boskovic, B.—Mernik, M.—Zumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on Evolutionary Computation, Vol. 10, 2006, No. 6, pp. 646–657, doi: 10.1109/TEVC.2006.872133.

[63] Kumar, M.—Kulkarni, A. J.—Satapathy, S. C.: Socio Evolution and Learning Optimization Algorithm: A Socio-Inspired Optimization Methodology. Future Generation Computer Systems, Vol. 81, 2018, pp. 252–272, doi: 10.1016/j.future.2017.10.052.

[64] Liang, J. J.—Qu, B. Y.—Suganthan, P. N.—Hernández-Díaz, A. G.: Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization. Technical Report 201212, Computational

Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2013.

Hojjat Emami received his B.Sc. degree in software engineering and his M.Sc. degree in artificial intelligence, both from the University of Tabriz, Iran. He received the Ph.D. degree in artificial intelligence from the Malek-Ashtar University of Technology, Tehran, Iran. His research interests are multi-agent systems, data mining, machine learning and optimization.