Advances in Engineering Software 92 (2016) 65–88

Contents lists available at ScienceDirect

Advances in Engineering Software

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

A novel nature-inspired algorithm for optimization: Virus colony search

Mu Dong Li*, Hui Zhao, Xing Wei Weng, Tong Han

Institute of Aeronautics and Astronautics Engineering, Air Force Engineering University, Xi'an, Shannxi 710038, PR China

ARTICLE INFO

Article history:
Received 23 June 2015
Revised 3 November 2015
Accepted 6 November 2015
Available online 10 December 2015

Keywords:
Nature-inspired algorithms
Optimization
Virus colony search
Survival strategy
Unconstrained optimization problems
Engineering design problems

ABSTRACT

This paper presents a new powerful nature-inspired method named Virus Colony Search (VCS). VCS simulates diffusion and infection strategies for the host cells adopted by virus to survive and propagate in the cell environment. With the strategies, the individual in the new algorithm explores and exploits the search space more efficiently. To verify the performance of our VCS, both the unconstrained classic and CEC2014 modern benchmark functions, and constrained engineering design optimization problems are employed. The experimental results, considering both convergence and accuracy simultaneously, demonstrate the effectiveness of VCS for global numerical and engineering optimization problems.

1. Introduction

Optimization is the process of searching for the global optima of a problem under a given circumstance. Lots of real-world complex optimization problems have emerged in many scientific fields such as engineering, economics and business which cannot be solved with a reasonable time or accuracy solutions by classical methods [1].

Nature provides a rich source such as mechanisms, principles and concepts for designing artificial computational methods to solve such complex optimization problems. In recent years, researchers have developed many nature-inspired (NI) algorithms that mimic the specific phenomena or behaviors of nature for the different optimization problems [2]. For example, genetic algorithm (GA) proposed by Holland [3] represents a fairly abstract model of Darwinian evolution and biological genetics; particle swarm optimization (PSO) conceived by Kennedy and Eberhart [4] mimics social behaviors of bird flocking; ant colony optimization (ACO), proposed by Marco Dorigo [5], is inspired by the foraging behaviors of the ant colonies. Compared with the classical heuristic methods, these algorithms have been confirmed with the excellent performance, especially for optimizing the multimodal, non-differentiable and discrete complex optimization problems. Meanwhile, these NIs have been widely used in many scientific fields such as dynamic optimization [6], IIR filter design [7], image processing [8], mechanical design problems [9], task scheduling [10], data mining applications [11], and many other engineering problems [12–14].

Generally, NIs can be divided into three main classes: evolutionary algorithms (EA), swarm intelligence (SI) algorithms and physics-based (PB) algorithms [2]. EAs are inspired by the genetic and evolutionary behaviors of creatures. GA and differential evolution (DE) [15] algorithm can be described as representative algorithms in EAs. The second main class of nature-inspired algorithms is swarm intelligence algorithms. Just like EAs, the SIs are usually inspired by the behaviors of intelligent nature creatures. Different from the EAs, the majority of SIs use genetic rules only and they always take full advantages of each solution in search space to provide better solutions to the problem. Some of the popular SIs are PSO, artificial bee colony (ABC) inspired by the honey-bees food searching behavior [16], cuckoo search (CS) inspired by parasitic bio-interactions in living creatures [17], animal migration optimization (AMO) algorithm inspired by the behavior of the animal migration [18], grey wolf optimizer (GWO) inspired by hunting mechanism of grey wolves [19] and dolphin echolocation algorithm (DEa) inspired by the behaviors of navigation and hunting of dolphin [20]. Physics-based algorithms are the third class of nature-inspired algorithms. Distinct from the EAs and SIs, the PBs are mostly inspired by physical rules in the nature. There are some popular PBs such as charged system search (CSS) [21], ray optimization (RO) [22], colliding bodies optimization (CBO) [23] and gravitational search algorithm (GSA) algorithm [24].These NIs share the same following characteristics: (1) make use of random variables; (2) do not require substantial gradient information; (3) have several parameters need to be fitted for the problem [25]. Each NI algorithm has unique advantages with respect to robustness and performance in noisy environments, in the presence of uncertain parameters, and in different problem spaces [26]. However, according to the "no free-lunch" theorem, there is no one NI algorithm to optimally solve all optimizing problems [27]. Therefore, new

* Corresponding author. Tel.: +86 18392193176.
E-mail addresses: modern_lee@163.com (M.D. Li), zhaohui_kgy@163.com (H. Zhao), wengxingwei_kgy@163.com (X.W. Weng), hantong_kgy@163.com (T. Han).

http://dx.doi.org/10.1016/j.advengsoft.2015.11.004
0965-9978/

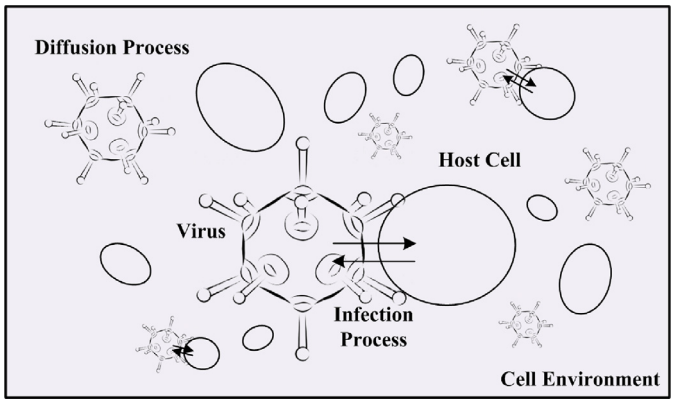


Fig. 1. The diagram for general growth of viruses in the cell environment.

high-performance NI algorithms are continuously needed to solve specific optimizing problems.

This paper introduces a new simple and powerful NI algorithm called Virus Colony Search (VCS).This algorithm simulates the virus infection and diffusion strategies for the host cells to survive and propagate in the cell environment. The main contribution of this paper is the presentation of a new algorithm with further insight into solving optimization problems. The proposed algorithm can achieve a solution that has the least (or at most, a small) error compared with the globally optimum solution within a minimal number of iterations, thus offering an improvement in terms of accuracy, convergence rate and simplicity of the operations.

Preliminary studies show that the NIs such as CMA-ES, ABC, CS, GSA, CoBiDE, GWO, AMO, SOS and other well-known optimization algorithms are very promising and could be used as the compared algorithms for evaluating VCS's performance in solving unconstrained global optimization problems. For constrained problems, VCS is compared with the results in the previous lectures of different algorithms.

The rest of this paper is organized as follows. Section 2 introduces the survival strategies of viruses. The detailed introduction of VCS is described in Section 3. The experiment sets and statistical results for different tests are shown in Section 4. Finally, Section 5 concludes the work and suggests some directions for the future.

2. Survival strategy of viruses

The proposed VCS algorithm simulates the infection and diffusion behaviors between viruses and host cells in the cell environment. In order to live and propagate, virus has to live on a host cell by diffusion and infection behaviors. Meanwhile, evolution behavior always occurs during the process of adapting to a changing cell environment. The following part in this section introduces the survival strategies of virus colony and describes the roles of virus, host cell and immune system.

A virus is a small infectious agent that survives only inside the living cells of other organisms. Viruses are considered by some researchers to be a life form, because they carry genetic material, reproduce, and evolve through natural selection [28]. The origins of viruses in the evolutionary history of life are unclear, but their growth generally can be summarized as two processes in Fig. 1: viruses diffusion and host cells infection. Meanwhile, the evolution and immune response of the host immune system are also occurred along with above two processes.

(1) Viruses diffusion: viruses randomly search for host cells to infect so as to absorb essential elements for growth (Diffusion process in Fig. 1). In this process, the random walk method of Gaussian walks [29] can be used to describe this behavior.

(2) Host cells infection: When found a host cell, virus infects and destroys it (Infection process in Fig. 1). Based on the essential elements of the host cell, virus can survive and reproduce itself until the death of the host cell. In other words, the host cell is "mutated" to be a virus with the process of reproduction. Inspired by evolutionary strategies with covariance matrix adaptation (CMA-ES) algorithm [30], this process is realized according to the CMA-ES operation to generate the new viruses.

(3) Immune response: Because the host immune system plays a major role to protect the host cell from infection or destruction, the virus will be selected during the growing process in order to survive. In other words, viruses that have better ability will be reserved for the next generation; otherwise they will be killed by the host immune system. It means evolution chance of the viruses which have not obtained a good ability will increase.

3. Virus colony search

According to the previous section, our Virus Colony Search algorithm employs three strategies including: Gaussian walks method for viruses diffusion, CMA-ES operation for host cells infection and evolution strategy for immune response. Actually, the first strategy is used to improve the exploitation properties. The second strategy is mainly to enhance the performance of exploration. For the third strategy, it is proposed to make full use of worse individuals so as to improve the search efficiency. Loosely speaking, VCS uses five simple rules to find a solution.

1. Two different groups: virus colony V_{pop} and host cell colony H_{pop}, are used in the VCS.
2. Each virus in the diffusion process creates a new random individual.
3. Each virus infects one host cell.
4. The reproduction of each virus is based on destroying the host cell to obtain nutrients.
5. According to the protection of the host immune system, only some of the best viruses remain in each generation, and the rest of the viruses are evolved so as to survive.

3.1. Viruses diffusion

In general, viruses exist in some certain mediums such as air, water or circulation systems of some organisms. The random-walking phenomenon occurred in some mediums is the main activity of a virus in this stage until it searched a host cell. Gaussian random walk, which has a promising performance in finding global optimum [31], is often used as the diffusion rule to describe the phenomenon of random diffusion [32]. For improving the exploitation performance and avoiding local optimal, this method is selected and the viruses diffusion is designed as:

$$Vpop'_i = \text{Gaussian}(G^g_{best}, \tau) + (r_1 \cdot G^g_{best} - r_2 \cdot Vpop_i) \quad (1)$$

where i is the random index selected from [1,2,3,..., N] and N is the population size. G^g_{best} is the best solution of the generation g. r_1 *and* r_2 are random values selected from [0, 1]. For the Gaussian parameters, the standard deviation ι is calculated by $\log(g)/g \cdot (Vpop_i - G^g_{best})$. In Eq. (1), the search direction $(r_1 \cdot G^g_{best} - r_2 \cdot Vpop_i)$ is used to avoid the local optimal which $Vpop_i$ is the ith current position of $Vpop$. And in order to improve the performance of local search, the term $\log(g)/g$ is designed so as to decrease the size of Gaussian jumps with the generation increased. From Eq. (1), it can be observed that the new individuals are generated around the global best solution by the Gaussian distribution Gaussian(G^g_{best}, τ) and adjusted by the search direction $(r_1 \cdot G^g_{best} - r_2 \cdot Vpop_i)$. On the other hand, with the

generation increased, the value of $\log(g)/g$ is decreased gradually. It leads to severe disturbance at the early generation while a slight disturbance at the late generation around the global best solution, which can help the individuals to get closer to the solutions.

3.2. Host cells infection

Once a host cell is infected, it will be invaded and destroyed by the virus until its death. Actually, this process can be explained as the process of interaction relations: the host cell provides the essential elements and the virus metabolizes the harmful substances leading to death of the host cell gradually. Finally, the host cell 'mutates' into a new virus. This process is mainly used to realize the information exchange and improve the exploration properties of the population. CMA-ES method is a famous evolutionary algorithm. It not only contains the mutation operation, but also considers the interactive relations between the individuals by covariance matrix adaptation (CMA), which adapts a full covariance matrix of a normal search (mutation) distribution [30]. Therefore this method is quite suitable for the behavior of the host cells infection and it is introduced in this paper. In order to reflect the interactive relations between the viruses and the host cells. The main steps of it can be summarized as follows.

Step. 1. Update the H_{pop} by:

$$Hpop_i^g = X_{mean}^g + \sigma_i^g \times \mathrm{N}_i(\mathbf{0}, \mathbf{C}^g) \tag{2}$$

where $\mathrm{N}_i(\mathbf{0}, \mathbf{C}^g)$ is a normal distribution with mean $\mathbf{0}$ and $D\times D$ covariance matrix $\mathbf{C}^g$, g is the current generation, D is the dimension of the problem and $\sigma^g > 0$ is the step size. X_{mean}^g is initialized by:

$$X_{mean}^0 = \frac{\sum_{i=1}^{N} Vpop_i}{N} \tag{3}$$

Step. 2. Select the best λ individuals from the previous phrase as a parental vector, and the center of the selected vector is calculated based on:

$$X_{mean}^{g+1} = \frac{1}{\lambda}\sum_{i=1}^{\lambda}\omega_i \cdot Vpop_i^{\lambda best} \mid \omega_i = \ln(\lambda+1)\Big/\left(\sum_{j=1}^{\lambda}(\ln(\lambda+1)-\ln(j))\right) \tag{4}$$

where $\lambda = \lfloor N/2 \rfloor$, ω_i is the recombination weight and i denotes the index of ith best individuals. Two so-called evolution paths are computed. They track the history of changes of the population mean with an exponential decay of the past.

$$p_\sigma^{g+1} = (1-c_\sigma)p_\sigma^g + \sqrt{c_\sigma(2-c_\sigma)\lambda_w}\frac{1}{\sigma^g}(\mathbf{C}^g)^{-1/2}(X_{mean}^{g+1} - X_{mean}^g) \tag{5}$$

$$p_c^{g+1} = (1-c_c)p_c^g + h_\sigma\sqrt{c_c(2-c_c)\lambda_w}\frac{1}{\sigma^g}(X_{mean}^{g+1} - X_{mean}^g) \tag{6}$$

where $\lambda_w^{-1} = \sum_{i=1}^{\lambda} w_i^2$, $(\mathbf{C}^g)^{-1/2}$ is symmetric, positive and satisfies $(\mathbf{C}^g)^{-1/2}(\mathbf{C}^g)^{-1/2} = (\mathbf{C}^g)^{-1}$. The cumulation parameters are generally set as $c_\sigma = (\lambda_w+2)/(N+\lambda_w+3)$, $c_c = 4/(N+4)$ and $h_\sigma = 1$ usually but $h_\sigma = 0$ if $\|p_\sigma^{g+1}\|$ is large [30].

Step. 3. Update the step size σ^{g+1} and the covariance matrix C^{g+1} by:

$$\sigma^{g+1} = \sigma^g \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{g+1}\|}{\mathrm{E}\|\mathrm{N}(0,\mathbf{I})\|} - 1\right)\right) \tag{7}$$

$$\mathbf{C}^{g+1} = (1-c_1-c_\lambda)\mathbf{C}^g + c_1 p_c^{g+1}(p_c^{g+1})^{\mathrm{T}} + c_\lambda\sum_{i=1}^{\lambda} w_i \frac{Vpop_i^{\lambda best} - X_{mean}^g}{\sigma^g}\cdot\frac{(Vpop_i^{\lambda best} - X_{mean}^g)^{\mathrm{T}}}{\sigma^g} \tag{8}$$

where $d_\sigma = 1 + c_\sigma + 2\max\{0, (\sqrt{\lambda_w - 1}/\sqrt{N+1}) - 1\}$ is usually close to 1 and c_1, c_λ obey

$$c_1 = \frac{1}{\lambda_w}\left(\left(1-\frac{1}{\lambda_w}\right)\min\left\{1, \frac{2\lambda_w - 1}{(N+2)^2 + \lambda_w}\right\} + \frac{1}{\lambda_w}\frac{2}{(N+\sqrt{2})^2}\right) \tag{9}$$

$$c_\lambda = (\lambda_w - 1)c_1 \tag{10}$$

where $0 \le c_\lambda \le 1$ is the updating rate for updating the covariance matrix C [30].

3.3. Immune response

As mentioned earlier, according to influence of the host cell immune system, the viruses with better performance are more likely to retain its ability to the next generation. However, worse viruses must be evolved themselves in case of being killed by the immune system. Thus, the following steps are used to realize the evolution of the viruses.

Step. 1. Calculate the performance ranks Pr according to fitness value of virus colony V_{pop} by:

$$Pr_{rank(i)} = \frac{(N-i+1)}{N} \tag{11}$$

where N is the population size of V_{pop}, $rank(i)$ means the fitness rank of the ith individual in V_{pop}.

Step. 2. Evolve the individuals of V_{pop} by:

$$\begin{cases} Vpop_{i,j}'' = Vpop_{k,j} - rand\cdot(Vpop_{h,j} - Vpop_{i,j}), & if\ r > Pr_{rank(i)} \\ Vpop_{i,j}'' = Vpop_{i,j}, & otherwise \end{cases} \tag{12}$$

where k, i, h are random indexes selected from [1, 2, 3, … , N], $i\neq k\neq h$ and $j \in [1, 2, 3, \ldots, d]$. *rand* and r are the random values chosen from the [0, 1]. As shown in Eqs. (11) and (12), the better individuals are quite possible to remain their better performance to the next generation.

In addition, the generation mechanisms of the new population obtained by above three behaviors processes may lead to crossing the search boundary. Thus, the boundary checking method is used to control the search boundary.

Step. 1. if $x_{i,j} < Low$, then $x_{i,j} = rand\times(Up\text{-}Low) + Low$.
Step. 2. if $x_{i,j} > Up$, then $x_{i,j} = \mathrm{rand}\times(Up\text{-}Low) + \mathrm{Low}$.

where Up and Low are the upper and lower bounds, respectively. $x_{i,j}$ is the jth dimension of ith solution.

3.4. Framework of VCS

By combining the three simulated behaviors of virus and the host cell, VCS is presented. The standard Virus Colony Search optimization algorithm can be described in Algorithm 1.

At each generation, for each virus, a viruses diffusion vector V_{pop}'$_i$ is generated by the Gaussian random walk method (i. e. Eq. (1)). Then, check the search bounds of the new population and update the V_{pop} by objective function values. Afterward, the host cell infection vector $Hpop_i^{g+1}$ is created by the individuals of V_{pop} (i. e. Eq. (2)). Then select the λ best individuals for the calculation of the X_{mean}^{g+1} (i. e. Eq. (4)) and other parameters (i. e. Eqs. (5–10)) according to CMA-ES method [30]. Again, check the search bounds of the new population H_{pop} and update the V_{pop}. Finally, immune response process makes a contribution to the evolution of the viruses. If the ranks of the individuals in V_{pop} are worse than the others, they are more likely to evolve themselves. The parameter Pr is used to calculate the performance ranks

of the individuals (i. e. Eq. (11)). According to Eq. (11), the individuals with worse fitness values have the smaller value of *Pr* and the random value *r* may greater than *Pr* with high probability. In result, they are more likely to evolve by Eq. (12). Instead, the better individuals with greater *Pr* may remain unchanged with high probability.

4. Experimental study

In this section, to study our new algorithm, experimental results are carried out based on the constrained and unconstrained benchmark functions. The 10 well-known classic benchmark functions [19] and 30 modern benchmark functions selected from CEC-2014 [33] are considered as the unconstrained benchmark functions. For constrained benchmark functions, three engineering design optimization problems commonly used in literature are employed in this paper. In addition, all the experiments in this section are performed on the computer with 3.20 GHz Intel(R) Core(TM) i5-3470 processor and 4GB of RAM using MATLAB 2013a.

4.1. Compared algorithms for unconstrained benchmarks

To evaluate optimization performance of our VCS for unconstrained benchmark functions, we compare VCS with eight state-of-the-arts nature-inspired optimization algorithms and Appendix A shows their download links.

(1) Artificial bee colony algorithm (ABC) [16] simulates the behaviors of bees for foregoing foods, which is a popular swarm intelligence algorithm. It divides the population into three main groups: employed bees, onlooker bees and scout bees.

Algorithm 1: Pseudo Code of VCS algorithm.

Input: ***N*: the population size**
***MaxFEs*: maximum number of function evaluations**
λ: number of the selected best individuals and $\lambda = \lfloor N/2 \rfloor$

g=0; /*The current generation*/
Generate the V_{pop} by randomly sampling from the search space;
Evaluate the objective function values of each individual in V_{pop}
FEs=*N*;
While *FEs*<=*MaxFEs* **do**
/* Viruses diffusion */
  for *i* from 1 to *N* **do**
    $Vpop'_i = \text{Gaussian}(G^g_{best}, \sigma) + (r_1 \cdot G^g_{best} - r_2 \cdot Vpop_i)$;
  end for
  Check the boundary;
  Evaluate *Vpop'*; *FEs*=*FEs*+*N*;
  Update *Vpop* with *Vpop'*;
  /* Host cells infection */
  for *i* from 1 to *N* **do**
    $Hpop^g_i = X^g_{\text{mean}} + \sigma^g_i \times \text{N}_i(\mathbf{0}, \mathbf{C}^g)$;
  end for
  Check the boundary;
  Evaluate H_{pop} *and update the* V_{pop}; *FEs*=*FEs*+*N*;
  Select the best λ individuals from the V_{pop}
  Calculate the weighted mean of the λ best individuals by $X^{g+1}_{mean} = 1/\lambda \sum_{i=1}^{\lambda} \omega_i \cdot Vpop_i^{\lambda best} \mid \omega_i = \ln(\lambda+1)/(\sum_{j=1}^{\lambda}(\ln(\lambda+1) - \ln(j)))$;
  /* Immune response */
  Calculate the *Pr* through $Pr_{rank(i)} = (N - i + 1)/N$;
  for *i* from 1 to *N* **do**
    for *j* from 1 to *d* **do**
      if $r > Pr_i$, **then** $Vpop_{i,j}'' = Vpop_{k,j} - rand \cdot (Vpop_{h,j} - Vpop_{i,j})$;
      else $Vpop_{i,j}'' = Vpop_{i,j}$; **end-if**
    end-for
  end-for
  Evaluate *Vpop''*; *FEs*=*FEs*+*N*;
  Update *Vpop* with *Vpop'*;
  g=g+1;
end while
Output: the individual with the smallest objective function value in the population.

(2) Cuckoo search (CS) is one of the famous nature-inspired algorithms, developed in 2009 by Xin-She Yang and Suash Deb [17]. Based on brood parasitism of some cuckoo species, CS is enhanced by the Levy flights random walk method.
(3) Animal migration optimization (AMO) algorithm [18], proposed in 2014, is a new swarm intelligence algorithm, which is inspired by the animal migration behavior.
(4) Grew wolf optimizer (GWO) which borrows ideas from the leadership hierarchy and hunting mechanism of grey wolves in nature [19].
(5) Symbiosis organisms search (SOS) [35] that simulates the interactive behavior seen among organisms in nature. Just like our VCS, the processes of SOS for finding solutions contain three main phases: mutualism phases, commensalism phase and parasitism phase.
(6) Evolutionary strategies with covariance matrix adaptation (CMA-ES) algorithm [30] is a classical evolutionary algorithm. Meanwhile, it is introduced as an operator in the process of host cell infection in VCS.
(7) Differential evolution based on covariance and bimodal distribution (CoBiDE) algorithm is a novel DE variant [34]. It has two main components: covariance matrix learning and bimodal distribution parameter setting.
(8) The gravitational search algorithm (GSA) [24] is a well-known physics-based algorithm, which performs the optimization by using a collection of masses interacting with each other based on the laws of gravity and motion.

4.2. Experiment results on classic benchmarks

In order to evaluate the performance of VCS for optimizing the classic benchmark functions, a series of 10 classic benchmark functions selected from [19] are used in this experiment. Appendix B shows the basic descriptions of these functions which can be divided into two main parts: unimodal functions (f_{01}–f_{04}) and multimodal functions (f_{05}–f_{10}). Among the multimodal functions, f_{05}–f_{08} are the high-dimensional multimodal functions and f_{09}–f_{10} are the low-dimensional multimodal functions. For each test function, 30 independent runs are performed with random seeds for each problem.

Our VCS has been compared with six NIs in this experiment: CMA-ES, CoBiDE, CS, AMO, ABC and GSA. For all the test functions, maximum numbers of function evolution (*MaxFEs*) are given in Table 1. The setting values of the compared algorithms and VCS are given below.

(1) CMA-ES: $\lambda = \lfloor N/2 \rfloor$, $\sigma = 0.3$ as in [30];
(2) CoBiDE: $pb = 0.4$, $ps = 0.5$ and mutation strategies and crossover strategies as in [34];
(3) CS: $b = 1.50$, $p_0 = 0.25$ as in [17];
(4) AMO: The number of animals in each group was set to 5 [18];
(5) GSA: $G_o = 100$ and $a = 20$ [24];
(6) ABC: *limit*=(*N*•*d*)/2; Size of Employed-bee=Onlooker-bee=(Colony size)/2 as in [36].
(7) VCS: $\lambda = \lfloor N/2 \rfloor$ and $\sigma = 0.3$.

To have a fair comparison, CS is adjusted based on fitness evolution numbers and ABC is modified according to [36]. In addition, in order to compare the above algorithms under the same number of maximum generation (*MaxG*) for each test function, the population size of VCS is set to 50 since it has three phases. The same as VCS, the population size of AMO and CS is 75 because there are two phases in each algorithm. As for the other compared algorithms, the population size is set to 150. As a result, every algorithm has the same function evolution numbers in each generation.

Table 2 reports the obtained mean results by the compared algorithms over the unimodal functions (f_{01}–f_{04}) based on 30 independent runs, where Best means the best solution, Worst stands for the

Table 1
Maximum number of function evolution for each classic benchmark function.

No.	MaxFEs	No.	MaxFEs	No.	MaxFEs	No.	MaxFEs	No.	MaxFEs
f_{01}	75,000	f_{03}	450,000	f_{05}	6,000	f_{07}	6,000	f_{09}	15,000
f_{02}	150,000	f_{04}	150,000	F_{06}	12,000	f_{08}	225,000	f_{10}	15,000

Table 2
Statistical results obtained by CMA-ES, ABC, GSA, CS, CoBiDE, AMO and VCS through 30 independent runs for 4 unimodal functions.

No.	Algorithms	CMA-ES	ABC	GSA	CS	CoBiDE	AMO	VCS
f_{01}	Best	1.5466E−18	2.7483E−03	1.3419E−18	1.3723E+01	3.5804E+00	2.7420E+02	**0**
	Worst	7.2744E−18	3.0461E−02	4.0454E−18	4.9191E+01	9.4953E+00	1.2554E+03	**0**
	Mean	4.1059E−18	1.4857E−02	2.4645E−18	3.0435E+01	6.2816E+00	6.9953E+02	**0**
	SD	1.6378E−18	6.8874E−03	6.2305E−19	7.5333E+00	1.4058E+00	3.0429E+02	**0**
	Rank	3	4	2	6	5	7	**1**
f_{02}	Best	1.3105E−15	2.0094E+01	4.9108E−10	3.1230E+00	4.7077E+00	1.0420E+01	**0**
	Worst	2.4015E−15	3.3388E+01	7.9136E−10	5.5631E+00	7.8040E+00	2.0371E+01	**0**
	Mean	1.8909E−15	2.7076E+01	6.5711E−10	4.3347E+00	5.8303E+00	1.4578E+01	**0**
	SD	2.4986E−16	3.2654E+00	7.3559E−11	5.8002E−01	6.3750E−01	2.5250E+00	**0**
	Rank	2	6	3	4	5	7	**1**
f_{03}	Best	9.1006E−26	4.6302E−03	2.2294E+01	9.1535E+00	4.7018E+00	2.5852E+02	**0**
	Worst	3.4208E+00	1.9977E−01	2.2955E+01	2.1139E+01	8.3360E+00	8.3854E+03	**2.1743E−29**
	Mean	1.3213E−01	5.5673E−02	2.2640E+01	1.7562E+01	6.6913E+00	2.1439E+03	**1.9816E−30**
	SD	6.2471E−01	5.1536E−02	1.1932E−01	2.8057E+00	8.2613E−01	1.6885E+03	**4.5362E−30**
	Rank	3	2	6	5	4	7	**1**
f_{04}	Best	3.4427E−28	4.5529E−09	1.1188E−18	1.3857E−02	4.0085E−04	1.1724E+02	**0**
	Worst	7.9388E−28	6.3186E−08	3.4994E−18	6.1930E−02	1.3594E−03	6.8062E+02	**0**
	Mean	4.9261E−28	1.9228E−08	1.7988E−18	3.1263E−02	7.6752E−04	2.8593E+02	**0**
	SD	1.0442E−28	1.3578E−08	5.1236E−19	1.2024E−02	2.1692E−04	1.1960E+02	**0**
	Rank	2	4	3	6	5	7	**1**
Average rank		2.5000	4.0000	3.5000	5.2500	4.7500	7	**1**
Overall rank		2	5	3	6	4	7	**1**

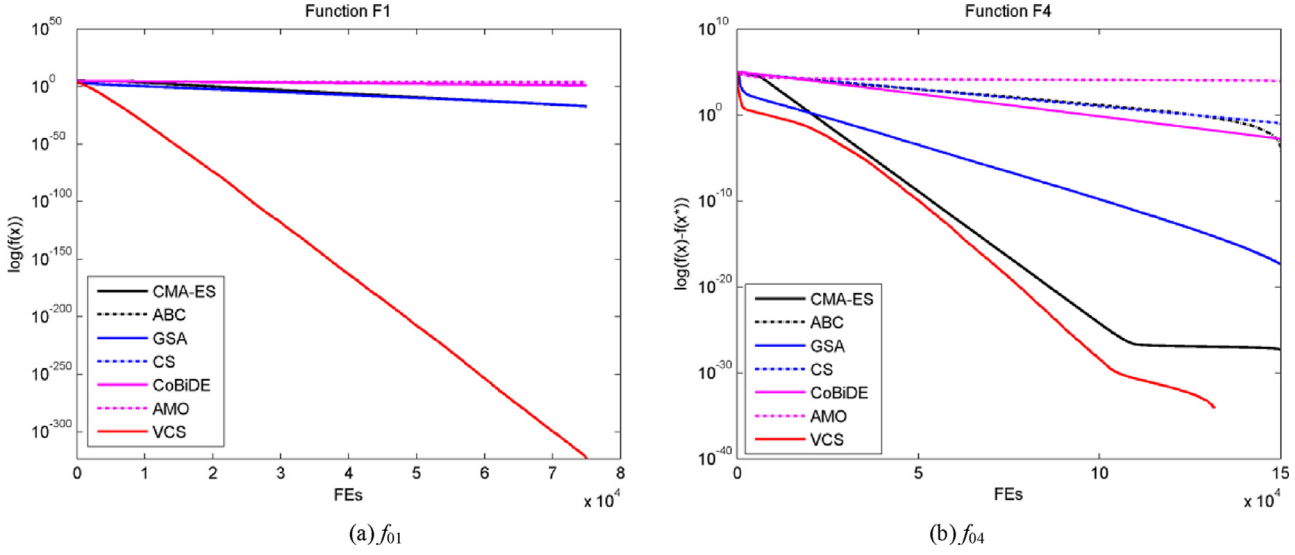


Fig. 2. Evolution of the mean function values derived from the algorithms versus the number of FEs on 2 classic unimodal benchmarks.

worst solution, Mean represents the mean solution and SD means the standard deviation. The best results are set in bold in the coming tables. Meanwhile, the algorithms are sorted based on the mean solution from small to large. In addition, we calculate the mean rank and obtain the overall rank. From Table 2, it can be observed that VCS can find the global optimum for all the unimodal functions the specified *MaxFEs* (see Table 1). Absolutely, VCS ranks the first according to the mean value from the last row of Table 2. As we known, the convergence rate of search algorithm is more important for the unimodal functions than the final results since there are other specific methods aiming at the unimodal functions. Fig. 2 shows the convergence rate of VCS and the other compared algorithms from a graphical point of view for four typical functions. As can be seen from Fig. 2, VCS has the faster convergence rate for the four unimodal functions compared with the other six algorithms.

For the high-dimensional multimodal functions (f_{05}~f_{08}) and the low-dimensional multimodal functions (f_{09}–f_{10}), Table 3 summarizes the average results obtained by the algorithms based on 30 independent runs. The algorithms are ranked based on the mean value of each test function and the average ranks for the six functions are calculated. As these functions have many local minima, they are often used to scrutinize the ability of algorithms to escape from poor local optima and obtain the near-global optima. From Table 3, it can be found that VCS nearly finds all the global optimal values of the high-dimensional functions except for function f_{08}. However, it has the best mean value for f_{08}. As seen from the last row of Table 3, VCS

Table 3
Statistical results obtained by CMA-ES, ABC, GSA, CS, CoBiDE, AMO and VCS through 30 independent runs for 6 multimodal functions.

No.	Algorithms	CMA-ES	ABC	GSA	CS	CoBiDE	AMO	VCS
f_{05}	Best	2.1600E+02	2.1430E+02	1.4224E+02	2.5648E+02	2.8735E+02	2.0867E+02	**0**
	Worst	2.9468E+02	2.6995E+02	2.1671E+02	3.1577E+02	3.4648E+02	2.6796E+02	**0**
	Mean	2.4845E+02	2.4081E+02	1.8010E+02	2.9583E+02	3.1979E+02	2.4232E+02	**0**
	SD	1.8676E+01	1.6164E+01	1.9757E+01	1.5054E+01	1.7891E+01	1.5658E+01	**0**
	Rank	5	3	2	6	7	4	**1**
f_{06}	Best	9.7327E+00	1.4552E+01	3.9747E+00	1.6479E+01	1.5618E+01	1.0899E+01	**8.8818E−16**
	Worst	2.1074E+01	1.7464E+01	5.8987E+00	1.9320E+01	1.8524E+01	1.4297E+01	**8.8818E−16**
	Mean	1.8479E+01	1.6152E+01	4.8523E+00	1.8494E+01	1.7373E+01	1.2654E+01	**8.8818E−16**
	SD	2.5365E+00	5.8207E−01	4.8096E−01	6.4563E−01	5.6357E−01	9.2371E−01	**0**
	Rank	6	4	2	7	5	3	**1**
f_{07}	Best	3.5130E+01	1.5131E+02	3.7085E+02	1.5819E+02	1.6160E+02	3.5736E+01	**0**
	Worst	7.4494E+01	2.8103E+02	5.7022E+02	2.7205E+02	3.0761E+02	1.0138E+02	**0**
	Mean	5.1799E+01	2.2930E+02	4.6054E+02	2.1598E+02	2.3909E+02	6.8466E+01	**0**
	SD	1.0415E+01	3.3075E+01	4.6377E+01	2.9080E+01	3.6583E+01	1.6680E+01	**0**
	Rank	2	5	7	4	6	3	**1**
f_{08}	Best	1.2618E−29	8.6578E−16	6.3881E−21	1.1712E−01	6.2759E−08	9.2321E−01	**1.5705E-32**
	Worst	3.1182E−29	2.2395E−15	4.6089E−02	9.7532E−01	5.6866E−07	1.3212E+01	**1.5705E−32**
	Mean	2.3958E−29	1.2126E−15	1.5363E−03	6.3884E−01	2.1831E−07	6.1927E+00	**1.5705E−32**
	SD	4.8182E−30	2.9913E−16	8.4147E−03	2.1463E−01	1.3920E−07	2.9376E+00	**5.5674E−48**
	Rank	2	3	5	6	4	7	**1**
f_{16}	Best	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00	**−1.0316E+00**
	Worst	−9.7215E-01	−1.0316E+00	−1.0276E+00	−1.0316E+00	−1.0316E+00	−1.0315E+00	**−1.0316E+00**
	Mean	−1.0277E+00	−1.0316E+00	−1.0313E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00	**−1.0316E+00**
	SD	1.0634E−02	6.1573E−12	9.0421E−04	6.4362E−08	7.5083E−11	2.5353E−05	**2.7610E−14**
	Rank	7	2	6	4	3	5	**1**
f_{21}	Best	−1.0153E+01	−1.0153E+01	−1.0153E+01	−1.0095E+01	−1.0153E+01	−1.0107E+01	**−1.0153E+01**
	Worst	−2.6828E+00	−1.0144E+01	−2.6829E+00	−9.2301E+00	−1.0133E+01	−2.5616E+00	**−1.0153E+01**
	Mean	−5.4214E+00	−1.0152E+01	−6.4844E+00	−9.7455E+00	−1.0151E+01	−7.7682E+00	**−1.0153E+01**
	SD	3.6607E+00	2.2751E−03	3.7466E+00	2.5613E−01	3.8507E−03	2.1549E+00	**9.0434E−14**
	Rank	7	2	6	4	3	5	**1**
Average rank		4.8333	3.1667	4.6667	5.1667	4.6667	4.5000	**1.0000**
Overall rank		6	2	3	7	3	5	**1**

ranks first for the classic multimodal functions, while CS performs the worst for these functions. Fig. 3 shows the difference between VCS and other compared algorithms in terms of convergence rate in solving four classic high-dimensional multimodal functions. When examined Fig. 3, VCS algorithm converges more rapidly for the functions compared with the other algorithms.

For the overall comparison on the 10 classic benchmark functions, the pair-wise comparison between the VCS and the other competitors based on the Wilcoxon signed ranks test [37] with a confidence level of 0.95 ($\alpha = 0.05$) is employed. The Wilcoxon signed ranks test, which is a simple, yet safe and robust, nonparametric test for pair-wise statistical comparisons, is usually used to detect the significant differences between two sample means, that is, the performance of two algorithms.

Table 4 summarizes the test results based on the best solution for each function with 30 independent runs, where '+' in the tables means the reference algorithm performed better than the compared algorithms, '–' indicates it performs worse and '=' means the both two algorithms performed equally. R+ represents the sum of ranks for the problems in which VCS outperformed the compared one, and R− means the sum of ranks for the opposite. In addition, the multiple comparisons among all the algorithms are performed by using the Friedman test which is a multiple comparisons test that aims to detect significant differences between the properties of two or more algorithms [37]. It's necessary to emphasize that the Friedman test is accomplished in this paper by using the KEEL software [38]. The average rankings obtained by Friedman test can be used as indicators to illustrate how successful the algorithm is. In other words, the lower the rank, the more successful the algorithm is. Based on the average value and standard deviation for each test function, Table 5 reports the ranks of Mean and SD among seven algorithms obtained by the Friedman's test with a confidence level of 0.95 ($\alpha = 0.05$), in which Iman-Davenport's procedure is used as a post hoc procedure. In Table 5, the result of p-values of Iman-Davenport is obtained by the compared algorithm (i.e. CMA-ES, ABC) vs. the control algorithm (VCS) respectively. When examined the results of last row '+\≈\−' in Table 4, it can be found that VCS has obviously high '+' counts than the other competitors. It means that VCS exhibits the statistically superior performance than the six compared algorithms in the pair-wise Wilcoxon signed ranks test at the 95% significance level. From the results of Table 5, it can be observed that VCS ranks first according to the mean value and standard deviation obtained by the 30 independent runs for each classic benchmark.

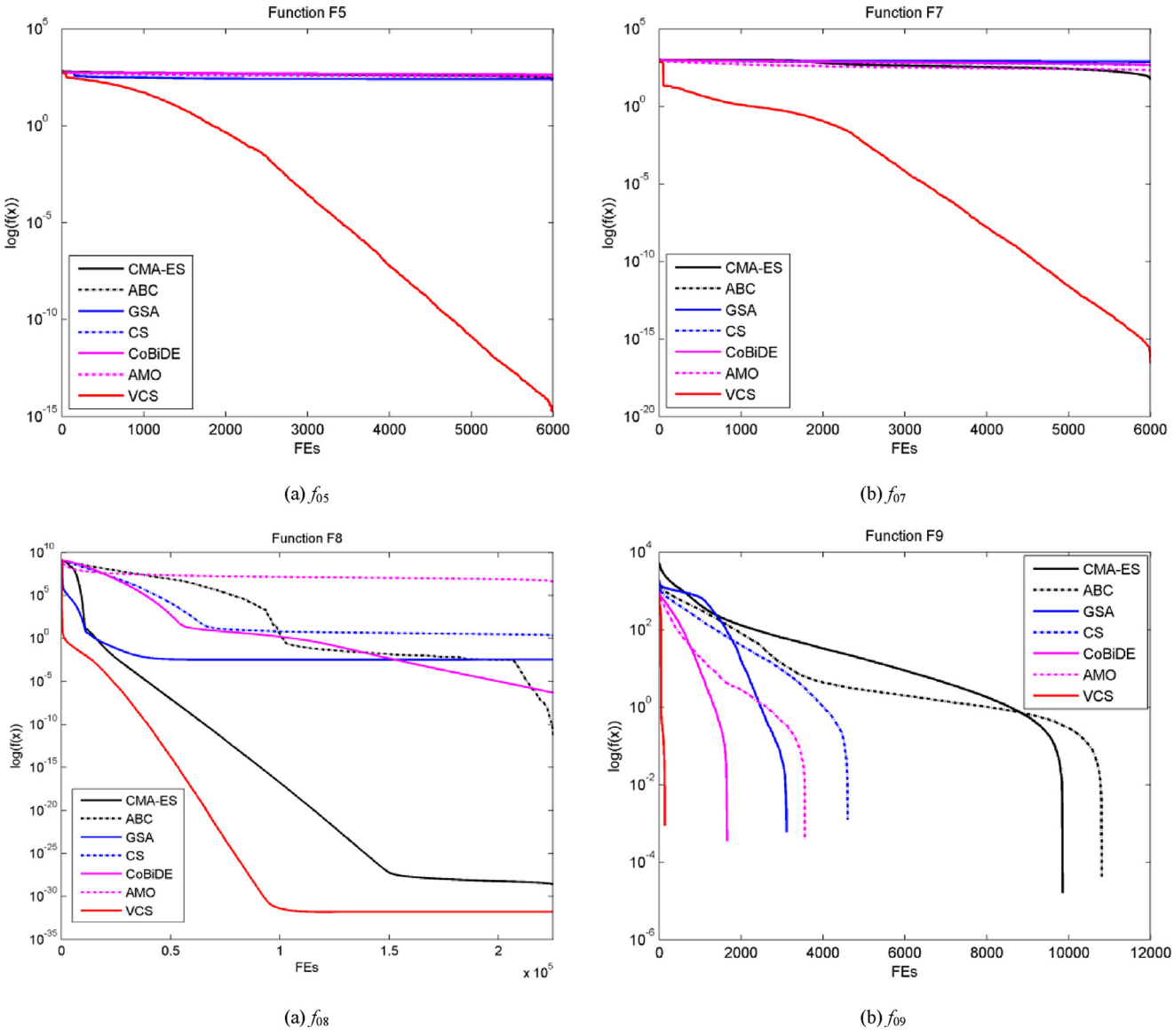
With regard to the time consumption for optimization of these 10 test functions, Table 6 presents the results of mean time obtained by the VCS and other compared algorithms with 30 independent runs. We rank the algorithms from smallest mean time of the test functions to the highest mean time. As can be concluded from Table 6, CMA-ES ranks first among the algorithms. VCS ranks fifth, but it is better than GSA and AMO. As it significantly outperforms the other compared algorithms for the 10 benchmark functions, it is acceptable with a little more time consumption.

4.3. Experiment results on CEC2014 benchmarks

We used 30 IEEE CEC2014 benchmark functions designed for the special issue and competition on single objective and complex real-parameter numerical optimization problems in this experiment. These benchmark functions have several novel features such as novel basic problems, composing test problems by extracting features dimension-wise from several problems, graded level of linkages, rotated trap problems, and so on. Table 7 summarizes several features of the 30 benchmark functions and the detailed descriptions of these functions are provided in [30]. To evaluate the performance of VCS on optimizing the CEC2014 benchmark functions, the results of VCS are compared with six state-of-the-art algorithms: CS [17], GWO [19],

Table 4
The results of Wilcoxon signed ranks test based on the best solution for each benchmarks with 30 independently runs ($\alpha = 0.05$).

No.	CMA-ES vs VCS				ABC vs VCS				GSA vs VCS				CS vs VCS				CoBiDE vs VCS				AMO vs VCS			
	p-value	R+	R−	Win	*p*-value	R+	R−	Win	*p*-value	R+	R−	Win	*p*-value	R+	R−	Win	*p*-value	R+	R−	Win	*p*-value	R+	R−	Win
f_{01}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{02}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{03}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{04}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{05}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{06}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{07}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{08}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{09}	1.7344E−06	465	0	+	1.1259E−05	446	19	+	3.8477E−02	108	28	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{10}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	7.2179E−03	300	78	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
+/≈/−		10/0/0				10/0/0				10/0/0				10/0/0				10/0/0				10/0/0		



(a) f_{05} (b) f_{07}

(a) f_{08} (b) f_{09}

Fig. 3. Evolution of the mean function values derived from the algorithms versus the number of FEs on 4 classic multimodal benchmarks.

Table 5
The results of Friedman test based on the mean value for clasic benchmark functions with 30 independent runs, in which Iman-Davenport's procedure is used as a post hoc procedure (VCS is the control method and $\alpha = 0.05$).

Friedman test	*p*-values of Iman-Davenport	Mean Rank	*p*-values of Iman-Davenport	SD Rank
CMA-ES	0.007006	3.5	0.007118	3.95
ABC	0.043545	3.7	0.038434	3.6
GSA	0.026051	3.9	0.001594	4.4
CS	0.000433	5.15	0.000525	4.7
CoBiDE	0.002262	4.7	0.001594	4.4
AMO	0.000238	5.3	0.000011	5.6
VCS	1	**1.75**	1	**1.35**

GSA [24], CMA-ES [30], SOS [35] and CoBiDE [34]. The dimension of the problems is set to 30 and the *MaxFEs* is 300,000. For a fair comparison, the algorithms are performed in the experiment under the condition of same *MaxFEs* and *MaxG*. In result, the population size is set to 37 for SOS because it has four phases, and it is set to 50 for VCS. For CS, the population size is set to 75. For the other algorithms, the population size is set to 150. The parameters settings for CMA-ES, GSA, CS, CoBiDE and VCS are the same as previous experiment. There are no specific algorithm parameters in SOS [35] and we set a=2−1×(2/*MaxG*) of GWO as in [19]. In addition, the Matlab codes of the compared algorithms can be downloaded from Appendix A.

Tables 8–11 summarize the results of the objective function error values ($f(x)$–$f(x^*)$) over 30 independent runs for each function, where x is the best solution in the population when the algorithm terminates and x^* is the global optimal solution. We rank the algorithms from smallest mean solution to the highest solution. At last, the average ranks and the overall ranks obtained by algorithms are concluded.

Table 6
Mean time consumption in seconds about CMA-ES, ABC, GSA, CS, CoBiDE, AMO and VCS through 30 independent runs on 10 classic functions (NA: not available).

No.	CMA-ES	ABC	GSA	CS	CoBiDE	AMO	VCS
f_{01}	6.0732E−01	7.2153E−01	2.0257E+01	1.4508E+00	1.3013E+00	1.1062E+01	4.3008E+00
f_{02}	1.8813E+00	2.0398E+00	4.0973E+01	3.4780E+00	3.1653E+00	2.2546E+01	8.9607E+00
f_{03}	5.6802E+00	6.0942E+00	2.1886E+02	1.0328E+01	9.6300E+00	6.6935E+01	2.9299E+01
f_{04}	1.3200E+00	1.4905E+00	4.2966E+01	2.9499E+00	2.6355E+00	2.2047E+01	8.9581E+00
f_{05}	6.7926E−02	7.7808E−02	1.7010E+00	1.3423E−01	1.2153E−01	9.3260E−01	3.4826E−01
f_{06}	1.8146E−01	2.0194E−01	3.5487E+00	3.2520E−01	2.9531E−01	1.9092E+00	7.3655E−01
f_{07}	7.7338E−02	8.8203E−02	1.7040E+00	1.4809E−01	1.3456E−01	9.3206E−01	4.2084E−01
f_{08}	6.9515E+00	7.2124E+00	7.2054E+01	9.4189E+00	9.0250E+00	3.8914E+01	1.2221E+01
f_{09}	8.8203E−02	1.3077E−01	2.2372E+00	2.2373E−01	1.9757E−01	3.5757E−01	3.0984E−01
f_{10}	3.0458E−01	3.3745E−01	2.5434E+00	4.3127E−01	4.0314E−01	7.5416E−01	5.5618E−01
Mean time	**1.7159E+00**	1.8394E+00	4.0684E+01	2.8888E+00	2.6909E+00	1.6638E+01	6.6111E+00
Overall rank	**1**	2	7	4	3	6	5

Table 7
The brief descriptions of the CEC2014 benchmark functions.

No.	Types	Name	Optimum
$f_{01(CEC)}$	Unimodal fnctions	Rotated high conditioned elliptic function	100
$f_{02(CEC)}$		Rotated bent cigar function	200
$f_{03(CEC)}$		Rotated discus function	300
$f_{04(CEC)}$	Simple mul-	Shifted and rotated Rosenbrock's function	400
$f_{05(CEC)}$		Shifted and rotated Ackley's function	500
$f_{06(CEC)}$	ti-	Shifted and rotated Weierstrass function	600
$f_{07(CEC)}$	modal	Shifted and rotated Griewank's function	700
$f_{08(CEC)}$	functions	Shifted Rastrigin's function	800
$f_{09(CEC)}$		Six Hump Camel Back	900
$f_{10(CEC)}$		Shifted and rotated Rastrigin's Function	1000
$f_{11(CEC)}$		Shifted and rotated Schwefel's Function	1100
$f_{12(CEC)}$		Shifted and rotated Katsuura Function	1200
$f_{13(CEC)}$		Shifted and rotated HappyCat Function	1300
$f_{14(CEC)}$		Shifted and rotated HGBat Function	1400
$f_{15(CEC)}$		Shifted and rotated Expanded Griewank's plus Rosenbrock's Function	1500
$f_{16(CEC)}$		Shifted and rotated Expanded Scaffer's F6 Function	1600
$f_{17(CEC)}$	Hybrid functions	Hybrid function 1 ($N = 3$)	1700
$f_{18(CEC)}$		Hybrid function 2 ($N = 3$)	1800
$f_{19(CEC)}$		Hybrid function 3 ($N = 4$)	1900
$f_{20(CEC)}$		Hybrid function 4 ($N = 4$)	2000
$f_{21(CEC)}$		Hybrid function 5 ($N = 5$)	2100
$f_{22(CEC)}$		Hybrid function 6 ($N = 5$)	2200
$f_{23(CEC)}$	Composition functions	Composition function 1 ($N = 5$)	2300
$f_{24(CEC)}$		Composition function 2 ($N = 3$)	2400
$f_{25(CEC)}$		Composition function 3 ($N = 3$)	2500
$f_{26(CEC)}$		Composition function 4 ($N = 5$)	2600
$f_{27(CEC)}$		Composition function 5 ($N = 5$)	2700
$f_{28(CEC)}$		Composition function 6 ($N = 5$)	2800
$f_{29(CEC)}$		Composition function 7 ($N = 3$)	2900
$f_{30(CEC)}$		Composition function 8 ($N = 3$)	3000
Search range: [−100, 100]		Dimension: $Dim = 30$	

Table 8
Statistical results obtained by CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS through 30 independent runs on CEC2014 unimodal functions.

No.	Algorithms	CMA-ES	GSA	GWO	CS	CoBiDE	SOS	VCS
$f_{01(CEC)}$	Best	1.8646E+04	1.9183E+07	7.3791E+06	2.2519E+06	5.4866E-01	2.8294E+07	**0**
	Worst	4.1790E+05	3.6521E+07	8.3884E+06	4.7135E+06	1.2756E+01	1.4610E+08	**1.5064E−08**
	Mean	9.3682E+04	2.6618E+07	7.8168E+06	3.2207E+06	3.3429E+00	6.9465E+07	**5.1440E−10**
	SD	8.0449E+04	4.0526E+06	3.0989E+05	6.7652E+05	3.3469E+00	2.9151E+07	**2.7489E−09**
	Rank	3	6	5	4	2	7	**1**
$f_{02(CEC)}$	Best	1.9224E+10	5.3749E+03	3.3780E+08	6.3682E+01	2.7410E−03	1.9174E+09	**0**
	Worst	3.1841E+10	1.1466E+04	3.3987E+08	2.6373E+02	3.2915E−02	4.9364E+09	**0**
	Mean	2.5541E+10	7.8523E+03	3.3861E+08	1.4753E+02	1.0187E−02	3.6133E+09	**0**
	SD	3.8502E+09	1.6593E+03	6.0011E+05	4.6467E+01	5.6804E−03	7.7578E+08	**0**
	Rank	7	4	5	3	2	6	**1**
$f_{03(CEC)}$	Best	6.0588E+03	6.3986E+04	2.1569E+04	9.7031E-01	2.4180E−07	1.0780E+04	**0**
	Worst	3.3837E+04	8.0944E+04	3.8009E+04	2.4785E+00	1.2453E−06	7.6645E+04	**1.1369E−13**
	Mean	1.4476E+04	7.3237E+04	2.7520E+04	1.7565E+00	5.1207E−07	2.3792E+04	**1.3263E−14**
	SD	5.6579E+03	4.0313E+03	7.5329E+03	4.6287E−01	2.3716E−07	1.3519E+04	**2.8649E−14**
	Rank	4	7	6	3	2	5	**1**
Average rank		4.6667	5.6667	5.3333	3.3333	2	6	**1**
Overall rank		4	6	5	3	2	7	**1**

Table 9
Statistical results obtained by CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS through 30 independent runs on CEC2014 simple multimodal functions.

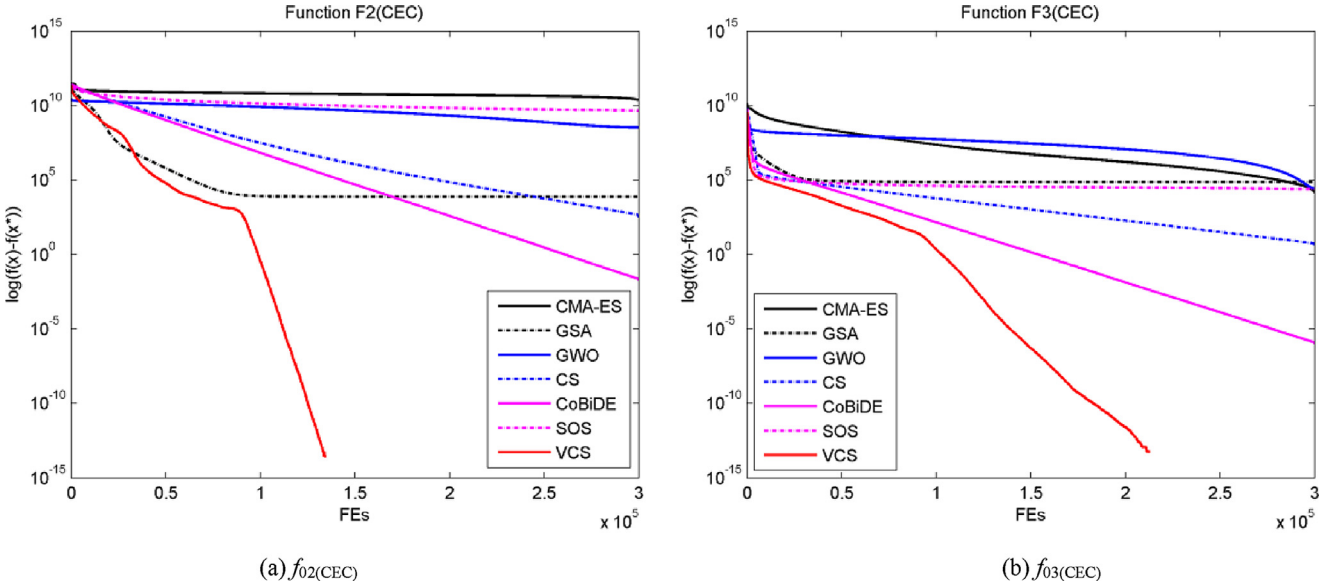
No.	Algorithms	CMA-ES	GSA	GWO	CS	CoBiDE	SOS	VCS
$f_{04(CEC)}$	Best	1.6648E+03	2.2364E+02	1.5638E+02	1.3712E+01	1.5273E+01	2.9519E+02	**0**
	Worst	3.6348E+03	3.6994E+02	1.5803E+02	8.0239E+01	1.7528E+01	5.2217E+02	**6.4965E+01**
	Mean	2.5006E+03	3.1281E+02	1.5693E+02	6.0822E+01	1.6535E+01	4.1011E+02	**2.1655E+00**
	SD	5.2836E+02	2.8508E+01	4.6474E−01	1.9447E+01	6.1149E−01	6.5770E+01	**1.1861E+01**
	Rank	7	5	4	3	2	6	**1**
$f_{05(CEC)}$	Best	2.0000E+01	**1.9998E+01**	2.0793E+01	2.0784E+01	2.0506E+01	2.0389E+01	2.0234E+01
	Worst	2.0000E+01	**1.9999E+01**	2.0962E+01	2.0944E+01	2.0743E+01	2.0646E+01	2.0569E+01
	Mean	2.0000E+01	**1.9999E+01**	2.0803E+01	2.0875E+01	2.0613E+01	2.0559E+01	2.0449E+01
	SD	1.2054E−05	**3.7032E−04**	3.1061E−02	3.7509E−02	4.9072E−02	6.3232E−02	7.5477E−02
	Rank	2	**1**	6	7	5	4	3
$f_{06(CEC)}$	Best	3.3851E+01	1.8305E+01	1.1612E+01	2.2535E+01	2.6397E+01	2.0358E+01	**2.0517E−01**
	Worst	4.1502E+01	2.5393E+01	1.1702E+01	2.7693E+01	3.1539E+01	2.5612E+01	**1.2326E+01**
	Mean	3.9060E+01	2.1678E+01	1.1626E+01	2.5867E+01	2.9433E+01	2.3295E+01	**7.5446E+00**
	SD	1.8249E+00	1.5922E+00	2.2479E−02	1.2244E+00	1.2975E+00	1.5534E+00	**3.0275E+00**
	Rank	7	3	2	5	6	4	**1**
$f_{07(CEC)}$	Best	1.6363E+02	**0**	7.7558E+00	9.3253E−05	3.2746E−08	1.8808E+01	**0**
	Worst	2.7677E+02	**0**	7.8025E+00	2.4310E−03	9.1176E−06	4.5360E+01	**0**
	Mean	2.2944E+02	**0**	7.7765E+00	8.7341E−04	4.6984E−07	3.3330E+01	**0**
	SD	2.3613E+01	**0**	1.4540E-02	6.8749E−04	1.6505E−06	6.9197E+00	**0**
	Rank	7	1	5	4	3	6	**1**
$f_{08(CEC)}$	Best	2.2310E+02	1.2338E+02	8.1032E+01	5.5353E+01	2.0908E+01	7.0217E+01	**1.9899E+00**
	Worst	3.1431E+02	1.7213E+02	8.2620E+01	1.0421E+02	**3.2159E+01**	9.6115E+01	9.7506E+01
	Mean	2.7691E+02	1.4045E+02	8.1521E+01	7.6318E+01	**2.8001E+01**	8.1628E+01	2.9086E+01
	SD	2.0542E+01	9.9125E+00	4.8146E−01	9.6242E+00	**2.7575E+00**	6.8428E+00	2.7970E+01
	Rank	7	6	5	3	1	4	2
$f_{09(CEC)}$	Best	**5.9698E+00**	1.3830E+02	8.6809E+01	1.1063E+02	9.9647E+01	1.2275E+02	6.0692E+01
	Worst	3.5870E+02	1.8307E+02	**8.8598E+01**	1.9086E+02	1.5528E+02	1.9829E+02	2.0695E+02
	Mean	3.1612E+02	1.5996E+02	**8.7582E+01**	1.4277E+02	1.3246E+02	1.6888E+02	1.3293E+02
	SD	6.1211E+01	1.1394E+01	**5.2755E−01**	2.0350E+01	1.2412E+01	1.7462E+01	4.3262E+01
	Rank	7	5	**1**	4	2	6	3
$f_{10(CEC)}$	Best	2.4983E−01	2.8860E+03	2.0527E+03	1.6318E+03	2.1188E+03	1.0922E+03	**1.3091E+00**
	Worst	6.1371E+03	4.0538E+03	2.0623E+03	2.3642E+03	2.9888E+03	1.7484E+03	**2.7687E+03**
	Mean	2.5646E+02	3.3925E+03	2.0557E+03	2.0434E+03	2.5415E+03	1.3840E+03	**1.3845E+02**
	SD	1.1184E+03	3.1447E+02	2.7058E+00	1.8731E+02	1.9107E+02	1.9598E+02	**5.1157E+02**
	Rank	2	7	5	4	6	3	**1**
$f_{11(CEC)}$	Best	**3.1229E−01**	3.4046E+03	2.0037E+03	2.9400E+03	5.0625E+03	3.7636E+03	1.1927E+03
	Worst	**7.5363E+02**	4.9657E+03	2.0216E+03	3.9844E+03	6.0817E+03	5.3848E+03	4.6321E+03
	Mean	**1.5910E+02**	4.1207E+03	2.0107E+03	3.4496E+03	5.6222E+03	4.4798E+03	2.0707E+03
	SD	**2.0332E+02**	4.0622E+02	5.0576E+00	2.2241E+02	2.5377E+02	4.1001E+02	6.4105E+02
	Rank	**1**	5	2	4	7	6	3
$f_{12(CEC)}$	Best	4.6648E−05	**3.8717E−06**	1.5043E−01	5.4709E−01	7.2543E−01	6.5808E−01	2.1283E−02
	Worst	6.3185E+00	**5.7610E−03**	1.5958E−01	1.0959E+00	1.3288E+00	1.2600E+00	8.8849E−01
	Mean	2.1114E−01	**9.0976E−04**	1.5224E−01	8.4278E−01	1.0689E+00	8.2918E−01	4.6944E−01
	SD	1.1535E+00	**1.2838E−03**	1.8504E−03	1.1701E−01	1.3670E−01	1.4366E−01	2.1133E−01
	Rank	3	**1**	2	6	7	5	4
$f_{13(CEC)}$	Best	3.0478E+00	2.4236E−-01	4.1202E−01	2.2991E−01	3.0897E−01	5.3491E−01	**1.7509E−01**
	Worst	4.1519E+00	3.8007E−01	4.3334E−01	4.0208E−-01	4.6378E−01	1.1267E+00	**4.5974E−01**
	Mean	3.6897E+00	3.3139E−01	4.2395E−01	3.1620E−01	4.1123E−01	7.7901E−01	**3.0058E−01**
	SD	2.6688E−01	3.1350E−02	9.1352E−03	3.8937E−02	4.6462E−02	1.4776E−01	**6.7759E−02**
	Rank	7	3	5	2	4	6	**1**
$f_{14(CEC)}$	Best	5.2995E+01	1.8786E−01	6.8862E−-01	1.9160E−01	1.8639E−01	6.0416E−01	**1.8180E−01**
	Worst	8.8239E+01	**2.7160E−01**	7.9910E−-01	2.7321E−01	2.9964E−01	1.9571E+01	3.4302E−01
	Mean	7.0192E+01	**2.3011E−01**	7.2211E−01	2.3719E−01	2.5570E−01	9.7569E+00	2.6678E−01
	SD	9.0881E+00	**2.1927E−02**	4.1523E−02	2.0520E−02	2.7765E−02	3.8714E+00	4.4628E−02
	Rank	7	1	5	2	3	6	4
$f_{15(CEC)}$	Best	2.1795E+00	6.8504E+00	1.8376E+01	8.6516E+00	1.0236E+01	9.5107E+01	**1.8706E+00**
	Worst	1.4113E+05	1.4266E+01	1.8768E+01	1.4740E+01	1.4947E+01	1.0233E+03	**5.8358E+00**
	Mean	1.0166E+04	1.0547E+01	1.8497E+01	1.1216E+01	1.3109E+01	2.4954E+02	**3.5165E+00**
	SD	3.2394E+04	1.6874E+00	1.0467E-01	1.4303E+00	1.1555E+00	2.0512E+02	**1.1467E+00**
	Rank	7	2	5	3	4	6	**1**
$f_{16(CEC)}$	Best	1.1641E+01	1.3309E+01	1.0478E+01	1.1927E+01	1.2079E+01	1.0353E+01	**8.4000E+00**
	Worst	1.4858E+01	1.4035E+01	1.0484E+01	1.2777E+01	1.2898E+01	1.2107E+01	**1.1239E+01**
	Mean	1.4082E+01	1.3697E+01	1.0479E+01	1.2494E+01	1.2603E+01	1.1275E+01	**1.0025E+01**
	SD	7.2818E−01	1.7303E−01	1.6244E−03	1.8355E−01	1.9095E−01	3.7043E−01	**7.3970E−01**
	Rank	7	6	2	4	5	3	**1**
Average rank		5.4615	3.5385	3.7692	3.9231	4.2308	5.0000	**2.0000**
Overall rank		7	2	3	4	5	6	**1**

For the unimodal functions ($f_{01(CEC)}$–$f_{03(CEC)}$), Table 8 reports the results obtained by the algorithms with 30 independent runs. Fig. 4 shows the convergence rate of the algorithms for two typical unimodal functions of CEC2014. The results from Table 8 obviously reflect that VCS performs better than the compared algorithms for the three CEC2014 unimodal functions. Meanwhile, it can find the global optimum for the three benchmark functions. The promising performance of convergence rate and accuracy can also be stressed in Fig. 4.

Table 9 summarizes the results obtained by the algorithms with 30 independent runs for the simple multimodal functions ($f_{04(CEC)}$–$f_{16(CEC)}$). For a graphical point of view, the convergence rates of the

Table 10
Statistical results obtained by CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS through 30 independent runs on CEC2014 hybrid functions.

No.	Algorithms	CMA-ES	GSA	GWO	CS	CoBiDE	SOS	VCS
$f_{17(CEC)}$	Best	1.4094E+03	5.0532E+05	4.9695E+05	6.9128E+03	1.4470E+03	7.8940E+05	**4.9765E+02**
	Worst	2.1128E+04	1.2834E+06	7.5276E+05	2.1280E+04	2.0384E+03	1.7760E+07	**1.6270E+03**
	Mean	5.2311E+03	8.5506E+05	6.4191E+05	1.2739E+04	1.7995E+03	5.6800E+06	**9.9098E+02**
	SD	3.7367E+03	1.7431E+05	8.1755E+04	3.1786E+03	1.3399E+02	3.6949E+06	**2.9295E+02**
	Rank	3	6	5	4	2	7	**1**
$f_{18(CEC)}$	Best	6.1396E+08	2.4138E+02	1.1505E+04	1.0458E+02	2.9247E+01	8.2086E+04	**1.7280E+01**
	Worst	2.1799E+09	7.7730E+02	1.1710E+04	2.0473E+02	**5.6614E+01**	9.8794E+05	1.9363E+04
	Mean	1.5227E+09	3.9029E+02	1.1524E+04	1.4187E+02	**4.5941E+01**	4.9498E+05	1.8850E+03
	SD	3.9300E+08	1.2508E+02	3.9422E+01	2.4367E+01	**6.2499E+00**	2.2637E+05	3.7912E+03
	Rank	7	3	5	2	**1**	6	4
$f_{19(CEC)}$	Best	1.7474E+02	4.0395E+01	1.1911E+01	8.3471E+00	1.1465E+01	1.3799E+01	**2.5066E+00**
	Worst	3.2950E+02	1.6062E+02	1.1937E+01	1.0674E+01	1.5616E+01	1.1974E+02	**6.3258E+01**
	Mean	2.5343E+02	1.3230E+02	1.1914E+01	9.8128E+00	1.3706E+01	3.7980E+01	**9.7539E+00**
	SD	3.9547E+01	2.7529E+01	5.6631E−03	7.1036E−01	8.4427E−01	2.9302E+01	**1.7928E+01**
	Rank	7	4	3	2	5	6	**1**
$f_{20(CEC)}$	Best	5.7168E+02	5.3242E+04	8.5773E+03	8.9498E+01	1.9646E+01	1.3252E+03	**1.1106E+01**
	Worst	1.6925E+04	1.1110E+05	8.7663E+03	1.3874E+02	**3.1446E+01**	5.3553E+04	1.7490E+02
	Mean	4.5100E+03	8.2162E+04	8.6373E+03	1.1183E+02	**2.5259E+01**	1.5919E+04	7.5878E+01
	SD	3.8909E+03	1.3461E+04	5.7698E+01	1.2278E+01	**3.2411E+00**	1.0913E+04	4.6494E+01
	Rank	4	7	5	3	**1**	6	2
$f_{21(CEC)}$	Best	1.6894E+03	1.0812E+05	6.6503E+05	1.3108E+03	4.5545E+02	1.0258E+05	**1.3748E+02**
	Worst	1.2299E+04	2.8053E+05	7.2753E+05	2.0159E+03	**9.3076E+02**	2.8184E+06	2.2953E+04
	Mean	6.8341E+03	1.7890E+05	6.8655E+05	1.6478E+03	**7.0633E+02**	7.8598E+05	1.3223E+03
	SD	2.7677E+03	3.1880E+04	2.1239E+04	1.8697E+02	**1.3084E+02**	6.1237E+05	4.0948E+03
	Rank	4	5	6	3	**1**	7	2
$f_{22(CEC)}$	Best	5.4451E+02	6.1397E+02	3.2162E+02	1.5216E+02	1.1036E+02	2.1809E+02	**2.3168E+01**
	Worst	2.0901E+03	1.3690E+03	3.8299E+02	4.5297E+02	4.1603E+02	8.9681E+02	**5.0746E+02**
	Mean	1.5963E+03	9.3864E+02	3.5852E+02	3.0321E+02	2.5065E+02	5.3606E+02	**2.3188E+02**
	SD	2.8397E+02	1.7978E+02	2.7359E+01	6.9544E+01	8.0520E+01	1.7375E+02	**1.3033E+02**
	Rank	7	6	4	3	2	5	**1**
Average rank		5.3333	5.1667	4.6667	2.8333	2.0000	6.1667	**1.8333**
Overall rank		6	5	4	3	2	7	**1**



(a) $f_{02(CEC)}$ (b) $f_{03(CEC)}$

Fig. 4. Evolution of the mean function error values derived from the algorithms versus the number of FEs on two CEC2014 unimodal benchmarks.

algorithms for several typical multimodal functions are shown in Fig. 5 and the graphical analysis results of the ANOVA tests are shown in Fig. 6.

According to the results from Table 9, it is obvious that VCS presents much better performance than the other algorithms for most simple multimodal functions. More specifically, VCS ranks best based on the mean error value of 30 independent runs for $f_{04(CEC)}$, $f_{06(CEC)}$, $f_{07(CEC)}$, $f_{10(CEC)}$, $f_{13(CEC)}$, $f_{15(CEC)}$ and $f_{16(CEC)}$, and it can find the global optimum for $f_{04(CEC)}$ and $f_{07(CEC)}$. While GSA performs better than the compared algorithms on $f_{05(CEC)}$, $f_{12(CEC)}$ and $f_{14(CEC)}$. CMA-ES ranks first for $f_{11(CEC)}$ and GWO performs the best performance for $f_{09(CEC)}$. For $f_{08(CEC)}$, VCS performs the comparable with CoBiDE. Generally speaking, VCS ranks first for the CEC2014 multimodal functions compared with the other algorithms according to the overall rank in the last line of Table 9. From the graphical results of Fig. 5 and Fig. 6, it can be concluded that VCS has a better performance of convergence rate and global optimization than the compared algorithms for most listed typical functions,

Table 11
Statistical results obtained by CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS through 30 independent runs on CEC2014 composition functions.

No.	Algorithms	CMA-ES	GSA	GWO	CS	CoBiDE	SOS	VCS
$f_{23(CEC)}$	Best	4.4865E+02	2.0000E+02	3.2632E+02	3.1524E+02	3.1401E+02	3.2174E+02	**200**
	Worst	6.8559E+02	2.0000E+02	3.2676E+02	3.1524E+02	3.1401E+02	3.3895E+02	**200**
	Mean	5.5069E+02	2.0000E+02	3.2645E+02	3.1524E+02	3.1401E+02	3.3172E+02	**200**
	SD	5.9239E+01	3.3474E−09	1.1153E−01	2.4473E−07	2.7507E−09	4.0455E+00	**0**
	Rank	7	2	5	4	3	6	**1**
$f_{24(CEC)}$	Best	2.0001E+02	2.0001E+02	2.0000E+02	2.2730E+02	2.0238E+02	2.5830E+02	**200**
	Worst	2.3710E+02	2.0004E+02	2.0000E+02	2.3331E+02	2.4714E+02	2.7624E+02	**200**
	Mean	2.0217E+02	2.0002E+02	2.0000E+02	2.2969E+02	2.3327E+02	2.6580E+02	**200**
	SD	8.3106E+00	5.0426E−03	2.2977E−04	1.3854E+00	1.3127E+01	4.0523E+00	**0**
	Rank	4	3	2	5	6	7	**1**
$f_{25(CEC)}$	Best	2.0645E+02	2.0000E+02	2.0838E+02	2.0620E+02	2.0019E+02	2.1235E+02	**200**
	Worst	2.0986E+02	2.0000E+02	2.0840E+02	2.0928E+02	2.0020E+02	2.2269E+02	**200**
	Mean	2.0829E+02	2.0000E+02	2.0839E+02	2.0766E+02	2.0019E+02	2.1621E+02	**200**
	SD	9.1352E−01	6.2944E−11	5.8837E−03	8.9509E−01	6.4084E−04	2.5961E+00	**0**
	Rank	5	2	6	4	3	7	**1**
$f_{26(CEC)}$	Best	1.0019E+02	1.0999E+02	1.0023E+02	1.0026E+02	1.0035E+02	1.0057E+02	**1.0015E+02**
	Worst	3.8955E+02	2.0002E+02	**1.0032E+02**	1.0043E+02	1.0052E+02	1.0102E+02	200
	Mean	1.2489E+02	1.7574E+02	**1.0025E+02**	1.0034E+02	1.0043E+02	1.0074E+02	1.0358E+02
	SD	5.8099E+01	3.3153E+01	**2.1780E-−02**	4.1301E−02	4.2066E−02	1.2249E−01	1.8210E+01
	Rank	6	7	**1**	2	4	5	3
$f_{27(CEC)}$	Best	4.0067E+02	2.4464E+02	5.5715E+02	4.0686E+02	8.2477E+02	4.2017E+02	**200**
	Worst	1.2850E+03	1.8996E+03	5.6143E+02	4.2596E+02	1.1465E+03	9.4685E+02	**2.0000E+02**
	Mean	1.0698E+03	8.4540E+02	5.5855E+02	4.1320E+02	1.0914E+03	5.0286E+02	**200**
	SD	2.2866E+02	5.6485E+02	1.0359E+00	4.1258E+00	6.6155E+01	1.4766E+02	**8.4444E−14**
	Rank	6	5	4	2	7	3	**1**
$f_{28(CEC)}$	Best	1.9180E+03	2.9771E+02	1.6569E+03	9.1006E+02	3.6550E+02	1.1414E+03	**200**
	Worst	3.8344E+03	1.8267E+03	1.6583E+03	1.0818E+03	3.6859E+02	1.7475E+03	**2.0000E+02**
	Mean	2.7227E+03	7.7441E+02	1.6576E+03	9.8426E+02	3.6747E+02	1.3347E+03	**2.0000E+02**
	SD	6.0676E+02	3.8619E+02	4.5857E−01	4.2457E+01	6.3722E−01	1.3634E+02	**1.5351E−13**
	Rank	7	3	6	4	2	5	**1**
$f_{29(CEC)}$	Best	8.2229E+06	2.0003E+02	9.5785E+03	1.3693E+03	2.1349E+02	1.3661E+04	**200**
	Worst	4.8809E+07	**2.0005E+02**	9.6263E+03	2.8273E+03	2.1757E+02	4.7505E+05	1.7865E+03
	Mean	2.6550E+07	**2.0004E+02**	9.6004E+03	1.7352E+03	2.1548E+02	1.1060E+05	9.7943E+02
	SD	9.0269E+06	**3.5678E−03**	1.3329E+01	2.5011E+02	9.4134E-01	1.1404E+05	3.8188E+02
	Rank	7	**1**	5	4	2	6	3
$f_{30(CEC)}$	Best	4.1640E+05	2.0001E+02	3.0828E+04	1.8935E+03	4.8714E+02	8.6639E+03	**2.0000E+02**
	Worst	9.0590E+05	6.3313E+04	3.2893E+04	4.0682E+03	**8.7817E+02**	8.9358E+04	3.6488E+03
	Mean	6.4777E+05	2.3116E+04	3.1416E+04	2.6249E+03	**7.2650E+02**	3.7318E+04	1.8598E+03
	SD	1.3141E+05	2.4399E+04	6.3652E+02	4.9666E+02	**9.9648E+01**	2.1243E+04	8.9493E+02
	Rank	7	4	5	3	**1**	6	2
Average rank		6.1250	3.3750	4.2500	3.5000	3.5000	5.6250	**1.6250**
Overall rank		7	2	5	3	3	6	**1**

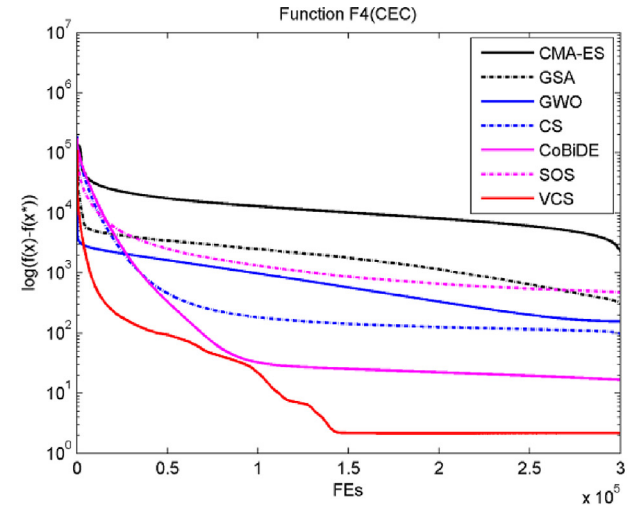
while CMA-ES performs better convergence rate than the VCS on $f_{11(CEC)}$.

For the hybrid functions ($f_{17(CEC)}$–$f_{22(CEC)}$), Table 10 reports the results obtained by the algorithms with 30 independent runs. Figs. 7 and 8 show the convergence rates and ANOVA tests of the algorithms for several hybrid functions. As mentioned above, the CEC2014 hybrid functions are more complex than the above two groups and all the algorithms can hardly find the global optimum for these six hybrid functions. However, according to the results from Table 10, VCS performs more acceptable results than the other compared algorithms. Meanwhile, the best solutions of the VCS for the six functions rank first and it can be concluded that VCS has a promising ability for searching global optimization, but the stability of VCS is relatively poor. Based on the overall rank in the last line of Table 10, VCS performs the best among the compared algorithms on the hybrid functions. According to the graphical results of Figs. 7 and 8, VCS performs better preference for the four typical hybrid functions.

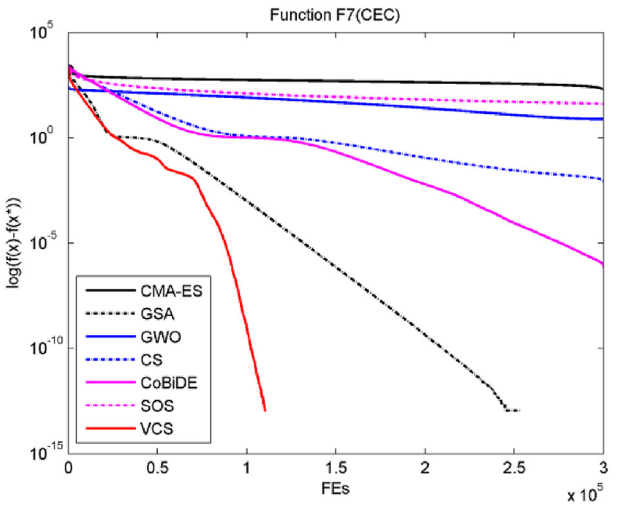
Table 11 summarizes the results obtained by the algorithms with 30 independent runs for the composition functions ($f_{23(CEC)}$–$f_{30(CEC)}$). The graphical analysis results of the ANOVA tests are presented in Fig. 9. According to the results from Table 11, it can be concluded that VCS presents the satisfactory results, especially for $f_{23(CEC)}$~$f_{25(CEC)}$, $f_{27(CEC)}$ and $f_{28(CEC)}$. In addition, it can be observed that VCS can find the minimum best solutions for all composition functions. GWO performs better for optimizing the $f_{26(CEC)}$ and GSA shows better results for $f_{29(CEC)}$. Overall, VCS ranks first for the eight composition functions. From the graphical results of Fig. 9, VCS performs obviously better than the compared algorithms.

For the overall comparison on the CEC2014 benchmark functions, we perform the pair-wise comparison between the VCS and the other competitors based on the Wilcoxon signed ranks test with a confidence level of 0.95 ($\alpha = 0.05$). And the multiple comparisons by Friedman's test with a confidence level of 0.95 ($\alpha = 0.05$) are employed, in which Iman-Davenport's procedure is used as a post hoc procedure for Friedman test and the VCS is used as a control algorithm [37]. Again, the Friedman test is accomplished in this experiment by using the KEEL software [38]. Table 12 summarizes the test results based on the best solution for each function with 30 independent runs and Table 13 reports the ranks of Mean and SD among seven algorithms obtained by the Friedman test. When examined the results of last row '+\≈\−' in Table 12, it can be observed that VCS has obviously high '+' counts than the other competitors. It means that VCS exhibits the statistically superior performance than the six compared algorithms in the pair-wise Wilcoxon signed ranks test at the 95% significance level. From the results of Table 13, it can be founded that VCS ranks first according to the mean value of the 30 independent runs for each CEC2014 benchmark. However, GWO ranks first on SD and VCS ranks third on it.

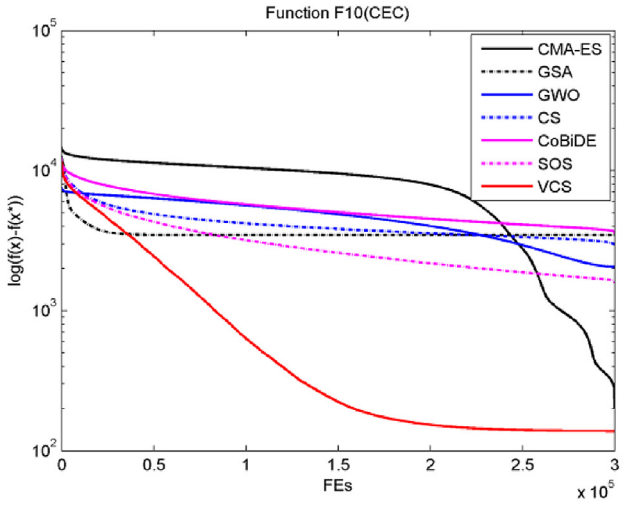
According to above statistical results and analysis, it can be concluded that VCS performs extremely better optimization



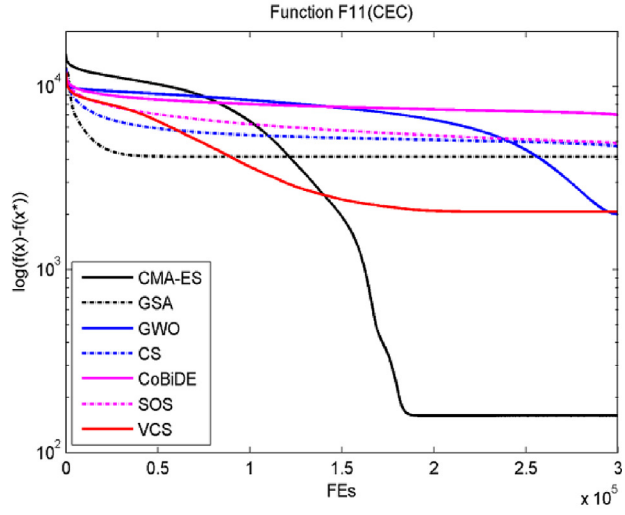
(a) $f_{04(CEC)}$



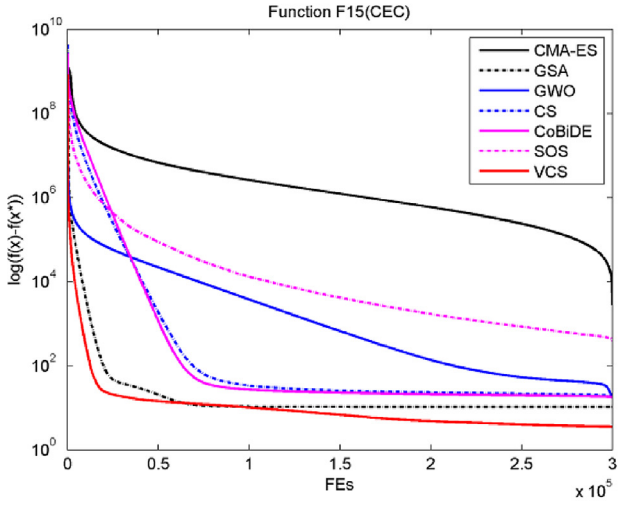
(b) $f_{07(CEC)}$



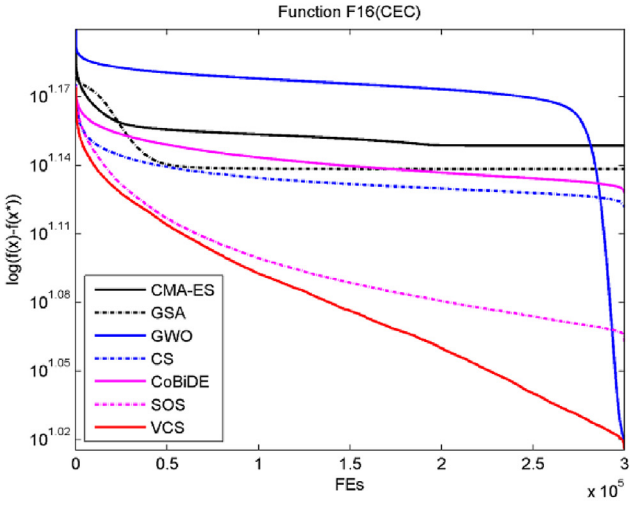
(c) $f_{10(CEC)}$



(d) $f_{11(CEC)}$



(e) $f_{15(CEC)}$



(f) $f_{16(CEC)}$

Fig. 5. Evolution of the mean function error values derived from the algorithms versus the number of FEs on six CEC2014 simple multimodal benchmarks.

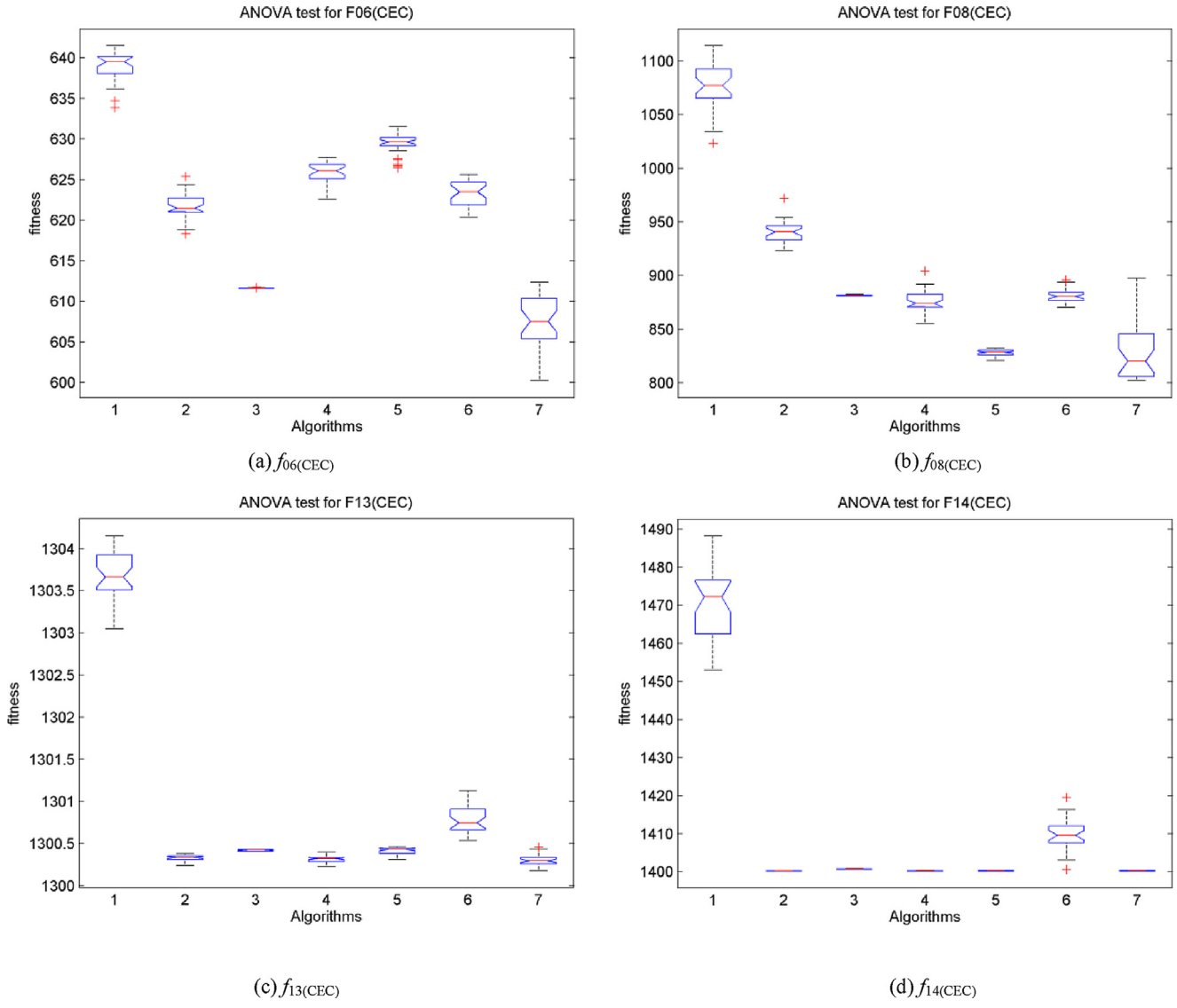


Fig. 6. ANOVA test for all algorithms on four CEC2014 simple multimodal benchmarks. Here 1, 2, 3, 4, 5, 6 and 7 is the algorithm index of CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS, respectively.

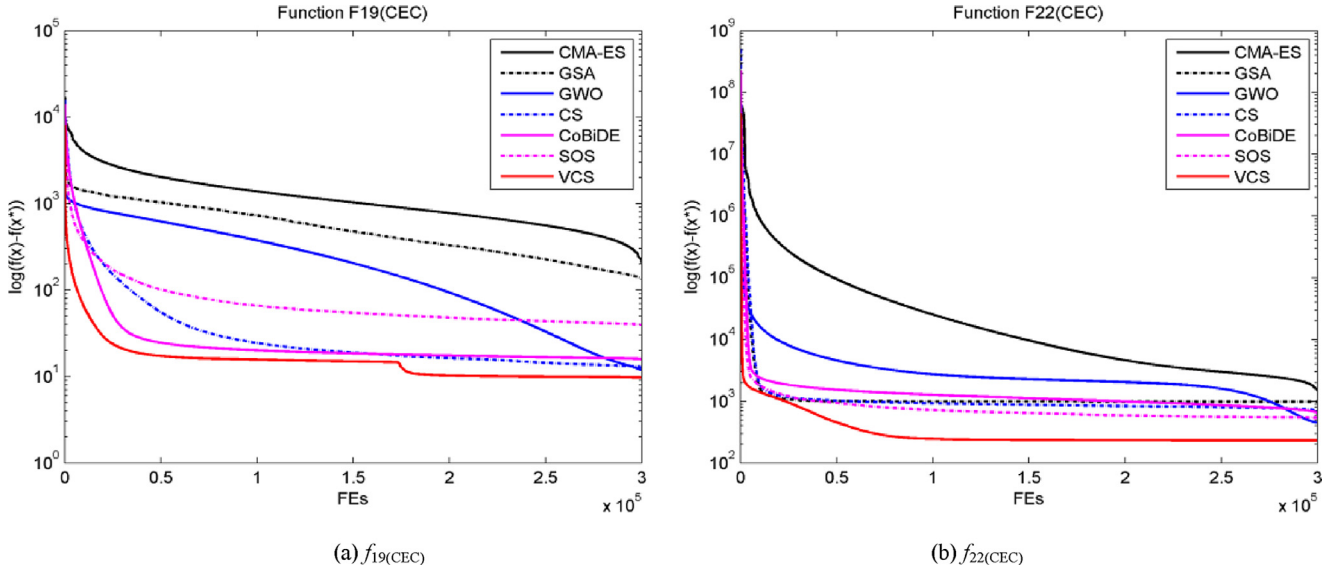


Fig. 7. Convergence rate of the mean function error values derived from the algorithms versus the number of FEs on two CEC2014 hybrid functions benchmarks.

Table 12
The results of Wilcoxon signed ranks test based on the best solution for each benchmark function of CEC 2014 with 30 independently runs ($\alpha = 0.05$).

No.	CMA-ES vs VCS				GSA vs VCS				GWO vs VCS				CS vs VCS				CoBiDE vs VCS				SOS vs VCS			
	p-value	R+	R−	win	*p*-value	R+	R−	win	*p*-value	R+	R−	win	*p*-value	R+	R−	win	*p*-value	R+	R−	win	*p*-value	R+	R-	win
f_{01}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{02}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{03}	1.7344E−06	465	0	+	1.7333E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7311E−06	465	0	+
f_{04}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	3.1123E−05	435	30	+	1.7344E−06	465	0	+
f_{05}	**1.7344E−06**	**0**	**465**	−	**1.7344E−06**	**0**	**465**	−	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.9209E−06	464	1	+	8.4661E−06	449	16	+
f_{06}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	3.1817E−06	459	6	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{07}	1.7344E−06	465	0	+	1.0000E−00	0	0	≈	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{08}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	3.8822E−06	457	8	+	4.2857E−06	456	9	+	5.7165E−01	260	205	≈	3.1817E−06	459	6	+
f_{09}	3.1817E−06	459	6	+	8.7297E−03	360	105	+	**7.5137E−05**	**40**	**425**	**-**	2.5364E−01	288	177	≈	9.7539E−−01	234	231	≈	1.9646E−03	383	82	+
f_{10}	7.0356E−01	214	251	≈	1.7344E−06	465	0	+	1.9209E−06	464	1	+	1.9209E−06	464	1	+	1.9209E−06	464	1	+	6.3391E−06	452	13	+
f_{11}	**1.7344E−06**	**0**	**465**	**-**	2.1266E−06	463	2	+	8.7740E−01	240	225	≈	3.8822E−06	457	8	+	1.7344E−06	465	0	+	1.9209E−06	464	1	+
f_{12}	**3.1123E−05**	**30**	**435**	−	**1.7344E−06**	**0**	**465**	**-**	**5.2165E−06**	**11**	**454**	**-**	2.3534E−06	462	3	+	1.7344E−06	465	0	+	2.8786E−06	460	5	+
f_{13}	1.7344E−06	465	0	+	4.7162E−02	329	136	+	2.8786E−06	460	5	+	1.6503E−01	300	165	≈	6.9838E−06	451	14	+	1.7344E−06	465	0	+
f_{14}	1.7344E−06	465	0	+	**3.8811E−04**	**60**	**405**	**-**	1.7344E−06	465	0	+	**3.3789E−03**	**90**	**375**	**-**	**2.2888E−01**	**174**	**291**	−	1.7344E−06	465	0	+
f_{15}	4.9080E−01	266	199	≈	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{16}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	4.9916E−03	369	96	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{17}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{18}	1.7344E−06	465	0	+	**9.7772E−02**	**152**	**313**	**-**	2.3534E−06	462	3	+	**9.8421E−03**	**107**	**358**	**-**	**2.0515E−04**	**52**	**413**	−	1.7344E−06	465	0	+
f_{19}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	2.7653E−03	378	87	+	2.7653E−03	378	87	+	2.7653E−03	378	87	+	1.2506E−04	419	46	+
f_{20}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.4839E−03	387	18	+	**6.3391E−06**	**13**	**452**	**-**	1.7344E−06	465	0	+
f_{21}	2.8434E−05	436	29	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	3.1123E−05	435	30	+	4.2767E−02	331	134	+	1.7344E−06	465	0	+
f_{22}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	8.9187E−05	426	42	+	7.7309E−03	362	103	+	7.0356E−01	251	214	≈	3.8822E−06	457	8	+
f_{23}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{24}	1.7344E−06	465	0	+	1.7344E−−06	465	0	+	1.5411E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{25}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{26}	9.7110E−05	422	43	+	3.1817E−06	459	6	+	1.7138E−01	166	299	≈	6.3198E−05	427	38	+	3.1123E−05	435	30	+	3.1123E−05	435	30	+
f_{27}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.6933E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{28}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{29}	1.7344E−06	465	0	+	**1.9209E−06**	**1**	**464**	**-**	1.7344E−06	465	0	+	3.1817E−06	459	6	+	**1.9209E−06**	**1**	**464**	**-**	1.7344E−06	465	0	+
f_{30}	1.7344E−06	465	0	+	2.0671E−02	345	120	+	1.7344E−06	465	0	+	2.7653E−03	378	87	+	**5.7517E−06**	**12**	**453**	−	1.7344E−06	465	0	+
+/≈/-		25/2/3				24/1/5				26/2/2				26/2/2				22/3/5				30/0/0		

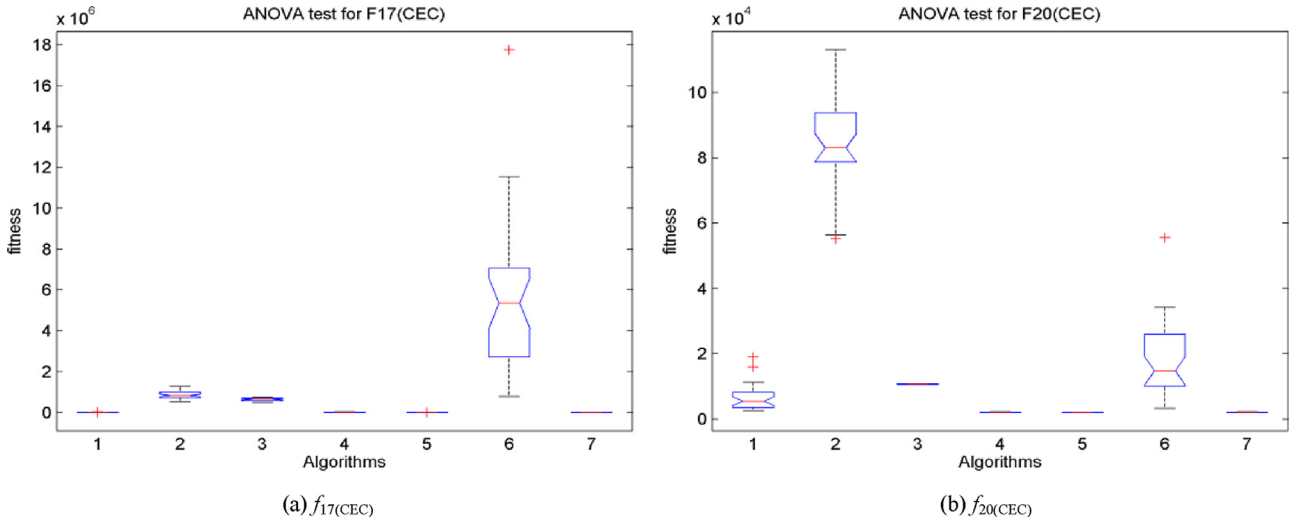


Fig. 8. ANOVA test for all algorithms on two CEC2014 hybrid functions benchmarks. Here 1, 2, 3, 4, 5, 6 and 7 is the algorithm index of CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS, respectively.

Table 13
The results of Friedman test based on the mean value for CEC2014 benchmark functions with 30 independent runs, in which Iman-Davenport's procedure is used as a post hoc procedure (VCS is the control method and $\alpha = 0.05$).

Friedman test	*p*-values of Iman-Davenport	Mean Rank	*p*-values of Iman-Davenport	SD rank
CMA-ES	0.000000	5.5167	0.000188	5.9333
GSA	0.000131	4.1	0.654001	4.1
GWO	0.000062	4.2	0.029162	**2.6333**
CS	0.004124	3.5667	0.198837	3.1333
CoBiDE	0.031444	3.1667	0.055829	2.7833
SOS	0.000000	5.4833	0.002086	5.5667
VCS	1	**1.9667**	1	3.85

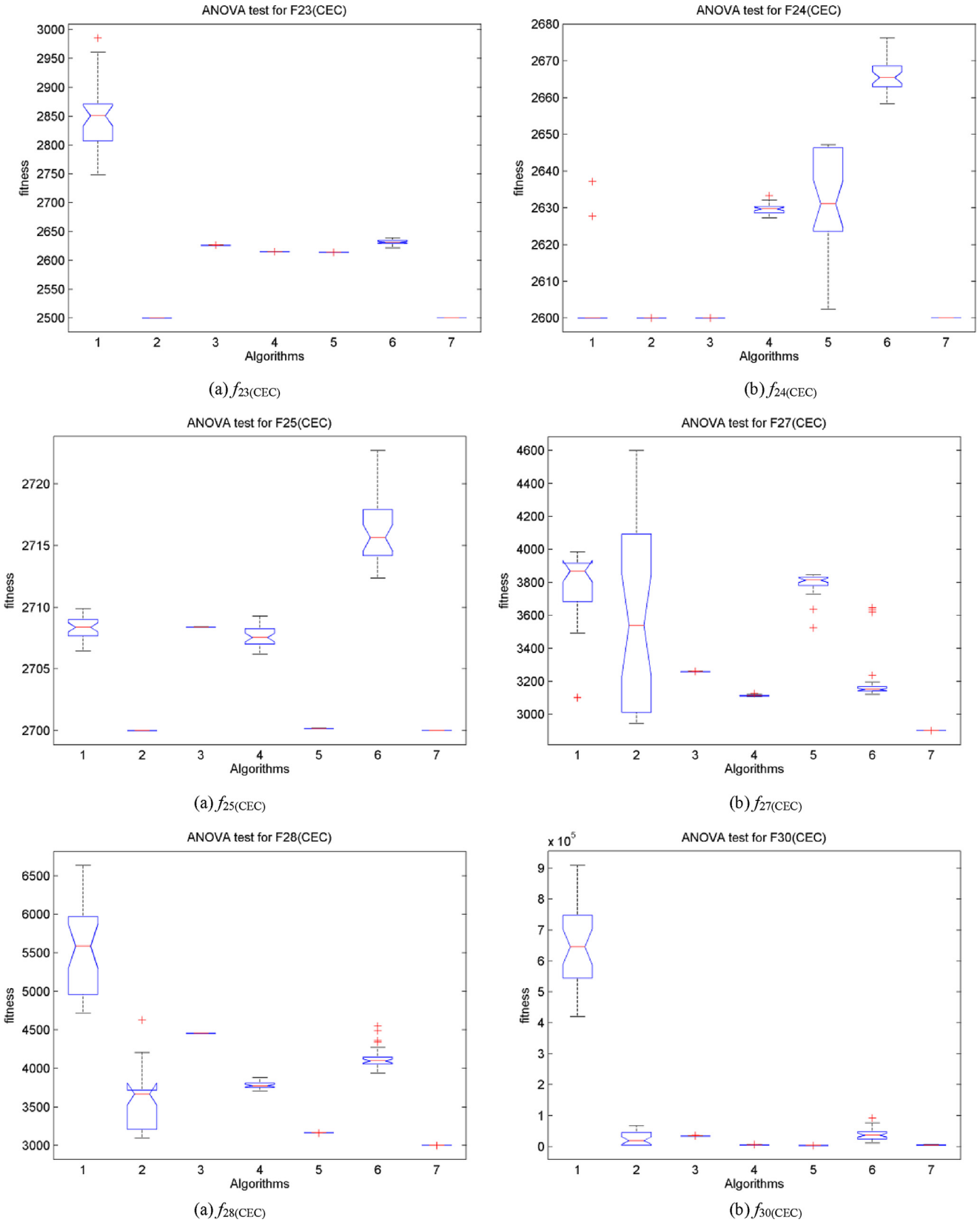
performance on convergence rate and accuracy compared with CMA-ES, GSA, GWO, CS, CoBiDE and SOS for CEC2014 benchmark functions.

4.4. Search behaviors and parameters analysis of VCS

This section experimentally studies the influence of the three search phases, as well as two important parameters for the optimization performance of VCS. To evaluate the search strategies of VCS, we set three conditions: VCS without virus diffusion phase (VCS_1), VCS without host cell infection phase (VCS_2) and VCS without immune response phase (VCS_3). These three conditions and standard VCS are tested on the several classic as well as CEC2014 benchmark functions. The f_{01}, f_{03}, f_{07} and f_{09} are used as the classic benchmark functions, and the $f_{02(CEC)}$, $f_{07(CEC)}$, $f_{19(CEC)}$ and $f_{25(CEC)}$ selected from CEC2014 are used as the modern benchmarks. In addition, the population for these three conditions is set to 75 because each of them has two phases. The other parameters are set according to its previous experiments.

Table 14 represents the average results of 10 runs for each condition and standard VCS. Fig. 10 shows the convergence rate of three conditions and standard VCS. The obtained results from Table 14 show that different condition for different benchmark function presents different performance. It's clear that VCS performs best for the eight test functions. Without the Gaussian random walk in the virus diffusion phase which has a promising performance in convergence rate, VCS1 performs the worst performance on f_{01}, f_{07} and f_{25CEC}. VCS2 exhibits the worst performance on f_{03}, f_{02CEC} and f_{07CEC} because the host cell infection phase can help the individuals to exchange the information so as to avoid local optimal. VCS3 shows the more proper performance compared with VCS1 and VCS2, but worse than VCS. It is because that the immune response phase is mainly used to adjust the individuals slightly so as to get closer to the solutions. Fig. 10 shows that VCS1 converges fast than the other conditions for f_{03} and $f_{02(CEC)}$. While VCS performs best for f_{07} and $f_{19(CEC)}$. Based on the results from Table 14 and Fig. 10, we can observe that each behavior in VCS can significantly affect the quality of final results, and they also show that three behaviors together make a coherent system to solve optimization problems.

There are two important parameters in VCS namely the number of selected best individuals λ and the step size σ. In order to learn the influence of the two parameters on the performance of the algorithm, the following two experiments are performed on the above selected 8 benchmark functions. Tabled 15 and 16 summarize the results obtained by the different two parameter settings based on 30 independent runs. The population size N=50 and the other parameters are set according to its previous experiments. From Table 15, we can observe that there is no significant difference with the different λ for f_{01} and $f_{25(CEC)}$. For f_{03}, f_{09} and $f_{19(CEC)}$, VCS performs the best when $\lambda = \lfloor N/2 \rfloor$. Meanwhile, it can be found that VCS exhibits the worse performance when λ=N or $\lambda = \lfloor N/10 \rfloor$ and it means that the larger or smaller λ value is not helpful to improve the performance of VCS. As we know, the primary design goal for λ is to make full use of best individuals as well as avoid local optimum. It means that larger λ value leads to useless of the best individuals and smaller λ value generates the high probability to run into partial optimization. In addition, because the number of the selected best individuals can directly affect the computational complexity, the more numbers, the larger time consumption. It can be proved from the results of Time in Table 15 that the time consumption becomes larger with the λ increased. Fig. 11 shows the convergence rate on f_{07} and $f_{02(CEC)}$. It can be seen that VCS with $\lambda = \lfloor N/2 \rfloor$ converges faster than the other conditions. Considering the trade-off between time consumption and convergence accuracy, λ is set to $\lambda = \lfloor N/2 \rfloor$ in this paper.

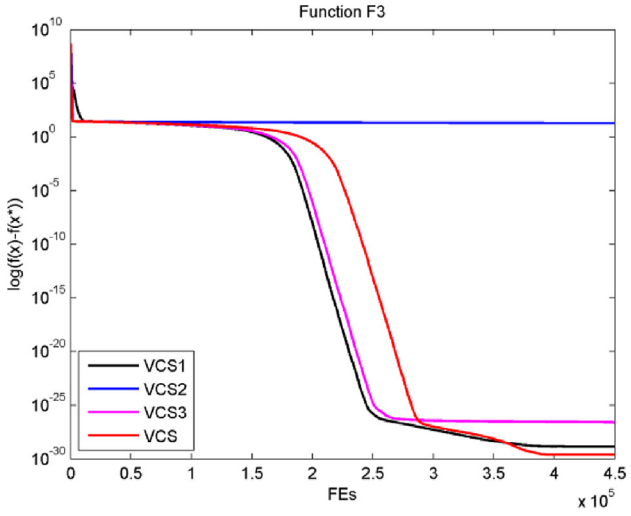


(a) $f_{23(CEC)}$ (b) $f_{24(CEC)}$

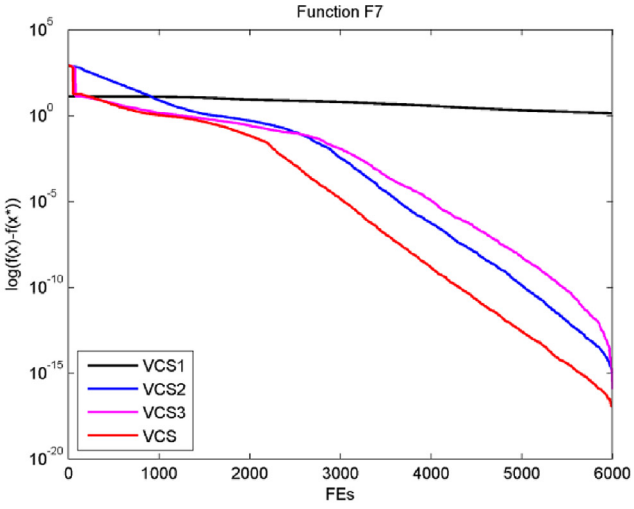
(a) $f_{25(CEC)}$ (b) $f_{27(CEC)}$

(a) $f_{28(CEC)}$ (b) $f_{30(CEC)}$

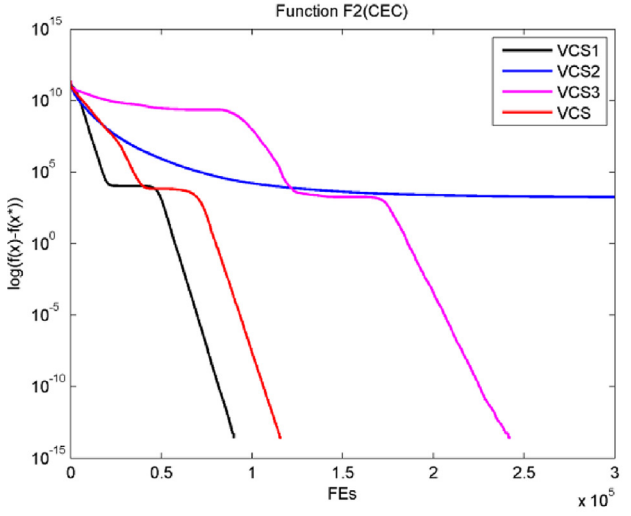
Fig. 9. ANOVA test for all algorithms on six CEC2014 composition functions benchmarks. Here 1, 2, 3, 4, 5, 6 and 7 is the algorithm index of CMA-ES, GSA, GWO, CS, CoBiDE, SOS and VCS, respectively.



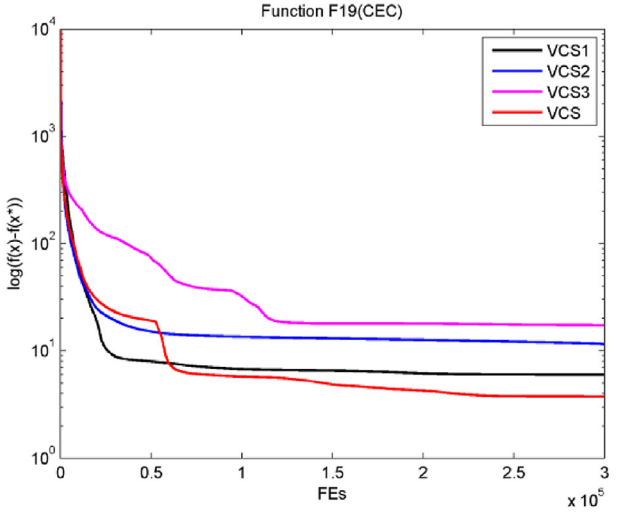
(a) f_{03}



(b) f_{07}

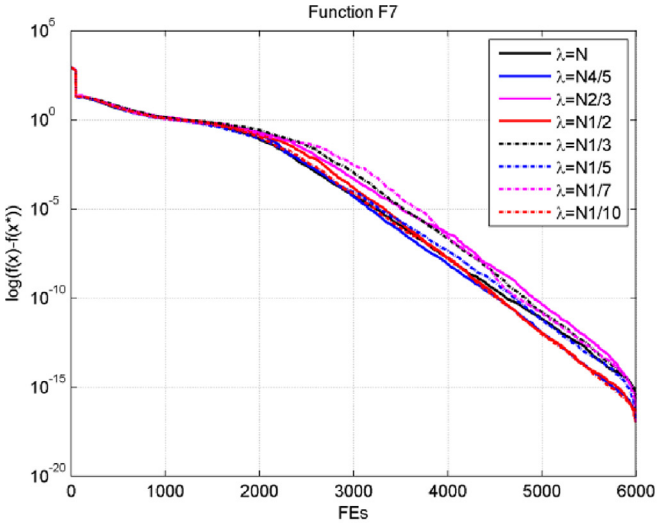


(a) $f_{02(CEC)}$

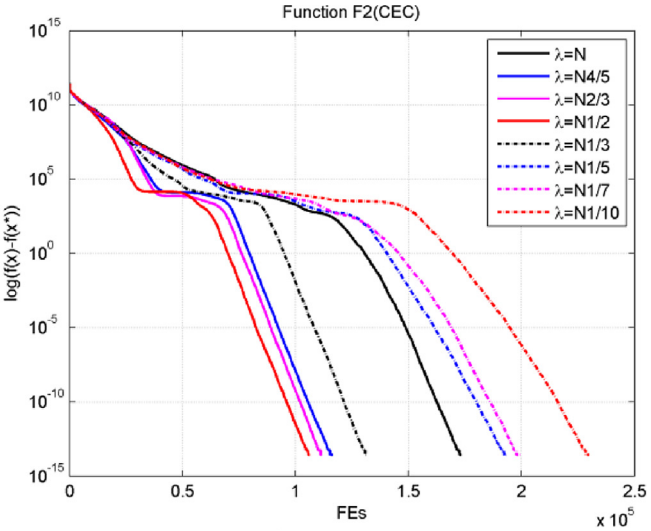


(b) $f_{19(CEC)}$

Fig. 10. Evolution of the mean function error values derived from the different conditions of VCS versus the number of FEs on four benchmark functions.



(a) f_{07}



(b) $f_{02(CEC)}$

Fig. 11. Evolution of the mean function error values derived from the different λ of VCS versus the number of FEs on two benchmark functions.

Table 14
Research on the influence of different search behaviors of VCS for optimizing 8 typical test functions.

No.	VCS_1		VCS_2		VCS_3		VCS	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_{01}	4.9610E−29	4.9966E−29	**0**	**0**	5.3614E−307	0	**0**	**0**
f_{03}	1.4156E−29	1.6084E−29	1.8783E+01	5.3982E−01	1.7598E-27	1.4997E−28	**2.4208E−30**	**3.9914E−30**
f_{07}	1.2179E+00	6.0279E−02	**0**	**0**	**0**	**0**	**0**	**0**
f_{09}	−1.0316E+00	5.5698E−13	−1.0316E+00	1.4673E−14	−1.0316E+00	5.7406E−06	**−1.0316E+00**	**8.3411E−16**
$f_{02(CEC)}$	**0**	**0**	1.7889E+03	3.7294E+03	**0**	**0**	**0**	**0**
$f_{07(CEC)}$	**0**	**0**	1.2506E−13	3.5951E-14	**0**	**0**	**0**	**0**
$f_{19(CEC)}$	6.0075E+00	2.0995E+00	1.1163E+01	1.8247E+01	1.7382E+01	1.9295E+01	**3.7478E+00**	**7.7829E−01**
$f_{25(CEC)}$	2.0000E+02	1.4430E−13	**200**	**0**	**200**	**0**	**200**	**0**

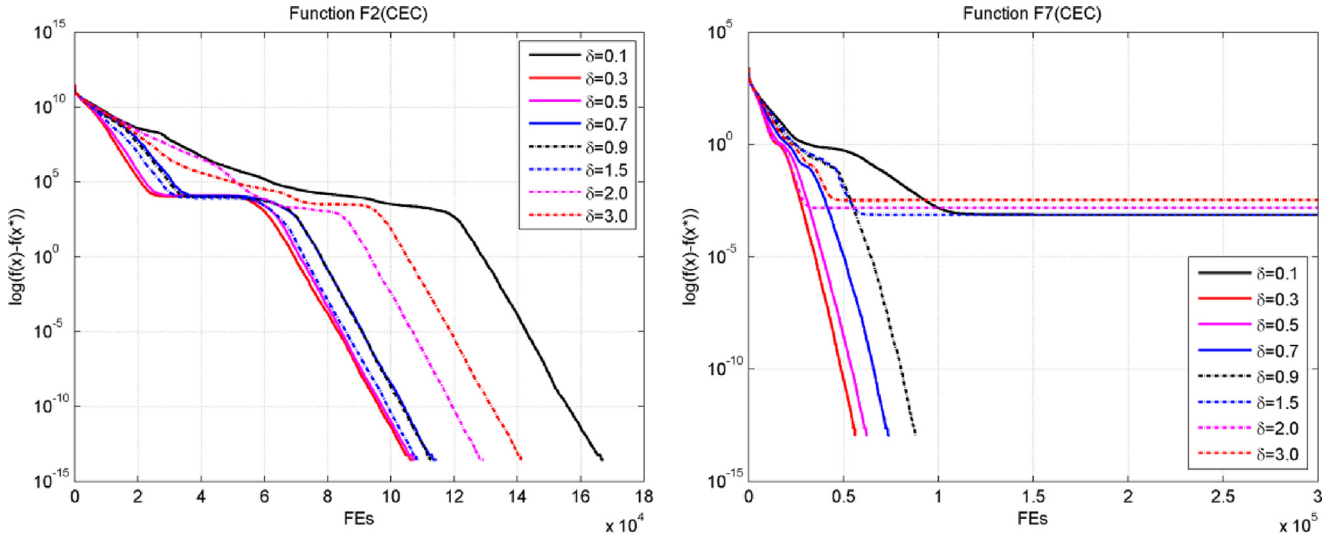
Table 15
Research on the influence of λ on the performance of VCS for optimizing 8 typical test functions ($\sigma = 0.3$).

F	λ	N	$\lfloor N4/5 \rfloor$	$\lfloor N2/3 \rfloor$	$\lfloor N/2 \rfloor$	$\lfloor N/3 \rfloor$	$\lfloor N/5 \rfloor$	$\lfloor N/7 \rfloor$	$\lfloor N/10 \rfloor$
f_{01}	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Time	4.1586E+00	4.1210E+00	4.1071E+00	4.0963E+00	4.0853E+00	4.0366E+00	3.9716E+00	**3.8854E+00**
f_{03}	Mean	3.3151E−29	3.2682E−29	5.3322E−30	**1.1278E−30**	2.2803E−30	9.4170E−30	1.3045E−29	6.1753E−31
	SD	4.1455E−29	4.5193E−29	4.2999E−30	**2.8566E−30**	7.2109E−31	1.6705E−30	8.3041E−30	1.9528E−30
	Time	2.8925E+01	2.8199E+01	2.7755E+01	2.6250E+01	2.5646E+01	2.5434E+01	2.5306E+01	**2.5189E+01**
f_{07}	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	5.5511E−17
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1.7554E−16
	Time	4.6325E−01	4.5161E−01	4.4519E−01	4.0554E−01	4.0308E−01	4.0250E−01	4.0035E−−01	**3.9878E−01**
f_{09}	Mean	−1.0316E+00	-−1.0316E+00	−1.0316E+00	**−1.0316E+00**	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00
	SD	2.8986E−11	4.5602E−12	9.8566E−14	**9.7271E−14**	1.4482E−11	2.4303E−11	3.5243E−11	1.4299E−11
	Time	3.0212E−01	2.9494E−01	2.9324E−01	2.8960E−01	2.8824E−01	2.8822E−01	2.8383E−01	**2.8014E-01**
$f_{02(CEC)}$	Mean	**0**	**0**	**0**	**0**	**0**	**0**	8.5265E−15	8.5265E−15
	SD	**0**	**0**	**0**	**0**	**0**	**0**	1.3729E−14	1.3729E−14
	Time	4.5213E+01	4.2374E+01	4.0127E+01	3.7576E+01	3.2854E+01	2.8635E+01	2.6443E+01	**2.6132E+01**
$f_{07(CEC)}$	Mean	9.8573E−04	9.8647E−04	**0**	**0**	**0**	1.7241E−03	1.9717E−03	2.2188E−03
	SD	3.1171E−03	3.1195E−03	**0**	**0**	**0**	5.4521E−03	4.3158E−03	3.5726E−03
	Time	5.1061E+01	4.9781E+01	4.6254E+01	4.2911E+01	3.6998E+01	2.5037E+01	2.2949E+01	**2.2933E+01**
$f_{19(CEC)}$	Mean	4.1571E+00	4.1672E+00	3.6910E+00	**3.4649E+00**	3.9994E+00	3.9566E+00	4.5904E+00	5.1323E+00
	SD	9.6638E−01	1.3612E+00	1.8496E+00	**6.1255E−-01**	5.0452E-01	1.4250E+00	1.8485E+01	1.2616E+00
	Time	2.8013E+01	2.7614E+01	2.7329E+01	2.6614E+01	2.6058E+01	2.5875E+01	2.5167E+01	**2.4319E+01**
$f_{25(CEC)}$	Mean	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Time	1.8945E+02	1.6134E+02	1.3571E+02	1.1241E+02	9.6867E+01	5.8431E+01	3.4949E+01	**3.0905E+01**

Table 16
Research on the influence of σ on the performance of VCS for optimizing 8 typical test functions ($\lambda = 0.5N$).

Function		$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.5$	$\sigma = 0.7$	$\sigma = 0.9$	$\sigma = 1.5$	$\sigma = 2.0$	$\sigma = 3.0$
f_{01}	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{03}	Mean	**1.8834E-30**	1.9177E-30	2.3035E-30	4.8421E-30	5.6369E-30	6.0841E-30	2.3925E-29	2.6217E-29
	SD	**3.2471E-30**	7.7834E-30	6.1684E-30	9.2544E-30	6.3660E-30	8.2488E-30	3.9619E-30	4.0815E-30
f_{07}	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{09}	Mean	−1.0316E+00	**−1.0316E+00**	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00	−1.0316E+00
	SD	2.2827E−10	**1.8185E−14**	1.9868E−13	4.5417E−12	8.7286E−12	4.6584E−10	3.1597E−10	1.5167E−10
$f_{02(CEC)}$	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
$f_{07(CEC)}$	Mean	7.3960E−02	**0**	**0**	**0**	**0**	**1.3164E−03**	2.4792E−03	3.2045E−03
	SD	2.3388E−03	**0**	**0**	**0**	**0**	**2.8696E−03**	3.1184E−03	4.1976E−03
$f_{19(CEC)}$	Mean	4.3594E+00	**3.8618E+00**	5.3759E+00	1.1725E+01	1.2368E+01	1.6300E+01	1.6784E+01	1.6912E+01
	SD	7.9491E−01	**9.3612E−01**	2.2377E+00	2.8850E+01	1.8856E+01	2.5209E+01	2.5277E+01	1.8870E+01
$f_{25(CEC)}$	Mean	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

According to Eq. (2) in host cell infection behavior, we can observe that the parameter σ is used to control the amplitude for generating the new individuals. A larger σ is effective for global search, while a smaller σ is useful for local search. From the results of Table 16, it can be concluded that VCS exhibits the same convergence rate and accuracy for f_{01}, f_{07}, $f_{02(CEC)}$ and $f_{25(CEC)}$ with different settings of σ. However, VCS performs the best when $\sigma = 0.1$ for optimizing f_{03}, which owes to its promising performance for local search if σ is smaller. Meanwhile, it shows the best performance when $\sigma = 0.3$ for optimizing f_{09}, $f_{07(CEC)}$ and $f_{19(CEC)}$. In addition, VCS performs the worse optimization qualities if the algorithm has a larger σ from Table 16, especially for the complicated test function (i. e. $f_{02(CEC)}$, $f_{07(CEC)}$ and $f_{19(CEC)}$). It is mainly because the larger σ leads to poor quality to get closer to the best solution. Fig. 12 shows the convergence rate with different σ for VCS on the two test functions. Considering the balance between the global search and local search, we set σ=0.3 in VCS for giving full play to optimization performance according to the above results.



(a) $f_{02(CEC)}$ (b) $f_{07(CEC)}$

Fig. 12. Evolution of the mean function error values derived from the different σ of VCS versus the number of FEs on two benchmark functions.

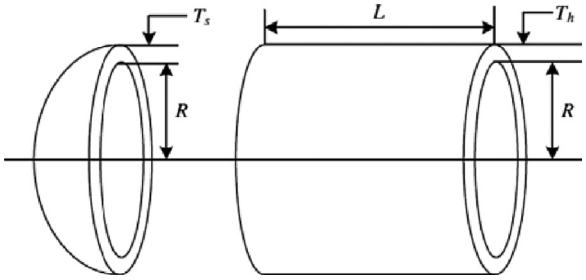


Fig. 13. Pressure vessel design problem.

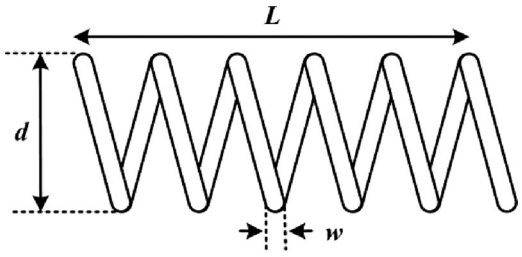


Fig. 14. Tension/compression spring design problem.

4.5. Experiment on the constrained optimization problem

Three well-known constrained engineering design optimization problems (see Appendix C) are used in this section: pressure vessel design problem, tension/compression spring problem and welded beam design problem. To evaluate the performance of VCS for optimizing the three constrained problems, the same constraint handling mechanism used in [39] is employed for VCS, and the results of 30 independent runs on each problem are summarized in Tables 17–22. In addition, the results of the compared algorithms for these three problems are directly taken from the related studies in the literature. For each problem, the population size of VCS is set to 20.

4.5.1. Pressure vessel problem

Pressure vessel design problem is a mixed type of optimization problem. Fig. 13 shows a cylindrical pressure vessel capped at both ends by hemispherical heads. The objective is to minimize the total cost ($f(x)$), including the cost of forming, material and welding. There are four variables in this problem: the thickness Ts (x_1), thickness of the head Th (x_2), the inner radius R (x_3) and the length of the cylindrical section of the vessel L (x_4).

Many researchers have been previously solved this problem as a benchmark structural optimization problem such as Gandomi et al [39], who applied the bat algorithm; Bernardino et al. [40], who employed a hybrid genetic algorithm based on artificial immune system (AIS-GA); Huang et al. [41], who used the co-evolutionary differential evolution (CDE); Wang et al. [42], who employed the differential evolution with level comparison (DELC); Gao and Hailu [43], who used the proposed comprehensive learning particle swarm optimizer algorithm (CLPSO); Hsieh [44], who used the bacterial gene recombination algorithm (BGRA); water cycle algorithm (WCA) designed by Eskandar [45]; mine blast algorithm (MBA) proposed by Sadollah et al. [46] and gaussian quantum-behaved particle swarm optimization (G-QPSO) [47].

Tables 17 and 18 summarize the best results and statistical results obtained by VCS based on 30 independent runs and other different optimizer from the above literature for this problem. From Table 18, the results obtained by VCS with 36,020 function evaluations are better than the compared algorithms except for BGRA. However, VCS offers the competitive best solutions in much less number of function evaluations than offered by the BGRA. Meanwhile, the best cost value of VCS is much lower than that of AIS-GA, CDE, CLPSO, BA, DELC and G-QPSO.

4.5.2. Tension/compression spring design problem

Spring design optimization problem is a well-known engineering design problem to investigate the superiority of an algorithm. The main objective of the problem is to design a spring for a minimum weight by achieving optimum values of the variables as shown in Fig. 14. It contains four nonlinear inequality constraints and three continuous variables: the wire diameter w (x_1), the mean coil diameter d (x_2) and the length (or number of coils) L (x_3).

This problem has been solved previously using BA, AIS-GA, CDE, DELC, BGRA, WCA, MBA, G-QPSO and hybrid evolutionary algorithm and adaptive constraint handling technique (HEAA) [48]. Table 19 shows the comparisons of the best solutions obtained by VCS and other compared algorithms. Table 20 lists the statistical results of

Table 17
Comparing of the best solution obtained by the different algorithms for pressure vessel design problem (NA: Not available).

DV	AIS-GA [40]	CDE [41]	DELC [42]	CLPSO [43]	BGRA [44]	WCA [45]	MBA [46]	G-QPSO [47]	VCS
x_1	0.8125	0.812500	0.8125	0.8125	0.75	0.7781	0.7802	0.8125	0.7781686413715
x_2	0.4375	0.437500	0.4375	0.4375	0.375	0.3846	0.3856	0.4375	0.3846491626265
x_3	42.094967	42.098411	42.0984455	42.0984	38.8601036	40.3196	40.4292	42.0984	40.3196187240987
x_4	176.67972	176.637690	176.636595	176.6366	221.3654714	200.0000	198.4964	176.6372	199.9999999999998
$g_1(x)$	0.000007	−6.677E−07	NA	−8.8000E−07	−8.29972E−13	−2.95E−11	0	−8.79E−07	−3.62510022000e−12
$g_2(x)$	0.035914	−0.035881	NA	−0.0359	−0.00427461	−7.15E−11	0	−3.58E−02	−1.44484424424e−12
$g_3(x)$	0.0625	−3.683016	NA	3.1227	−1.57971E−05	−1.35E−06	−86.3645	−0.2179	**0**
$g_4(x)$	−63.320282	−63.36231	NA	−63.3634	−18.6345	−40.0000	−41.5035	−63.3628	−40.0000000000002
$f(x)$	6060.138	6059.7340	6059.7143	6059.7143	**5850.383061**	5885.3327	5889.3216	6059.7208	5885.332773601229

Table 18
Comparing of statistical results obtained by the different algorithms for pressure vessel design problem (NFEs: number of function evaluations).

Algorithm	Worst	Mean	Best	SD	Runs	NFEs
BA [39]	6,318.95	6,179.13	6,059.71	137.223	50	15,000
AIS-GA [40]	6845.496	6385.942	6060.138	NA	30	80,000
CDE [41]	6371.0455	6085.2303	6059.7340	43.0130	50	204,800
DELC [42]	6,059.7143	6,059.7143	6,059.7143	2.1E-11	30	30,000
CLPSO [43]	NA	6066.0311	6059.7143	12.2718	100	60,000
BGRA [44]	**5850.3944511879**	**5850.383062429**	**5850.3830609**	**7.49E−03**	**30**	**200,000**
WCA [45]	6590.2129	6198.6172	5885.3327	213.0490	25	27,500
MBA [46]	6392.5062	6200.64765	5889.3216	160.34	100	70,650
G-QPSO [47]	7544.4925	6440.3786	6059.7208	448.4711	50	8000
VCS	6854.186813925565	6122.332086401949	5885.339709022323	204.15272	100	7980
	6214.691405673249	5911.620312729348	5885.332773601229	57.358871	100	36,020

Table 19
Comparing of the best solution obtained by the different algorithms for tension/compression spring design problem.

DV	AIS-GA [40]	CDE [41]	DELC [42]	BGRA [44]	WCA [45]	MBA [46]	G-QPSO [47]	HEAA [48]	VCS
x_1	0.0516608	0.051609	0.051689061	0.05167471	0.051680	0.051656	0.051515	0.0516895376	0.051685684299756
x_2	0.3560323	0.354714	0.356717741	0.35637260	0.356522	0.355940	0.352529	0.3567292035	0.356636508703361
x_3	11.329555	11.410831	11.28896566	11.3092294	11.300410	11.344665	11.538862	11.288293703	11.29372966824506
$g_1(x)$	−0.000006	−3.90E−05	NA	−1.1709E−09	−1.65E−13	0	−4.83E−05	NA	−5.29158938888e−11
$g_2(x)$	−0.000013	−1.83E−04	NA	−2.8630E−08	−7.9E−14	0	−3.57E−05	NA	−2.672018162286e−11
$g_3(x)$	−4.052324	−4.048627	NA	−4.0531	−4.053399	−4.052248	−4.0455	NA	−4.053625168175300
$g_4(x)$	−0.728204	−0.729118	NA	−0.727968	−0.727864	−0.728268	−0.73064	NA	−0.727785204664589
$f(x)$	0.0126666	0.0126702	0.012665233	0.012665237	0.012665	0.012665	0.012665	0.012665233	**0.012665222962643**

Table 20
Comparing of statistical results obtained by the different algorithms for tension/compression spring design problem.

Algorithm	Worst	Mean	Best	SD	Runs	NFEs
BA [39]	0.0168954	0.01350052	0.012665233	0.001420272	50	5,000
AIS-GA [40]	0.013880	0.012974	0.012666	NA	30	36,000
CDE [41]	0.012790	0.012703	0.0126702	2.7000E−05	50	204,800
DELC [42]	0.012665575	0.012665267	0.012665233	1.3E−07	30	20,000
BGRA [44]	0.012755029	0.0012682401	0.0126652373	1.9963E−05	30	200,000
WCA [45]	0.012952	0.012746	0.012665	8.06E−05	25	11,750
MBA [46]	0.012900	0.012713	0.012665	6.30E−05	100	7650
G-QPSO [47]	0.017759	0.013524	0.012665	0.001268	50	2000
HEAA [48]	0.012665240	0.012665234	0.012665233	1.4E−09	30	24,000
VCS	0.016324508126357	0.013249353201184	0.012665233297529	6.15806e−04	30	2000
	0.012995994270332	**0.012720016028153**	**0.012665222962643**	**6.52696e−05**	**30**	**11,720**

this problem obtained by VCS with two sets and compares the results with solutions reported by previous researchers. The first set of results corresponds to 11,720 function evaluations and the second set corresponds to 2000 function evaluations using VCS. When examined Table 20, it can be found that the best function value is 0.012665222962643 with 11,720 function evaluations obtained by VCS. In addition, based on 2000 NFEs, VCS can archive the acceptable best result compared with G-QPSO.

4.5.3. Welded beam design problem

The welded beam design optimization problem (see Fig. 15) involves four design variables including the width h (x_1) and length l (x_2) of the welded area, the depth t (x_3) and thickness b (x_4) of the main beam. The main goal is to minimize the overall fabrication cost, under the bending stress σ_d (30,000 psi), appropriate constraints of shear stress τ_d (13,600 psi), maximum end deflection δ_d (0.25 in) and loading condition P (6000 lb).

The optimization methods previously applied to this problem such as CDE, DELC, WCA, MBA, genetic algorithm based co-evolution model (GA3) [49], hybrid particle swarm optimization (HPSO) [50], hybrid nelder-mead simplex search and particle swarm optimization (NM-PSO) [51], hybrid genetic algorithm (HGA) [52], DE [53], and MGA [54].

The comparison of best solution with previous methods is given in Table 21. Table 22 lists the statistical results obtained by VCS and other nine algorithms. From Table 22, it can be observed that the best

Table 21
Comparing of the best solution obtained by the different algorithms for welded beam design problem.

DV	CDE [41]	DELC [42]	WCA [45]	MBA [46]	GA3 [49]	HPSO [50]	NM-PSO [51]	HGA [52]	VCS
$h(x_1)$	0.203137	0.20572964	0.205728	0.205729	0.205986	0.20573	0.20583	0.2057	0.205729639786080
$l(x_2)$	3.542998	3.47048867	3.470522	3.470493	3.471328	3.470489	3.468338	3.4705	3.470488665627995
$t(x_3)$	9.033498	9.03662391	9.036620	9.036626	9.020224	9.036624	9.036624	9.0366	9.036623910357633
$b(x_4)$	0.206179	0.20572964	0.205729	0.205729	0.206480	0.20573	0.20573	0.2057	0.205729639786080
$g_1(x)$	−44.578568	NA	−0.034128	−0.001614	−0.103049	-0.025399	−0.02525	1.988676	−1.81898940354e−12
$g_2(x)$	−44.663534	NA	−3.49E−05	−0.016911	−0.231747	−0.053122	−0.053122	4.481548	0
$g_3(x)$	−0.003042	NA	−1.19E−06	−2.40E−07	−5E−04	0	0.0001	0	−3.33066907387e−16
$g_4(x)$	−3.423726	NA	−3.432980	−3.432982	−3.430044	−3.432981	−3.433169	−3.433213	−3.432983785362249
$g_5(x)$	−0.078137	NA	−0.080728	−0.080729	−0.080986	−0.08073	−0.08083	−0.080700	−0.080729639786080
$g_6(x)$	−0.235557	NA	−0.235540	−0.235540	−0.235514	−0.235540	−0.235540	−0.235538	−0.235540322584754
$g_7(x)$	−38.028268	NA	−0.013503	−0.001464	−58.646888	−0.031555	−0.031555	2.603347	−1.81898940354e-12
$f(x)$	1.733462	1.724852	1.724856	1.724853	1.728226	1.724852	**1.724717**	1.724852	1.724852308597364

Table 22
Comparing of statistical results obtained by the different algorithms for welded beam design problem.

Algorithm	Worst	Mean	Best	SD	Runs	NFEs
CDE [41]	1.824105	1.768158	1.733461	0.022194	50	204,800
DELC [42]	1.724852	1.724852	1.724852	4.1E−13	30	20,000
WCA [45]	1.744697	1.726427	1.724856	4.29E−03	25	46,450
MBA [46]	1.724853	1.724853	1.724853	6.94E−19	100	47,340
GA3 [49]	1.785835	1.771973	1.748309	1.12E−02	11	900,000
HPSO [50]	1.814295	1.749040	1.724852	4.01E−02	30	81,000
NM-PSO [51]	**1.733393**	**1.726373**	**1.724717**	**3.50E−03**	**30**	**80,000**
DE [53]	1.824105	1.768158	1.733461	2.21E−02	100	204,800
MGA [54]	1.9950	1.9190	1.8245	5.37E−02	30	NA
VCS	1.724854932783389	1.724852837940976	1.724852329051929	8.35206e−07	30	12,020
	1.724852308597365	1.724852308597367	1.724852308597364	2.44432e−15	30	36,020

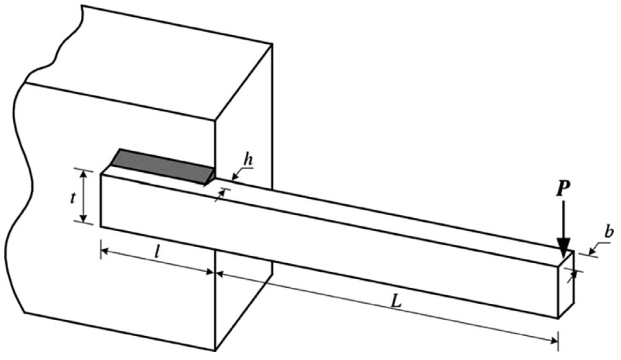


Fig. 15. Welded beam design Problem.

solution is obtained by NM-PSO with an objective function value of $f(x) = 1.724717$ after 80,000 function evaluations. By using the proposed algorithm, the best solution of $f(x) = 1.724852308597364$ is obtained with 36,020 function evaluations, which ranks second from the statistical results. It is worth to mention that the best solution of $f(x)$ by applying the VCS method is 1.724852329051929 after 12,020 function evaluations with SD=8.35206e-07, which is also a satisfactory result with less NFEs compared with CDE, WCA, GA3, HPSO, DE and MGA.

5. Conclusion

Nature-inspired algorithms which mimic the specific phenomena or behaviors of nature have become a hotspot in the optimization computation in recent years. This paper proposes a new optimization algorithm namely virus colony search algorithm, which simulates the virus diffusion and infection strategies for the host cells in the cell environment. Our method employs three main behaviors including: viruses diffusion, host cells infection and immune response. In the first behavior, each virus in the colony performs the diffusion process realized by the Gaussian random walk method to generate the new virus around its current position. The second behavior simulates the infection process for host cell based on CMA-ES method. The last behavior mimics the immune response phenomena of the host cell system by evolving some worse viruses based on the statistical approach. The simulation and the statistical results for the constraint and unconstraint optimization problems, show that the proposed VCS algorithm can equal or outperforms other nature-inspired algorithms used in this paper. In addition, the VCS offers competitive solutions compared with other metaheuristic optimizers based on the reported results in this research. For future work, the performance of time consumption and robustness for optimizing the high-dimensional complex functions needs to be further enhanced. In addition, the binary virus colony search algorithm is expected to be developed for the combinational optimization problems.

Appendix A

CoBiDE: http://ist.csu.edu.cn/YongWang.htm
CMA-ES: http://dces.essex.ac.uk/staff/qzhang/
ABC: http://mf.erciyes.edu.tr/abc/software.htm
CS: http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm
GWO: http://www.mathworks.cn/matlabcentral/fileexchange/44974-grey-wolf-optimizer–gwo-
GSA: http://www.mathworks.com/matlabcentral/fileexchange/27756-gravitational-search-algorithm-gsa
AMO: http://yinmh.nenu.edu.cn/
SOS: http://www.mathworks.com/matlabcentral/fileexchange/47465-sos-m
VCS: It can be obtained from Cor. Author (MuDong Li): Email: modern_lee@163.com

Appendix B

B.1. Four classic unimodal benchmark functions

$f_{01}(x) = \sum_{i=1}^{D} x_i^2$, Dim = 30, Range = [−100, 100], Optimum = 0 (Sphere)

$f_{02}(x) = \max_i\{|x_i|, 1 \le i \le D\}$, Dim = 30, Range = [−100, 100], Optimum = 0 (Schwefel 2.21)

$f_{03}(x) = \sum_{i=1}^{D} 100(x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2$, Dim = 30, Range = [−30, 30], Optimum = 0 (Rosenbrock)

$f_{04}(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$, Dim = 30, Range = [−100, 100], Optimum = 0 (Step)

B.2. Four classic high-dimensional multimodal benchmark functions

$f_{05}(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$, Dim = 30, Range = [−5.12, 5.12], Optimum = 0 (Rastrigin)

$f_{06}(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i))$, Dim = 30, Range = [−32, 32], Optimum = 8.8818e-16 (Ackley)

$f_{07}(x) = \frac{1}{4000}\sum_{i=1}^{D} (x_i^2) - (\prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}})) + 1$, Dim = 30, Range = [−600, 600], Optimum = 0 (Griewank)

$f_{08}(x) = \frac{\pi}{D}\{10\sin^2(\pi y_i) + \sum_{i-1}^{D-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)\} + \sum_{i-1}^{D} u(x_i, 10, 100, 4) \quad y_i = 1 + \frac{x_i+1}{4} \quad u(x_i, a, k, m)$

$$= \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a, \\ k(-x_i - a)^m & x_i < a \end{cases}$$

Dim = 30, Range = [−50, 50], Optimum = 0 (Penalized)

B.3. Two classic low-dimensional multimodal benchmark functions

$f_{09}(x) = 4x_1^2 - 2.1x_1^4 + 1/3x_1^6 + x_1x_2 - 4x_2^2 + x_2^4$, Dim = 2, Range = [−5, 5], Optimum = -1.03163 (Six Hump Camel Back)

$f_{10}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$, Dim = 4, Range = [0, 10], Optimum = -10.1532 (Langermann 5)

Appendix C

C.1. Pressure vessel problem

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to:

$$g_1(X) = -x_1 + 0.0193x_3 \le 0$$
$$g_2(X) = -x_2 + 0.00954x_3 \le 0$$
$$g_3(X) = -\pi x_3^2x_4 - 4/3\pi x_3^2 + 1,296,000 \le 0$$
$$g_4(X) = x_4 - 240 \le 0$$
$$0 \le x_1,\ x_2 \le 100$$
$$10 \le x_3,\ x_4 \le 200$$

C.2. Tension/compression spring design problem

$$f(x) = (x_3 + 2){x_1}^2x_2$$

Subject to

$$g_1(X) = 1 - x_2^3x_3/71785x_1^4 \le 0$$
$$g_2(X) = x_2(4x_2 - x_1)/12566x_1^3(x_2 - x_1) + 1/5108x_1^2 - 1 \le 0$$
$$g_3(X) = 1 - 140.45x_1/x_2^2x_3 \le 0$$
$$g_4(X) = 2(x_1 + x_2)/3 - 1 \le 0$$
$$0.05 \le x_1 \le 2.00$$
$$0.25 \le x_2 \le 1.30$$
$$2.00 \le x_3 \le 15.00$$

C.3. Welded beam design problem

$$f(x) = 1.10471{x_1}^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Subject to:

$$g_1(x) = \tau(x) - \tau_d \le 0$$
$$g_2(x) = \sigma(x) - \sigma_d \le 0$$
$$g_3(x) = x_1 - x_4 \le 0$$
$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0$$
$$g_5(x) = 0.125 - x_1 \le 0$$
$$g_6(x) = \delta(x) - \delta_d \le 0$$
$$g_7(x) = P - P_c(x) \le 0$$
$$0.1 \le x_1, x_4 \le 2$$
$$0.1 \le x_2, x_3 \le 10$$

where,

$$\tau(x) = \sqrt{(\tau'(x))^2 + 2\tau'(x)\tau''(x)x_2/2R + (\tau''(x))^2}$$
$$\tau'(x) = P/\sqrt{2}x_1x_2 \quad \tau''(x) = MR/J$$
$$M = P(L + x_2/2) \quad R = \sqrt{x_2^2/4 + (x_1 + x_3)^2/4}$$
$$J = 2\{\sqrt{2}x_1x_2[x_2^2/12 + (x_1 + x_3)^2/4]\}$$
$$\sigma(x) = 6PL/x_3^2x_4 \quad \delta(x) = 4PL^3/Ex_3^3x_4$$
$$P_c(x) = 4.013E(\sqrt{x_3^2x_4^6/36}/L^2)(1 - \sqrt{E/4G}x_3/2L)$$
$$P = 6000lb \quad L = 14in \quad E = 30 \times 10^6\text{psi} \quad G = 12 \times 10^6\text{psi}$$
$$\tau_d = 13,600\text{psi} \quad \sigma_d = 30,000\text{psi} \quad \delta_d = 0.25in$$

References

[1] Talbi EG. Metaheuristics: From Design to Implementation, vol. 74. John Wiley & Sons; 2009.
[2] Yang X-S. Nature-inspired nature-based Algorithms. Luniver Press; 2008.
[3] Holland J. Adaptation in natural and artificial systems. Cambridge, MA: MIT Press; 1992.
[4] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of sixth international symposium on micro machine and human science (SMMHS-1995); 1995. p. 39–43.
[5] Socha K, Dorigo M. Ant colony optimization for continuous domains. Eur J Oper Res 2008;185(3):1155–73.
[6] Helbig M, Engelbrecht AP. Performance measures for dynamic multi-objective optimization algorithms. Inf Sci 2013;250:61–81.
[7] Upadhyay P, Kar R, Mandal D, Ghoshal SP. Craziness based particle swarm optimization algorithm for IIR. AEU-Int J Electron C 2014;68(5):369–78.
[8] Mylonas SK, Stavrakoudis DG, Theocharis JB. GeneSIS: a GA-based fuzzy segmentation algorithm for remote sensing images. Knowl-Based Syst 2013;54:86–102.
[9] Rao RV, Savasani VJ, Vakharia DP. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Comput Aided Des 2011;43:303–15.
[10] Kim HJ, Kang SH. Communication-aware task scheduling and voltage selection for total energy minimization in a multiprocessor system using Ant Colony Optimization. Inf Sci 2011;181(18):3995–4008.
[11] Chen Y-J, Wong M-L, Li H-B. Applying Ant Colony Optimization to configuring stacking ensembles for data mining. Expert Syst Appl 2014;41(6):2688–702.
[12] Behera S, Sahoo S, Pati BB. A review on optimization algorithms and application to wind energy integration to grid. Renew Sust Energ Rev 2015;48:214–27.
[13] Su Z-K, Wang H-L. A novel robust hybrid gravitational search algorithm for reusable launch vehicle approach and landing trajectory optimization. Neurocomputing 2015;162:116–27.
[14] Gao X-X, Yang H-X, Lin L, Koo P. Wind turbine layout optimization using multi-population genetic algorithm and a case study in Hong Kong offshore. J Wind Eng Ind Aerod 2015;139:89–99.
[15] Price KV, Storn RM, Lampinen JA. Differential evolution: a practical approach to global optimization. Berlin, Germany: Springer-Verlag; 2005.
[16] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71.
[17] Yang X-S, Deb S. Cuckoo search via levy flights. In: Proceedings of World Congress on Nature and Biologically Inspired Computing (NABIC-2009); 2009. p. 210–14.
[18] Li X, Zhang J, Yin M. Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput Appl 2014;24:1867–77.
[19] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61.
[20] Kaveh A, Farhoudi N. A new optimization method: dolphin echolocation. Adv Eng Softw 2013;59:53–70.

[21] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213:267–89.
[22] Kaveh A, Khayatazad M. A novel meta-heuristic method: ray optimization. Comput Struct 2012;112-113:283–94.
[23] Kaveh A, Mahdavi VR. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014;139:18–27.
[24] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179:2232–48.
[25] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Inf Sci 2013;237:82–117.
[26] Ishibuchi H, Yoshida T, Murata T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 2003;7:204–23.
[27] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1:67–82.
[28] Steinhardt E, Israeli C, Lambert RA. Studies on the cultivation of the virus of vaccinia. J Inf Dis 1913;13(2):294–300.
[29] Kaplan DT, Peacock LG. Coarse-grained embeddings of time series: random walks, Gaussian random processes, and deterministic chaos. Physica D 1993;64(4):431–54.
[30] Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMAES). Evol Comput 2003;11(1):1–18.
[31] Vicsek T. Fractal growth phenomena. World Scientific Publishing Company Incorporated; 1992.
[32] Falconer KJ. Random fractals. Math Proc Camb Philos Soc 1986;100(3):559–82.
[33] Liang J., Qu B.-Y., Suganthan P.N. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Technical Report, 2013, p. 1-32.
[34] Wang Y, Li H-X, Huang T-W, Li L. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. Appl Soft Comput 2014;18:232–47.
[35] Cheng M-Y, Prayogo D. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. Comput Struct 2014;139:98–112.
[36] Mernik M, Liu S-H, Karaboga MD, Črepinšek M. On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation. Inf Sci 2015;291:115–27.
[37] Derrac J, Garcia S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 2011;1:3–18.
[38] Alcalá-Fdez J, et al. KEEL: a software tool to assess evolutionary algorithms to data mining problems. Soft Comput 2009;13:307–18 < http://www.keel.es/ >.
[39] Gandomi A, Yang X-S, Alavi A, Talatahari S. Bat algorithm for constrained optimization tasks. Neural Comput Appl 2013;22(6):1239–55.
[40] Bernardino H, Barbosa I, Lemonge A. A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: Proceedings of ieee congress on evolutionary computation (CEC-2007); 2007. p. 646–53.
[41] Huang F-Z, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 2007;186(1):340–56.
[42] Wang L, Li LP. An effective differential evolution with level comparison for constrained engineering design. Struct Multidiscip Opt 2010;41:947–63.
[43] Gao L, Hailu A. Comprehensive learning particle swarm optimizer for constrained mixed-variable optimization problems. Int J Comput Intell Syst 2010;3(6):832–42.
[44] Hsieh TJ. A bacterial gene recombination algorithm for solving constrained optimization problems. Appl Math Comput 2014;231:187–204.
[45] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 2012;110-111:151–66.
[46] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 2013;13(5):2592–612.
[47] Coelho LDS. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst Appl 2010;37:1676–83.
[48] Wang Y, Cai Z, Zhou Y, Fan Z. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint handling technique. Struct Multidisc Optim 2009;37:395–413.
[49] Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113–27.
[50] He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 2007;186:1407–22.
[51] Zahara E, Kao YT. Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Syst Appl 2009;36:3880–6.
[52] Yuan Q, Qian F. A hybrid genetic algorithm for twice continuously differentiable NLP problems. Comput Chem Eng 2010;34:36–41.
[53] Lampinen J. A constraint handling approach for the differential evolution algorithm. IEEE Trans Evol Comput 2002:1468–73.
[54] Coello CAC. Constraint-handling using an evolutionary multiobjective optimization technique. Civ Eng Environ Syst 2000;17:319–46.