algorithms

Article

Modified Cuckoo Search Algorithm with Variational Parameters and Logistic Map

Liping Liu, Xiaobo Liu *, Ning Wang and Peijun Zou

School of Software, Central South University, Changsha 410075, China; lpliu@csu.edu.cn (L.L.); ninganing@csu.edu.cn (N.W.); a318312@csu.edu.cn (P.Z.)

* Correspondence: xiaoboliu@csu.edu.cn; Tel.: +86-0731-8265-5415

Received: 17 January 2018; Accepted: 14 March 2018; Published: 15 March 2018

Abstract: Cuckoo Search (CS) is a Meta-heuristic method, which exhibits several advantages such as easier to application and fewer tuning parameters. However, it has proven to very easily fall into local optimal solutions and has a slow rate of convergence. Therefore, we propose Modified cuckoo search algorithm with variational parameter and logistic map (VLCS) to ameliorate these defects. To balance the exploitation and exploration of the VLCS algorithm, we not only use the coefficient function to change step size α and probability of detection p_a at next generation, but also use logistic map of each dimension to initialize host nest location and update the location of host nest beyond the boundary. With fifteen benchmark functions, the simulations demonstrate that the VLCS algorithm can over come the disadvantages of the CS algorithm.In addition, the VLCS algorithm is good at dealing with high dimension problems and low dimension problems.

Keywords: cuckoo search; logistic map; variational parameter; coefficient function

1. Introduction

Optimization problems are prevalent in society, such as profit maximization, minimum error, and so on [1,2]. To solve this kind of problems, many Meta-heuristic algorithms have been proposed, such as genetic algorithms (GA) [3–5], tabu search [6–8], simulated annealing [9–11], particle swarm optimization [12–14], ant colony optimization [15–17], etc. Cuckoo search (CS) [18] is inspired by nature, which is a concise method and easy to implement. Cuckoo search is widely used in the real world. For example, Shair et al. [19] developed a new approach which is CS algorithm in cutting stock problem. Medjahed et al. [20] proposed a new framework for band selection problem based on binary cuckoo search. However, CS method is not perfect, the main drawbacks being it easily falls into the local optimal solution and the slow rate of convergence [21]. Li and Yin [22] used self adaptive parameter method to improve CS. Wang et al. [23] presented a novel cuckoo search based on chaos theory and elitism. Huang et al. [24] proposed a Chaos-enhanced cuckoo search that use logistic map to ameliorate CS. Liu and Fu [25] proposed a cuckoo search algorithm based on frog leaping local search and chaos theory. Zheng and Zhou [26] used Gaussian distribution to initiate the CS algorithm, which only considered the initial part was not comprehensive. Li and Cao [27] used a DE algorithm and a CS algorithm to propose a new hybrid optimization algorithm. These algorithms actually have some improvement; however, some of the them make the CS become difficult to implement, while others increase the complexity of the CS. Therefore, many future studies are necessary to develop new effective cuckoo search algorithms for optimization problems [22].

Chiroma et al. [28] found that the population reduction and usage of biased random walk are the most frequently used modifications. To improve the CS algorithm, this paper chooses a different method. This paper proposes modified cuckoo search algorithm with variational parameters and logistic sequences (VLCS). VLCS uses logistic map of each dimension to initialize host nest location and update the location of host nest beyond the boundary, which can guarantee that the location of the nest is only calculated once. VLCS also uses coefficient function to change α and p_a at next generation, which greatly accelerates the convergence rate.

The rest of this paper is organized as follows. Section 2 describes the CS algorithm and analyzes the defects of CS algorithm. Section 3 proposes the corresponding solution for the drawbacks of CS algorithm and the VLCS algorithm. Simulation experiments are presented in Section 4. Finally, conclusions are presented in Section 5.

2. Preliminary

2.1. Cuckoo Search Algorithm

Some cuckoos have an aggressive and complicated reproduction strategy. Yang and Deb imitated the strategy and proposed the CS algorithm. The CS [18] obeys three rules: (1) Each cuckoo lays one egg at a time, and dumps its egg in randomly chosen nest. (2) The best nests with high quality of eggs will carry over to the next generations. (3) The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$. Based on these rules, the pseudo code of CS is shown in Algorithm 1.

Algorithm 1: Cuckoo Search via Lévy Flights

```
Input: Objective function f(x), x = $(x_1, \ldots, x_d)^T$
Output: Postprocess results and visualization
1  Generate initial population of n host nests $\mathbf{x}_i$ ($i$ = 1,2, ... ,$n$);
2  while ($t < MaxGeneration$)or(stop criterion) do
3      Get a cuckoo randomly by Lévy flights and evaluate its quality/fitness $F_i$;
4      Choose a nest among n (say, j) randomly;
5      if $F_i > F_j$ then
6          replace j by the new solution;
7      fraction ($p_a$) of worse nests are abandoned and new ones are built;
8      Keep the best solutions or nests with quality solutions;
9      Rank the solutions and find the current best;
10 final ;
11 return Post-process results and visualization;
```

2.2. The Disadvantages of the Cuckoo Search Algorithm

Cuckoo search algorithm has three major drawbacks.

1. Initialization
   Cuckoo search algorithm uses the random number to initiate these location of nests. Sometimes, the location of these nests will be the same, and sometimes the location of these nests are not properly dispersed in a defined area. Therefore, it causes repeated calculations and the easy chance to fall into local optimal solution [24].
2. Parameters α and p_a
   In most cases, Yang and Deb used $\alpha = O(L/10)$ or $\alpha = O(L/100)$, where L is the characteristic scale of the problem of interest [29]. Yang and Deb also suggested p_a = 0.25 [18]. In other words, α and p_a are fixed number. The properties of the two parameters are the shortcomings of the algorithm, because p_a and α should be changed with the progress of iterator, when CS algorithm search a local optimal solution and the global optimal solution.

3. Boundary issue
   CS algorithm uses Lévy flights and random walk to find nest location [18,30]. The locations of some nests may be out of the boundary; when this happens CS algorithm uses the boundary value to replace these location. The bound dealing method will result in a lot of nests at the same location on the boundary, which is inefficient.

3. Modified Cuckoo Search Algorithm: VLCS

This section puts forward the corresponding solution for the drawbacks of the CS algorithm.

3.1. Nest Location of Each Host Are Initialized by Logistic Map of Each Dimension

In this paper, the nest location of each host is initialized by logistic map of each dimension for four reasons. Firstly, the location x_d is found by the logistic map and will not repeat. This means that the speed of convergence can be accelerated. Secondly, logistic map is simple and easy to implement. Thirdly, logistic map is easy to embed into every part of the CS algorithm. Fourthly, since each dimension requires a chaotic map, using other chaotic maps will increase the complexity of the CS algorithm. Logistic map [31] is defined as:

$$x_{n+1} = \mu x_n(1 - x_n), n = 0, 1, 2, 3, \ldots \tag{1}$$

where $x_n \in (0, 1)$, the control parameter $\mu \in [0, 4]$ and n represents the n-th iteration. Chaos phenomenon occurs when $\mu = 4$ [32]. The pseudo code of initialization is shown in Algorithm 2.

Algorithm 2: Nest location of each host is initialized by logistic map of each dimension

```
Input: d represents the dimension;
       Lb is the lower bound of Objective function f(x), Lb = $(Lb_1, \ldots, Lb_d)^T$;
       Ub is the upper bound of Objective function f(x), Ub = $(Ub_1, \ldots, Ub_d)^T$;
Output: nest
1 Generate global random number r, r = $(r_1, \ldots, r_d)^T$,$r_d \in (0, 1)$;
2 $n$ is equal to the number of nests;
3 for $k = 1 : n$ do
4     r = 4*r.*(1−r);                    % Logistic map creates chaos when $\mu = 4$;
5     nest($k$, :) = Lb + (Ub − Lb).*r;  % Initial population of n host nests;
6 final;
7 return nest;
```

3.2. Step Size and p_a Are Changed by Coefficient Function

This article uses the coefficient function to overcome the second disadvantage in Section 2. The coefficient function [33] is defined as:

$$\varepsilon_\alpha = 10^{(10*\tan(\arctan(0.2)*(2*r_{evolving}-1)))} \tag{2}$$

$$r_{evolving} = cur\_iteration / total\_iteration \tag{3}$$

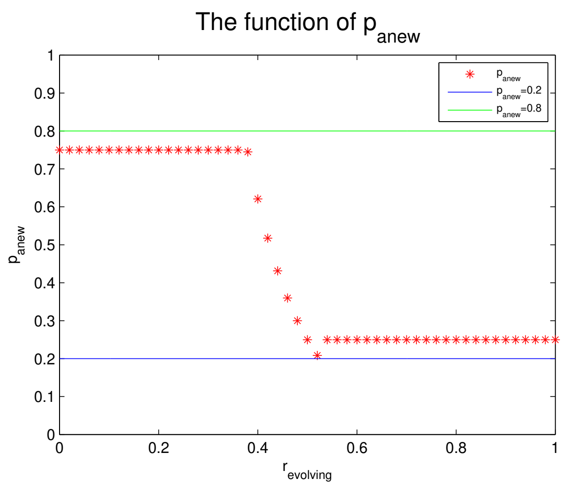
In Equation (3), the cur_iteration means the current number of iteration and the total_iteration means the total number of iteration. The coefficient function in Equation (2) is used to adjust α and p_a (Equation (4)).

$$\alpha_{new} = \alpha/\varepsilon_\alpha$$
$$p_{anew} = \begin{cases} p_{anewLowerValue} & \text{if} \quad p_\alpha/\varepsilon < p_{anewLower} \\ p_{anewUpperValue} & \text{else if} \quad p_\alpha\varepsilon > p_{anewUpper} \\ p_\alpha/\varepsilon & \text{else} \end{cases} \tag{4}$$

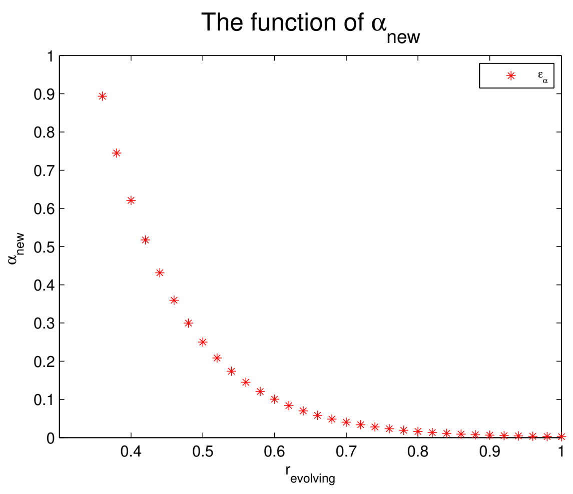
$p_{anewLowerValue} \in (0,1)$, $p_{anewLower} \in (0,1)$, $p_{anewUpperValue} \in (0,1)$ and $p_{anewUpper} \in (0,1)$. To find the optimal parameters, this paper use a template to group parameters and set the change size to 0.5. The template shown in Figure 1d and groups shown in Table 1. In Table 1, each row changes $p_{anewLowerValue}$ and $p_{anewUpperValue}$, and each column changes $p_{anewLower}$ and $p_{anewUpper}$. This paper selects $\alpha = 1$ and $p_a = 0.25$, which are mentioned by Yang and Deb [18] and recent literature [28]. With functions of Table 2, we did 9600 experiments, and the results are shown in Figure 1c. Then, we selects the optimal parameters from those experiments, which are red group 20 in Table 1. These parameters are $p_{anewLowerValue} = 0.25$, $p_{anewLower} = 0.2$, $p_{anewUpperValue} = 0.75$ and $p_{anewUpper} = 0.8$. Then, these values are used to draw Figure 1a. In Figure 1a, at the beginning of the $r_{evolving}$, the $p_{anew} = 0.75$ can guarantee that the VLCS algorithm can jump out of the local optimal solution. At middle of the $r_{evolving}$, the change value of p_{anew} can improve the convergence rate. At the end of the $r_{evolving}$, the $p_{anew} = 0.25$ can guarantee the accuracy of the VLCS algorithm convergence. In Figure 1b, the interval of α_{new} decreases as $r_{revolving}$ increases. The α_{new} replaces the fixed step size of α and the p_{anew} replaces the fixed p_a, which accelerate convergence of cuckoo search algorithm.

Table 1. Parameters groups.

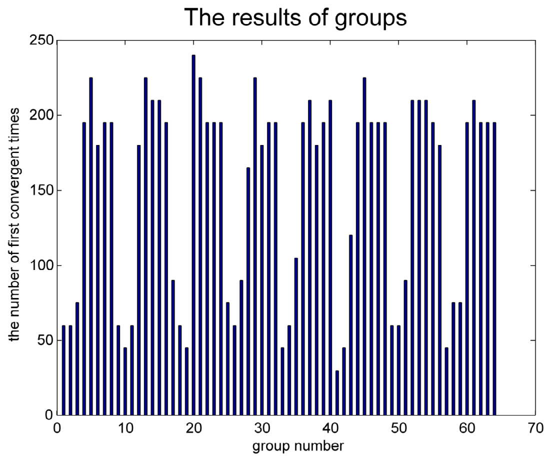
0.10	0.10	0.15	0.10	…	0.25	0.10	…	0.40	0.10	0.45	0.10
0.90	0.90	0.85	0.90		0.75	0.90		0.60	0.90	0.55	0.90
group 1		group 2			group 4			group 7		group 8	
0.10	0.15	0.15	0.15	…	0.25	0.15	…	0.40	0.15	0.45	0.15
0.90	0.85	0.85	0.85		0.75	0.85		0.60	0.85	0.55	0.85
group 9		group 10			group 12			group 15		group 16	
0.10	0.2	0.15	0.20	…	0.25	0.20	…	0.40	0.20	0.45	0.20
0.90	0.80	0.85	0.80		0.75	0.80		0.60	0.80	0.55	0.80
group 17		group 18			group 20			group 23		group 24	
⋮		⋮		⋮	⋮		⋮	⋮		⋮	
0.10	0.45	0.15	0.45	…	0.25	0.45	…	0.40	0.45	0.45	0.45
0.90	0.55	0.85	0.55		0.75	0.55		0.60	0.55	0.55	0.55
group 57		group 58			group 60			group 63		group 64	



(**a**) function p_{anew}



(**b**) function α_{new}



(**c**)The results of groups

$p_{anewLowerValue}$	$p_{anewLower}$
$p_{anewUpperValue}$	$p_{anewUpper}$

(**d**)group template

Figure 1. Parameter selection.

Table 2. benchmark function.

Test Function	**Dimension**	**Range**	**Optimum**
$f_{01} = \sum_{i=1}^{d} x_i^2$	15	$x_i \in [-5.12, 5.12]$	0
$f_{02} = -cos(x)cos(y)exp[-(x-\pi)^2 - (y-\pi)^2]$	2	$x, y \in [-100, 100]$	-1
$f_{03} = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	15	$x \in [-5, 5]$	0
$f_{03} = -\sum_{i=1}^{5} cos[(i+1)x+1] \sum_{i=1}^{5} cos[(i+1)y+1]$	2	$x, y \in [-10, 10]$	-186.7309
$f_{04} = \frac{1}{4000} \sum_{i=1}^{d} x^2 - \prod_{i=0}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1$	15	$x \in [-600, 600]$	0
$f_{05} = \begin{array}{l} -20exp[-20\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}] \\ -exp[\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)] + (20+e) \end{array}$	15	$x \in [-32.768, 32.768]$	0
$f_{06} = \sum_{i=1}^{d-1}[(1-x_i)^2 + 100(x_{i+1} - x_i^2)^2]$	16	$x \in [-10, 10]$	0
$f_{07} = \sum_{i=1}^{d}[-x_i sin(\sqrt{\|x_i\|})]$	10	$x \in [-500, 500]$	-4189.829
$f_{08} = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]$	10	$x \in [-5.12, 5.12]$	0
$f_{09} = -\sum_{i=1}^{d} sin(x_i)[sin(\frac{ix_i^2}{\pi})]^{2m}$	5	$m = 10, x \in [0, \pi]$	-4.6877

Table 2. *Cont.*

Test Function	Dimension	Range	Optimum
$f_{10} = 10 * d + \sum_{i=1}^{d}(x_i^2 - 10cos(2\pi x_i))$	20	$x \in [-10, 10]$	-654.6
$f_{11} = \sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	30	$x \in [-100, 100]$	0
$f_{12} = \sum_{i=1}^{d}(x_i + 0.5)^2$	30	$x \in [-100, 100]$	0
$f_{13} = \sum_{i=1}^{d} ix_i^4 + random[0, 1)$	30	$x \in [-1.28, 1.28]$	0
$f_{14} = \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i) + 10]$	30	$x \in [-5.12, 5.12]$	0
$f_{15} = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})cosx_1 + 10$	2	$x_1, x_2 \in [-5, 15]$	0.398

3.3. Boundary Is Constrained by Logistic Map of Each Dimension

For the boundary issue, the paper also uses the global random number **r**, which is changed by logistic map.The reason for using logistic map is the same as in Section 3.1. Algorithms 2 and 3 use the same **r**. This **r** can guarantee that the location of each nest is calculated only once.That means the **r** reduces repeated calculations and accelerates the convergence process. The pseudo code of boundary processing is shown in Algorithm 3.

Algorithm 3: Boundary is constrained by logistic map of each dimension

```
Input: d represents the dimension;
       Global random number r, r = $(r_1, \ldots, r_d)^T$, $r_d \in (0, 1)$;
       $nest(i, :) = [x_1, x_2, \ldots, x_d]$, $i \in [1, n]$;
Output: nest
outLbMatrix is $1 \times d$ matrix,outLbMatrix= $[OL_1, OL_2, \ldots, OL_d]$,$OL_d = 1$;
outUbMatrix is $1 \times d$ matrix,outUbMatrix= $[OU_1, OU_2, \ldots, OU_d]$,$OU_d = 1$;
for $j = 1 : d$ do
    if the $x_d$ of $nest_i$ out of Lb then
        $OL_j = 1$;
    else
        $OL_j = 0$;
for $j = 1 : d$ do
    if the $x_d$ of $nest_i$ out of Ub then
        $OU_j = 1$;
    else
        $OU_j = 0$;
for $i = 1 : n$ do
    for $j = 1 : d$ do
        if $(outLbMatrix(j) > 0)$ then
            $r(j) = 4 * r(j) * (1 - r(j))$;
            $nest(i, j) = Lb(j) + (Ub(j) - Lb(j)) * r(j)$;
        if $(outUbMatrix(j) > 0)$ then
            $r(j) = 4 * r(j) * (1 - r(j))$;
            $nest(i, j) = Lb(j) + (Ub(j) - Lb(j)) * r(j)$;
final ;
return nest;
```

3.4. Proposed VLCS algorithm

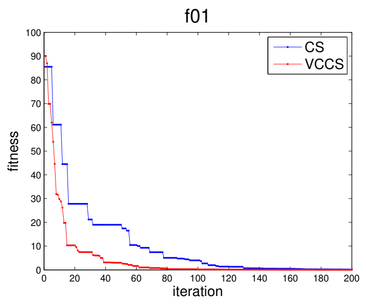
This paper proposes VLCS algorithm to balance the exploitation and exploration. The VLCS algorithm uses logistic map of each dimension to handle global random $\boldsymbol{r}$, which is used in initiation and boundary issue. Logistic map can guarantee that the location of each nest will only be calculated once. The VLCS algorithm uses α_{new} and p_{anew} to replace the fixed parameters of the CS algorithm and improve the performance of CS algorithm. The pseudo code of VLCS algorithm is shown in Algorithm 4.

Algorithm 4: VLCS algorithm

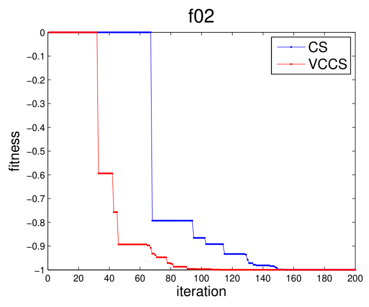
```
   Input: d represents the dimension;
          Objective function f(x), x=$(x_1, ..., x_d)^T$;
          Global random number r, r=$(r_1, \ldots, r_d)^T$,$r_d \in (0,1)$;
   Output: Postprocess results and visualization
 1 Initiation part uses Algorithm 2;
 2 n is equal to the number of nests;
 3 while $cur\_iteration < total\_iteration$ do
 4     Get a nest randomly by Lévy flights with $\alpha_{new}$;
 5     The location of nest is bounded by Algorithm 3;
 6     Then, evaluate its quality/fitness $F_i$;
 7     Choose a nest among n (say, j) with maximum fitness $F_j$;
 8     if $F_i > F_j$ then
 9         replace j by the new solution;
10     A fraction ($p_{anew}$) of worse nests are abandoned;
11     New ones are built by Algorithm 3;
12     Keep the best solutions (or nests with quality solutions);
13     Rank the solutions and find the current best;
14 Postprocess results and visualization;
15 final ;
16 return Postprocess results and visualization;
```

4. Simulation Experiments

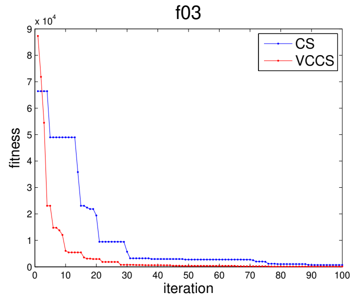
This paper selects 15 benchmark functions [34–36] to prove that the VLCS algorithm is better than the CS algorithm.The simulation environment is Matlab R2014a. The α_{new} and p_{anew} are shown in Equation (4). The benchmark functions and conditions are shown in Table 2. In Table 2, the dimensions of f02, f08 and f15 are two, and the others' dimensions are not less than ten. To compare with other algorithms, the *nest* number of n is 15 [18,24]. If the number of nest is changed, the relative convergence rates of the VLCS algorithm and the CS algorithm do not change. Of course, if n is increased, the VLCS algorithm and the CS algorithm will converge faster than other meta-heuristic algorithms. The CS [18] was proposed by Xin She Yang, who has already attested that it is better than other meta-heuristic algorithms. Section 3.1 illustrates the benefits of logistic map. Most of modified CS algorithms are mainly focused on choosing chaotic maps (e.g., [23–25]). These studies do not mention how to reduce repeated calculation in each dimension. This means the VLCS algorithm converges faster than most of other modified CS algorithms, because the VLCS algorithm uses logistic map of each dimension to initialize host nest location and update the location of host nest beyond the boundary. Therefore, this paper compares the CS algorithm and the VLCS algorithm, and shows the results in Figure 2. There is little difference between the VLCS algorithm and the CS algorithm in the function f04, f13 and f14 because the three functions are simpler than other functions in Figure 2 and do not need to carry out complex calculations. From the results, f07, f08, f09, f10 and f14 show two things clearly. Firstly, the convergence speed of the VLCS algorithm is faster than the CS algorithm, because, at the middle of the iteration, the CS algorithm easily falls into the local optimal solution and needs time to jump out of the local optimal solution. Secondly, the convergence accuracy of the VLCS algorithm is better than the CS algorithm because the VLCS algorithm does not do redundant calculations in the same place, even in the same dimension. Actually, if the other pictures in Figure 2 are enlarged, it is easy to get the same conclusion. The VLCS algorithm solves the three disadvantages of the CS algorithm mentioned in Section 2.2. In addition, Table 2 contains high-dimensional and low-dimensional test functions. This means the VLCS algorithm applies not only to low-dimensional problems but also to high-dimensional problems.



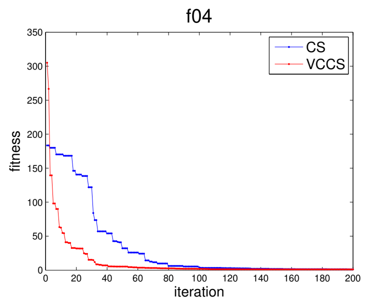
(**a**)



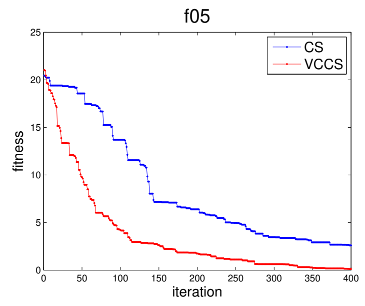
(**b**)



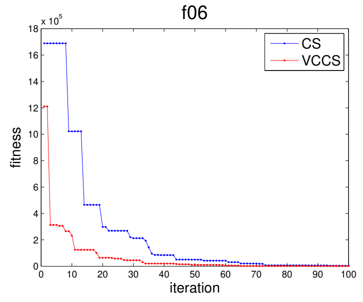
(**c**)



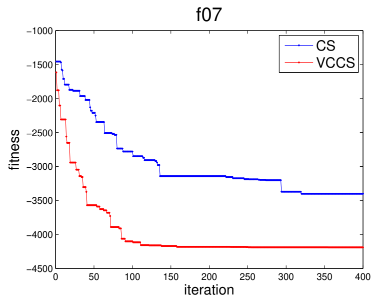
(**d**)



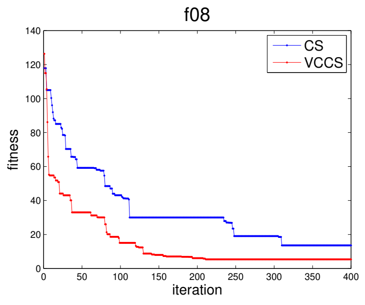
(**e**)



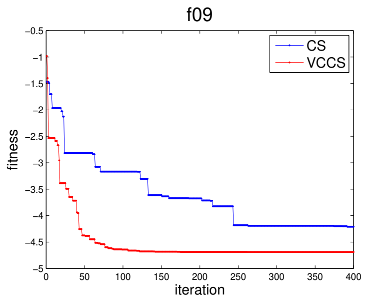
(**f**)



(**g**)



(**h**)



(**i**)

Figure 2. *Cont.*

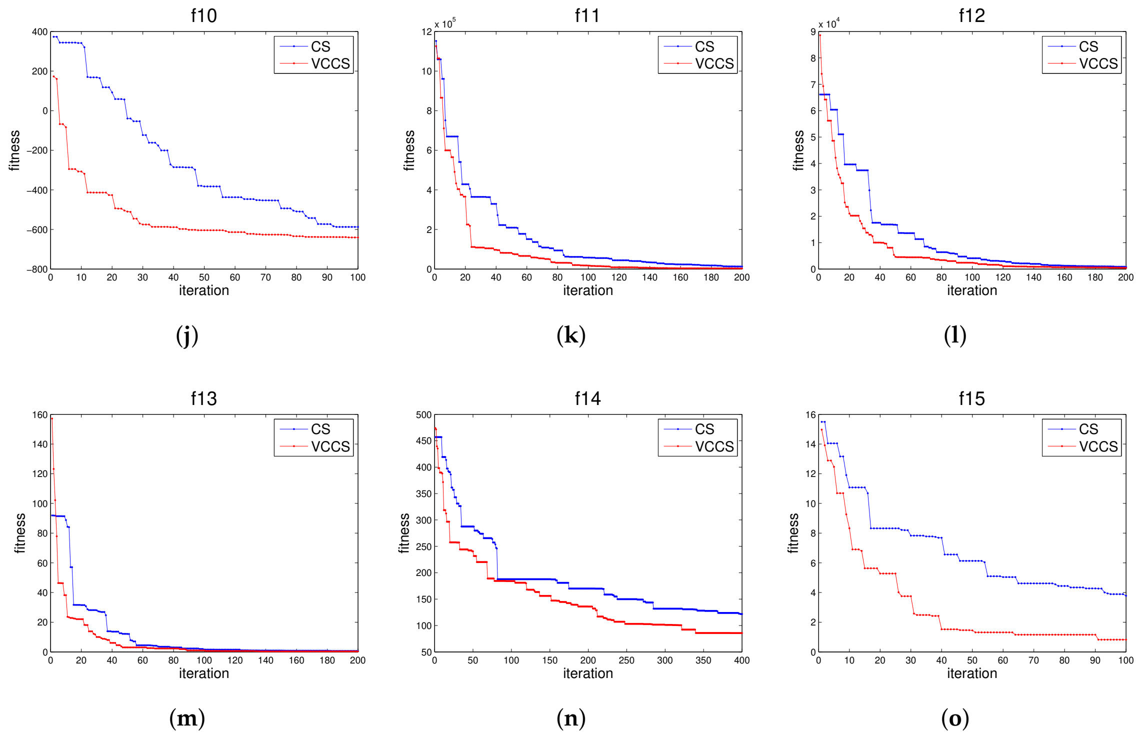


Figure 2. Convergence performance of the CS and the VLCS.

5. Conclusions

In this paper, the VLCS algorithm consists of coefficient function, the standard CS algorithm and each dimension with one logistic map. The logistic map is used to handle the initial problem and boundary issue. Using the logistic map guarantees to reduce the complexity of the VLCS algorithm and improve the computational efficiency. The reasons are analyzed in Section 3.1. The coefficient function is used to calculate α_{new} and p_{anew}. The coefficient function is used to change α_{new} and p_{anew} when the iteration increases. That is why the VLCS algorithm not only can reduce repeated calculation, but also can accelerate the speed of convergence. Furthermore, the VLCS algorithm can prevent itself from falling into a local optimum. The coefficient function is analyzed in Section 3.2. After the analysis of Section 4 simulation experiments, the VLCS algorithm is good at dealing with high dimension problems and low dimension problems.

In the future, we will come up with a more precise way to deal with α and p_a and use the VLCS algorithm to solve other practical engineering problems and real-world problems such as image hiding, power distribution, AODV routing protocol and so on. We believe that the VLCS will promote the development of the algorithm and be very useful in real-world problems.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (Grant No. 71633006 and Grant No. 61672540), and The Fund of Postgraduate Student Independent Innovation Project of Central South University (Grant No. 2017zzts733).

Author Contributions: L.L. and X.L. conceived and designed the experiments; X.L. performed the experiments; N.W. analyzed the data; P.Z. contributed reagents/materials/analysis tools; X.L. wrote the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shubin, G.R. Optimization Problem Formulation for Multidisciplinary Design. *Siam J. Optim.* **1993**, *4*, 754–776.
2. Ponsich, A.; Jaimes, A.L.; Coello, C.A.C. A Survey on Multiobjective Evolutionary Algorithms for the Solution of the Portfolio Optimization Problem and Other Finance and Economics Applications. *IEEE Trans. Evolut. Comput.* **2013**, *17*, 321–344.
3. Gaobo, Y.; Xingming, S.; Xiaojing, W. A Genetic Algorithm based Video Watermarking in the DWT Domain. In Proceedings of the International Conference on Computational Intelligence and Security, Guangzhou, China, 3–6 November 2006; pp. 1209–1212.
4. Tao, C.; Zhang, Y.; Jiang, J.J. Estimating system parameters from chaotic time series with synchronization optimized by a genetic algorithm. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **2007**, *76*, 016209, doi:10.1103/PhysRevE.76.016209.
5. Szpiro, G.G. Forecasting chaotic time series with genetic algorithms. *Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.* **1997**, *55*, 2557–2568.
6. Chou, Y.H.; Kuo, S.Y.; Chen, C.Y.; Chao, H.C. A Rule-Based Dynamic Decision-Making Stock Trading System Based on Quantum-Inspired Tabu Search Algorithm. *IEEE Access* **2014**, *2*, 883–896.
7. Wei, K.C.; Sun, X.; Chu, H.; Wu, C.C. Reconstructing permutation table to improve the Tabu Search for the PFSP on GPU. *J. Supercomput.* **2017**, *73*, 4711–4738.
8. Alidaee, B.; Ramalingam, V.P.; Wang, H.; Kethley, B. Computational experiment of critical event tabu search for the general integer multidimensional knapsack problem. *Ann. Oper. Res.* **2017**, 1–17, doi:10.1007/s10479-017-2675-0.
9. Bandyopadhyay, S.; Saha, S.; Maulik, U.; Deb, K. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. *IEEE Trans. Evolut. Comput.* **2008**, *12*, 269–283.
10. Mamano, N.; Hayes, W.B. SANA: Simulated annealing far outperforms many other search algorithms for biological network alignment. *Bioinformatics* **2017**, *33*, 2156–2164.
11. Angland, P.; Haberberger, D.; Ivancic, S.T.; Froula, D.H. Angular filter refractometry analysis using simulated annealing. *Rev. Sci. Instrum.* **2017**, *88*, 103510, doi:10.1063/1.4991511.
12. Fei, G. Parameter estimation for chaotic system based on particle swarm optimization. *Acta Phys. Sin.* **2006**, *55*, 577–582.
13. Garcia-Nieto, J.; Olivera, A.C.; Alba, E. Optimal Cycle Program of Traffic Lights With Particle Swarm Optimization. *IEEE Trans. Evolut. Comput.* **2013**, *17*, 823–839.
14. Salahi, M.; Jamalian, A.; Taati, A. Global minimization of multi-funnel functions using particle swarm optimization. *Neural Comput. Appl.* **2013**, *23*, 2101–2106.
15. Martens, D.; Backer, M.D.; Haesen, R.; Vanthienen, J.; Snoeck, M.; Baesens, B. Classification with Ant Colony Optimization. *IEEE Trans. Evolut. Comput.* **2007**, *11*, 651–665.
16. Yang, Q.; Chen, W.N.; Yu, Z.; Gu, T.; Li, Y.; Zhang, H.; Zhang, J. Adaptive Multimodal Continuous Ant Colony Optimization. *IEEE Trans. Evolut. Comput.* **2017**, *21*, 191–205.
17. Ye, K.; Zhang, C.; Ning, J.; Liu, X. Ant-colony algorithm with a strengthened negative-feedback mechanism for constraint-satisfaction problems. *Inf. Sci.* **2017**, *406–407*, 29–41.
18. Yang, X.S.; Deb, S. Cuckoo Search via Lévy Flights. In Proceedings of the World Congress on Nature & Biologically Inspired Computing, 2009 (NaBIC 2009), Coimbatore, India, 9–11 December 2010; pp. 210–214.
19. Shair, E.F.; Shen, Y.K.; Abdullah, A.R.; Jaafar, H.I.; Saharuddin, N.Z.; Abidin, A.F.Z. Cuckoo Search Approach for Cutting Stock Problem. *Int. J. Inf. Electron. Eng.* **2015**, *5*, 138–143.
20. Medjahed, S.A.; Saadi, T.A.; Benyettou, A.; Ouali, M. Binary cuckoo search algorithm for band selection in hyperspectral image classification. *IAENG Int. J. Comput. Sci.* **2015**, *42*, 1–9.
21. Walton, S.; Hassan, O.; Morgan, K.; Brown, M.R. Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos Solitons Fractals* **2011**, *44*, 710–718.
22. Li, X.; Yin, M. Modified cuckoo search algorithm with self adaptive parameter method. *Inf. Sci.* **2015**, *298*, 80–97.
23. Wang, G.G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. A Novel Cuckoo Search with Chaos Theory and Elitism Scheme. In Proceedings of the International Conference on Soft Computing and Machine Intelligence, New Delhi, India, 26–27 September 2015; pp. 64–69.

24. Huang, L.; Ding, S.; Yu, S.; Wang, J.; Lu, K. Chaos-enhanced Cuckoo search optimization algorithms for global optimization. *Appl. Math. Model.* **2016**, *40*, 3860–3875.
25. Liu, X.; Fu, M. Cuckoo search algorithm based on frog leaping local search and chaos theory. *Appl. Math. Comput.* **2015**, *266*, 1083–1092.
26. Zheng, H.; Zhou, Y. A novel Cuckoo Search optimization algorithm base on gauss distribution. *J. Comput. Inf. Syst.* **2012**, *8*, 4193–4200.
27. Li, M.; Cao, D. Hybrid optimization algorithm of Cuckoo Search and DE. *Comput. Eng. Appl.* **2013**, *49*, 57–60.
28. Chiroma, H.; Herawan, T.; Fister, I., Jr.; Fister, I.; Abdulkareem, S.; Shuib, L.; Hamza, M.F.; Saadi, Y.; Abubakar, A. Bio-Inspired Computation: Recent Development on the Modifications of the Cuckoo Search Algorithm. *Appl. Soft Comput.* **2017**, *61*, 149–173.
29. Yang, X.S.; Deb, S. Cuckoo search: Recent advances and applications. *Neural Comput. Appl.* **2014**, *24*, 169–174.
30. Sharma, A.; Singh, R.; Liaw, P.K.; Balasubramanian, G. Cuckoo searching optimal composition of multicomponent alloys by molecular simulations. *Scr. Mater.* **2017**, *130*, 292–296.
31. Kocarev, L.; Jakimoski, G. Logistic map as a block encryption algorithm. *Phys. Lett. A* **2001**, *289*, 199–206.
32. Wu, G.C.; Baleanu, D. Discrete fractional logistic map and its chaos. *Nonlinear Dyn.* **2013**, *75*, 283–287.
33. Jia, B.; Yu, B.; Wu, Q.; Wei, C.; Law, R. Adaptive affinity propagation method based on improved cuckoo search. *Knowl.-Based Syst.* **2016**, *111*, 27–35.
34. Chattopadhyay, R. A study of test functions for optimization algorithms. *J. Optim. Theory Appl.* **1971**, *8*, 231–236.
35. Shang, Y.W.; Qiu, Y.H. A Note on the Extended Rosenbrock Function. *Evolut. Comput.* **2006**, *14*, 119–126.
36. Schoen, F. A wide class of test functions for global optimization. *J. Glob. Optim.* **1993**, *3*, 133–137.