Improved Cuckoo Search Algorithm for Global Optimization

Ehsan Valian, Shahram Mohanna and Saeed Tavakoli

University of Sistan and Baluchestan

Abstract- The cuckoo search algorithm is a recently developed meta-heuristic optimization algorithm, which is suitable for solving optimization problems. To enhance the accuracy and convergence rate of this algorithm, an improved cuckoo search algorithm is proposed in this paper. Normally, the parameters of the cuckoo search are kept constant. This may lead to decreasing the efficiency of the algorithm. To cope with this issue, a proper strategy for tuning the cuckoo search parameters is presented. Considering several well-known benchmark problems, numerical studies reveal that the proposed algorithm can find better solutions in comparison with the solutions obtained by the cuckoo search. Therefore, it is anticipated that the improved cuckoo search algorithm can successfully be applied to a wide range of optimization problems.

Keywords: Cuckoo search algorithm, global optimization, Lévy flight, meta-heuristic, tuning

1. Introduction

Because of computational drawbacks of conventional numerical methods in solving complex optimization problems, researchers may have to rely on meta-heuristic algorithms. Over the last decades, many meta-heuristic algorithms have been successfully applied to various engineering optimization problems (Sim 2003; Qing 2006; Zhang 2006; Sickel 2007; Sanchis 2008; Marinakis 2008; Serrurier 2008). For most complicated real-world optimization problems, they have provided better solutions in comparison with conventional numerical methods.

To imitate natural phenomena, most meta-heuristic algorithms combine rules and randomness. These phenomena include the biological evolutionary processes, such as genetic algorithm (GA) (Holland 1975; Goldberg 1989), evolutionary algorithm (Fogel 1996; De Jong 1975) and differential evolution (DE) (Storn 1996), animal behavior, such as particle swarm optimization (PSO) (Kennedy 1995), tabu search (Glover 1977) and ant colony algorithm (ACA) (Dorigo 1996), as well as physical annealing processes, such as simulated annealing (SA) (Kirkpatrick 1983).

imitating animal behavior. The optimal solutions obtained by the CS are far better than the best solutions obtained by efficient particle swarm optimizers and genetic algorithms (Yang 2010). This paper develops an Improved Cuckoo Search (ICS) algorithm for unconstrained optimization problems. To enhance accuracy and convergence rate of the CS, the ICS employs an improved method for generating new solution vectors.

2. Cuckoo search algorithm

To describe the CS more clearly, the breed behavior of certain cuckoo species is briefly reviewed.

2.1. Cuckoo breeding behaviour

The CS was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of host birds. Some cuckoos have evolved in such a way that female parasitic cuckoos can imitate the colors and patterns of the eggs of a few chosen host species. This reduces the probability of the eggs being abandoned and, therefore, increases their re-productivity (Payne 2005). It is worth mentioning that several host birds engage direct conflict with intruding cuckoos. In this case, if host birds discover the eggs are not their own, they will either throw them away or simply abandon their nests and build new ones, elsewhere.

Parasitic cuckoos often choose a nest where the host bird just laid its own eggs. In general, the cuckoo eggs hatch slightly earlier than their host eggs. Once the first cuckoo chick is hatched, his first instinct action is to evict the host eggs by blindly propelling the eggs out of the nest. This action results in increasing the cuckoo chick's share of food provided by its host bird (Payne 2005). Moreover, studies show that a cuckoo chick can imitate the call of host chicks to gain access to more feeding opportunity.

The CS models such breeding behavior and, thus, can be applied to various optimization problems. Yang and Deb

(Yang 2009; Yang 2010), discovered that the performance of the CS can be improved by using Lévy Flights instead of simple random walk.

2.2. *Lévy Flights*

In nature, animals search for food in a random or quasi-random manner. Generally, the foraging path of an animal is effectively a random walk because the next move is based on both the current location/state and the transition probability to the next location. The chosen direction implicitly depends on a probability, which can be modeled mathematically. Various studies have shown that the flight behavior of many animals and insects demonstrates the typical characteristics of Lévy flights (Brown 2007). A Lévy flight is a random walk in which the step-lengths are distributed according to a heavy-tailed probability distribution. After a large number of steps, the distance from the origin of the random walk tends to a stable distribution.

2.3. *Cuckoo Search Implementation*

Each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The aim is to employ the new and potentially better solutions (cuckoos) to replace not-so-good solutions in the nests. In the simplest form, each nest has one egg. The algorithm can be extended to more complicated cases in which each nest has multiple eggs representing a set of solutions (Yang 2009; Yang 2010). The CS is based on three idealized rules:

• Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;

• The best nests with high quality of eggs (solutions) will carry over to the next generations;

• The number of available host nests is fixed, and a host can discover an alien egg with probability $p_a \in [0,1]$. In this case, the host bird can either throw the egg away or abandon the nest to build a completely new nest in a new location (Yang 2009).

For simplicity, the last assumption can be approximated by a fraction p_a of the n nests being replaced by new nests, having new random solutions. For a maximization problem, the quality or fitness of a solution can simply be proportional to the objective function. Other forms of fitness can be defined in a similar way to the fitness function in genetic algorithms (Yang 2009).

Based on the above-mentioned rules, the basic steps of the CS can be summarized as the pseudo code, as follows (Yang 2009):

```
begin
  Objective function f(x),   x = (x_1,....., x_d)^T
  Generate initial population of
     n host nests x_i (i = 1,2,..., n)
  While (t < MaxGeneration) or (stop criterion)
    Get a cuckoo randomly by Lévy flights
     evaluate its quality / fitness F_i
    Choose a nest among n (say. j) randomly
    if (F_i > F_j),
       replace j by the new solution;
    end if
    A fraction (p_a) of worse nests
        are abandoned and new ones are built;
   Keep the best solutions
        (or nests with quality solutions);
   Rank the solutions and find the current best
  end while
  Postprocess results and visualization
end
```

When generating new solutions $x_i(t+1)$ for the i^{th} cuckoo, the following Lévy flight is performed

$$x_i(t+1) = x_i(t) + \alpha \oplus \text{Lévy}(\lambda) \quad (1)$$

where $\alpha > 0$ is the step size, which should be related to the scale of the problem of interest. The product $\oplus$ means entry-wise multiplications (Yang 2010). In this research work, we consider a Lévy flight in which the step-lengths are distributed according to the following probability distribution

$$\text{Lévy}\, u = t^{-\lambda}, 1 < \lambda \le 3 \quad (2)$$

which has an infinite variance. Here, the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail.

It is worth pointing out that, in the real world, if a cuckoo's egg is very similar to a host's eggs, then this cuckoo's egg is less likely to be discovered, thus the fitness should be related to the difference in solutions. Therefore, it is a good idea to do a random walk in a biased way with some random step sizes (Yang 2009).

3. Improved Cuckoo Search

The parameters p_a, λ and α introduced in the CS help the algorithm to find globally and locally improved solutions, respectively. The parameters p_a and α are very important parameters in fine-tuning of solution vectors, and can be potentially used in adjusting convergence rate of algorithm. The traditional CS algorithm uses fixed value for both p_a and α. These values are set in the initialization step and cannot be changed during new generations. The main drawback of this method appears in the number of iterations to find an optimal solution. If the value of p_a is small and the value of α is large, the performance of the algorithm will be poor and leads to considerable increase in number of iterations. If the value of p_a is large and the value of α is small, the speed of convergence is high but it may be unable to find the best solutions.

The key difference between the ICS and CS is in the way of adjusting p_a and α. To improve the performance of the CS algorithm and eliminate the drawbacks lies with fixed values of p_a and α, the ICS algorithm uses variables p_a and α .In the early generations, the values of p_a and α must be big enough to enforce the algorithm to increase the diversity of solution vectors. However, these values should be decreased in final generations to result in a better fine-tuning of solution vectors. The values of p_a and α are dynamically changed with the number of generation and expressed in Equations 3-5, where NI and gn are the number of total iterations and the current iteration, respectively.

$$P_a(gn) = P_{a\,max} - \frac{gn}{NI}\left(P_{a\,max} - P_{a\,min}\right) \qquad (3)$$

$$\alpha(gn) = \alpha_{max}\exp(c.gn) \qquad (4)$$

$$c = \frac{1}{NI}\,Ln\left(\frac{\alpha_{min}}{\alpha_{max}}\right) \qquad (5)$$

4. Case studies: analysis and discussion

To verify the reliability of ICS algorithm, several well-known test functions (Zou 2010), as shown in Table 1, are considered. In the experiments, the parameters of CS and ICS algorithms are shown in Table 2, where N is the number of decision variables.

4.1. Comparison of the ICS and CS algorithms

To optimize the given test functions, a MATLAB code, using MATLAB Ver.7.10, is developed based on the ICS. In this version of MATLAB, the numbers smaller than 4.9407e-324 are considered as zero. The PC used is an INTEL32, X2, 3.0GHz having 4GB of memory. To show the effectiveness of the proposed algorithm, it is compared with the CS algorithms.

Considering functions $f_1 - f_{15}$, Tables 3-5 show the optimization results of the implementation of the CS and ICS algorithms for N=10, 30 and 50. Thirty independent experiments are carried out in each case, and the optimization results are reported. In these tables, the parameter "iter"' refers to the maximum iteration number, and the parameter "SD" represents the standard deviation.

Considering the best, worst, mean and SD criteria, it can be seen from Table 3 that all best results for N=10 are given by the ICS.

For N=30, Table 4 shows that the ICS leads to better results than the CS in all criteria, in all test functions except f_2, f_7, f_8, f_{10}, f_{11}, and f_{12}. For f_8, f_{10}, and f_{11} the CS gives better results than the ICS in all criteria. For f_2 and f_{12} the worst, mean and SD criteria provided by the ICS is better than those given by the CS.

For N=50, Table 5 shows that the ICS leads to better results than the CS in all criteria, in f_1, f_3, f_4, f_6, f_9, f_{12}, f_{13}, and f_{14}. For f_7, f_8, f_{10}, and f_{11} the CS gives better results than the ICS in all criteria. Given the second test function, the worst, mean and SD criteria provided by the ICS is better than those given by the CS. For f_5, and f_{15} the ICS manages to better results in terms of the best, worst and mean criteria.

4.2. Effects of changing the optimization parameters on the performance of the ICS

In this subsection, the effect of changing p_a and α on the performance of the ICS is investigated. For N=10, N=30 and N=50, the iteration number is set to 1000, 3000 and 5000 and the effects of changing p_a and α on the performance of the ICS are shown in Tables 6-8 and Tables 9-11, respectively. Thirty independent runs are carried out in each case and the mean and SD values are obtained.

As can be seen from Tables 6-8, if the minimum value of p_a is decreased with no change in the value of α, better results may be obtained in most cases. It seems that for test functions with high decision variables, an increase in the maximum value of p_a leads to better results.

Tables 9-11 show that a reduction in the minimum value of α with no change in the value of p_a does not have any significant effect on the performance of the algorithm. However, the performance of the algorithm may deteriorate by an increase in the maximum value of α. Various tests show that the suitable algorithm parameters leading to good results can approximately be $p_{a\,min} = 0.005$, $p_{a\,max} = 1$, α_{min} =0.05, and $\alpha_{max} = 0.5$.

5. Conclusions

In this paper, an improved cuckoo search algorithm enhancing the accuracy and convergence rate of the cuckoo search algorithm was proposed. The impact of keeping the parameters of the cuckoo search algorithm constant was discussed and a strategy for improving the performance of the algorithm by properly tuning these parameters was presented. According to the simulation results, the proposed approach performed well in several benchmark problems in terms of the accuracy of the solutions found.

References

Brown C, Liebovitch LS, Glendon R (2007) Lévy flights in Dobe Ju/hoansi foraging patterns, Hum Ecol. 35:129-138.

De Jong K (1975) Analysis of the behavior of a class of genetic adaptive systems, Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Dorigo M, Maniezzo V, Golomi A (1996) Ant system: optimization by a colony of cooperating agents. IEEE T Syst Man Cy B 26 (1) 29–41.

Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution, John Wiley, Chichester, UK.

Glover F (1977) Heuristic for integer programming using surrogate constraints, Decision Sci. 8 (1):156–166.

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning, Addison Wesley, Boston, USA.

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, USA.

Kennedy J, Eberhart RC (1995) Particle swarm optimization, In: Proceedings of IEEE International Conference on Neural Networks. 1942–1948.

Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing, Sciene. 220:671–680.

Koza JR (1990) Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems,Stanford University, USA.

Marinakis Y, Marinaki M, Dounias G (2008) Particle swarm optimization for pap- smear diagnosis. Expert Syst Appl 35 (4):1645–1656.

Payne R B, Sorenson M D, Klitz K (2005) The cuckoos. Oxford University Press.

Qing AY (2006) Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems. IEEE T Geosci Remote 44 (1):116–125.

Sanchis J, Martanez MA, Blasco X (2008) Integrated multiobjective optimization and a priori preferences using genetic algorithms. Inform Sciences 178(4):931–951.

Serrurier M, Prade H (2008) Improving inductive logic programming by using simulated annealing. Inform Sciences 178 (6):1423–1441.

Sickel JHV, Lee KY, Heo JS (2007) Differential evolution and its applications to power plant control. In: International Conference on Intelligent Systems Applications to Power Systems, ISAP, 1–6.

Sim KM, Sun WH (2003) Ant colony optimization for routing and load-balancing: survey and new directions. IEEE T Syst Man Cy A 33 (5):560–572.

Storn R (1996) Differential evolution design of an IIR-filter. Paper presented in: IEEE International Conference on Evolutionary Computation, Nagoya, 268–273 1996.

Yang XS, Deb S (2009) Cuckoo search via Lévy flights, In: Proceeings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India), 210-214.

Yang XS, Deb S (2010) Engineering Optimisation by Cuckoo Search, Int. J. Mathematical Modelling and Numerical Optimisation, 1(4):330–343.

Zhang JJ, Zhang YH, Gao RZ (2006) Genetic algorithms for optimal design of vehicle suspensions. Paper presented in IEEE International Conference on Engineering of Intelligent Systems, 1–6.

Zou DX, et al., (2010) Novel global harmony search algorithm for unconstrained problems, Neurocomputing. 73:3308-3318.

Table1: Benchmark problems

Name	Test Function	Search Space	Optimum
f_1 (Sphere function)	$\min f_1 = \sum_{i=1}^{N} x_i^2$	$[-100,100]^n$	0
f_2 (Rosenbrock function)	$\min f_2 = \sum_{i=1}^{N-1}\left(100\left(x_{i-1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right)$	$[-100,100]^n$	0
f_3 (Generalized Rastrigrin function)	$\min f_3 = \sum_{i=1}^{N}\left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	$[-10,10]^n$	0
f_4 (Generalized Griewank function)	$\min f_4 = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^n$	0
f_5 (Ackley's function)	$\min f_5 = 20 + e - 20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}}\right) - \exp\left(\frac{\sum_{i=1}^{N}\cos\left(2\pi x_i\right)}{N}\right)$	$[-32,32]^n$	0
f_6 (Schwefel's problem 2.22)	$\min f_6 = \sum_{i=1}^{N}\lvert x_i\rvert + \prod_{i=1}^{N}\lvert x_i\rvert$	$[-100,100]^n$	0
f_7 (Schwefel's problem 2.26)	$\min f_7 = 418.9829N - \sum_{i=1}^{N}\left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	$[-500,500]^n$	n=10, 1.2728e-04 n=30, 3.8183e-04 n=50, 6.3638e-04
f_8 (Rotated hyper-ellipsoid function)	$\min f_8 = \sum_{i=1}^{N}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100,100]^n$	0
f_9	$\min f_9 = \sum_{i=1}^{N} z_i^2 - 450$	$[-100,100]^n$	-450
f_{10}	$\min f_{10} = \sum_{i=1}^{N}\left(\sum_{j=1}^{i} z_j\right)^2 - 450$	$[-100,100]^n$	-450
f_{11}	$\min f_{11} = \sum_{i=1}^{N}\left(\sum_{j=1}^{i} z_j\right)^2\left(1 + 0.4 * \lvert N(0,1)\rvert\right) - 450$	$[-100,100]^n$	-450
f_{12}	$\min f_{12} = \sum_{i=1}^{N-1}\left(100\left(z_{i-1} - z_i^2\right)^2 + \left(z_i - 1\right)^2\right) + 390$	$[-100,100]^n$	390
f_{13}	$\min f_{13} = \frac{1}{4000}\sum_{i=1}^{N} z_i^2 - \prod_{i=1}^{N}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 - 180$	$[-600,600]^n$	-180
f_{14}	$\min f_{14} = 20 + e - 20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{N} z_i^2}{N}}\right) - \exp\left(\frac{\sum_{i=1}^{N}\cos\left(2\pi z_i\right)}{N}\right) - 140$	$[-32,32]^n$	-140
f_{15}	$\min f_{15} = \sum_{i=1}^{N}\left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) - 330$	$[-5,5]^n$	-330

Table 2: Algorithms' parameters

Algorithms	Number of Dimension	Number of Generation	p_a	λ	α
CS	N=10 N=30 N=50	1000 3000 5000	0.1	1.5	0.25
ICS	N=10 N=30 N=50	1000 3000 5000	p_a (max) = 0.5 p_a (min) = 0.05	1.5	α(max) = 0.5 α(min) = 0.01

Table 3: The optimization results of the CS and ICS algorithms for $f_1 - f_{15}$ (N=10)

Function	Algorithm	iter	Best	Worst	Mean	SD
f_1	CS	1000	4.9826e-18	1.5338e-15	2.5086e-16	3.5572e-16
	ICS	1000	**2.3482e-25**	**5.9057e-21**	**4.2599e-22**	**1.1657e-21**
f_2	CS	1000	7.5742e-01	7.4146e+01	5.1543e+00	1.3235e+01
	ICS	1000	**2.3671e-02**	**1.1031e+01**	**3.7763e+00**	**3.2777e+00**
f_3	CS	1000	2.4935e+00	1.1197e+01	5.8173e+00	2.2131e+00
	ICS	1000	**1.4728e+00**	**5.0832e+00**	**3.1597e+00**	**9.1871e-01**
f_4	CS	1000	1.2824e-02	1.0010e-01	5.3369e-02	2.1703e-02
	ICS	1000	**2.7491e-03**	**6.3613e-02**	**3.4795e-02**	**1.6791e-02**
f_5	CS	1000	3.1724e-08	1.1169e-05	1.1978e-06	2.4007e-06
	ICS	1000	**2.1651e-11**	**3.4678e-06**	**1.5947e-07**	**6.5468e-07**
f_6	CS	1000	8.7718e-06	3.9122e-03	2.5590e-04	7.0628e-04
	ICS	1000	**1.8418e-11**	**1.2817e-08**	**8.0449e-10**	**2.3225e-09**
f_7	CS	1000	2.0399e+02	8.1834e+02	4.7802e+02	1.8304e+02
	ICS	1000	**7.1429e-01**	**6.9924e+02**	**3.8381e+02**	**1.6208e+02**
f_8	CS	1000	8.5143e-09	1.6294e-06	2.3584e-07	3.5945e-07
	ICS	1000	**3.5209e-11**	**1.0386e-07**	**1.5240e-08**	**2.6462e-08**
f_9	CS	1000	**-450**	**-450**	**-450**	2.9856e-14
	ICS	1000	**-450**	**-450**	**-450**	**1.8283e-14**
f_{10}	CS	1000	**-450**	-4.5000e+02	**-450**	3.7084e-07
	ICS	1000	**-450**	**-450**	**-450**	**6.2205e-08**
f_{11}	CS	1000	-4.5000e+02	-4.4990e+02	-4.4999e+02	2.0502e-02
	ICS	1000	**-450**	**-4.5000e+02**	**-4.5000e+02**	**4.2198e-04**
f_{12}	CS	1000	3.9031e+02	4.7311e+02	3.9788e+02	1.5458e+02
	ICS	1000	**3.9011e+02**	**4.0259e+02**	**3.9329e+02**	**2.9123e+00**
f_{13}	CS	1000	-1.7998e+02	-1.7992e+02	-1.7995e+02	1.5447e-02
	ICS	1000	**-1.8000e+02**	**-1.7993e+02**	**-1.7996e+02**	**1.4554e-02**
f_{14}	CS	1000	**-140**	-1.3884e+02	-1.3996e+02	2.1090e-01
	ICS	1000	**-140**	**-140**	**-140**	**2.9775e-07**
f_{15}	CS	1000	-3.2848e+02	-3.1921e+02	-3.2422e+02	2.2932e+00
	ICS	1000	**-3.2979e+02**	**-3.2421e+02**	**-3.2698e+02**	**1.0785e+00**

Table 4: The optimization results of the CS and ICS algorithms for $f_1 - f_{15}$ (N=30)

Function	Algorithm	iter	Best	Worst	Mean	SD
f_1	CS	3000	1.2421e-15	7.3248e-13	6.6222e-14	1.3432e-13
	ICS	3000	**2.9807e-22**	**4.1015e-20**	**9.5438e-21**	**1.1279e-20**
f_2	CS	3000	**1.8759e+00**	1.5392e+02	4.0384e+01	3.5250e+01
	ICS	3000	9.8653e+00	**7.9836e+01**	**2.6678e+01**	**1.3697e+01**
f_3	CS	3000	2.0423e+01	5.9923e+01	3.6513e+01	9.2702e+00
	ICS	3000	**1.1939e+01**	**2.9747e+01**	**2.2296e+01**	**4.1242e+00**
f_4	CS	3000	1.6098e-14	7.5394e-02	1.0846e-02	2.0119e-02
	ICS	3000	**1.1102e-16**	**4.9058e-08**	**3.1173e-09**	**1.1340e-08**
f_5	CS	3000	5.7283e-07	4.3404e+00	2.0926e+00	8.4421e-01
	ICS	3000	**7.0179e-10**	**2.0393e+00**	**3.0880e-01**	**5.9632e-01**
f_6	CS	3000	1.4660e-05	9.8736e+01	3.7578e+00	1.7985e+01
	ICS	3000	**2.1208e-10**	**2.8235e-06**	**2.1058e-07**	**5.4655e-07**
f_7	CS	3000	1.4402e+03	**3.1537e+03**	**2.4094e+03**	4.3460e+02
	ICS	3000	**1.4154e+03**	3.3428e+03	2.5541e+03	**4.1141e+02**
f_8	CS	3000	**7.9269e-01**	**8.0358e+00**	**2.8559e+00**	**1.6281e+00**
	ICS	3000	1.1422e+00	1.5802e+01	5.2848e+00	3.4241e+00
f_9	CS	3000	**-450**	**-450**	**-450**	2.3366e-13
	ICS	3000	**-450**	**-450**	**-450**	**1.0449e-13**
f_{10}	CS	3000	**-4.4936e+02**	**-4.4103e+02**	**-4.4762e+02**	**1.8178e+00**
	ICS	3000	-4.4847e+02	-4.0914e+02	-4.4034e+02	8.6941e+00
f_{11}	CS	3000	**-2.3566e+02**	**6.9350e+02**	**2.4178e+02**	**2.8887e+02**
	ICS	3000	-1.5538e+02	1.3238e+03	3.7542e+02	3.3414e+02
f_{12}	CS	3000	**3.9042e+02**	5.6100e+02	4.4837e+02	4.2947e+01
	ICS	3000	3.9679e+02	**4.7957e+02**	**4.1953e+02**	**2.1417e+01**
f_{13}	CS	3000	-1.8000e+02	-1.7997e+02	-1.8000e+02	7.9881e-03
	ICS	3000	**-180**	**-180**	**-180**	**2.0596e-10**
f_{14}	CS	3000	-1.4000e+02	-1.3373e+02	-1.3786e+02	1.0946e+00
	ICS	3000	**-140**	**-1.3577e+02**	**-1.3978e+02**	**8.2844e-01**
f_{15}	CS	3000	-3.1378e+02	-2.8555e+02	-3.0235e+02	7.0745e+00
	ICS	3000	**-3.1666e+02**	**-2.9349e+02**	**-3.0686e+02**	**5.5904e+00**

Table 5: The optimization results of the CS and ICS algorithms for $f_1 - f_{15}$ (N=50)

Function	Algorithm	iter	Best	Worst	Mean	SD
f_1	CS	5000	5.4547e-13	4.9841e-10	2.7691e-11	9.0314e-11
	ICS	5000	**1.6080e-20**	**1.1525e-18**	**2.0675e-19**	**2.3478e-19**
f_2	CS	5000	**9.8038e+00**	7.3955e+02	1.5139e+02	1.2737e+02
	ICS	5000	3.3689e+01	**1.7960e+02**	**7.6080e+01**	**3.9611e+01**
f_3	CS	5000	4.6418e+01	1.0788e+02	7.1486e+01	1.3445e+01
	ICS	5000	**3.3463e+01**	**7.1683e+01**	**5.3111e+01**	**1.0093e+01**
f_4	CS	5000	1.9523e-12	5.1408e-02	8.5133e-03	1.4217e-02
	ICS	5000	**3.3307e-16**	**3.6921e-09**	**1.2358e-10**	**6.7400e-10**
f_5	CS	5000	2.2001e+00	6.8027e+00	4.0143e+00	**1.0781e+00**
	ICS	5000	**8.2101e-09**	**3.5109e+00**	**8.0832e-01**	1.2424e+00
f_6	CS	5000	8.8532e-04	2.4962e+02	2.7147e+01	6.2970e+01
	ICS	5000	**3.9764e-10**	**8.7367e-05**	**3.8818e-06**	**1.59140e-05**
f_7	CS	5000	**2.7726e+03**	**6.0688e+03**	**4.4152e+03**	**7.2422e+02**
	ICS	5000	2.9766e+03	6.1253e+03	4.7645e+03	7.6395e+02
f_8	CS	5000	**2.9885e+01**	**1.9126e+02**	**9.1219e+01**	**4.4384e+01**
	ICS	5000	3.4801e+01	3.5944e+02	1.7395e+02	7.4926e+01
f_9	CS	5000	**-450**	**-450**	**-450**	4.1492e-11
	ICS	5000	**-450**	**-450**	**-450**	**2.2466e-13**
f_{10}	CS	5000	**-4.2126e+02**	**-3.1512e+02**	**-3.7815e+02**	**3.0840e+01**
	ICS	5000	-3.9594e+02	-6.3480e+01	-2.4841e+02	7.1120e+01
f_{11}	CS	5000	**1.7480e+03**	**5.9222e+03**	**3.1950e+03**	**9.4380e+02**
	ICS	5000	2.0394e+03	6.1629e+03	4.1276e+03	1.1680e+03
f_{12}	CS	5000	4.1866e+02	1.2019e+03	5.6663e+02	1.5325e+02
	ICS	5000	**4.1500e+02**	**5.5800e+02**	**4.6508e+02**	**4.2375e+01**
f_{13}	CS	5000	**-180**	-1.7991e+02	-1.7999e+02	2.1715e-02
	ICS	5000	**-180**	**-180**	**-180**	**2.7292e-12**
f_{14}	CS	5000	-1.3805e+02	-1.3008e+02	-1.3577e+02	**1.3777e+00**
	ICS	5000	**-1.4000e+02**	**-1.3107e+02**	**-1.3897e+02**	2.0029e+00
f_{15}	CS	5000	-2.9677e+02	-2.4743e+02	-2.7692e+02	**1.1287e+01**
	ICS	5000	**-2.9753e+02**	**-2.6720e+02**	**-2.8149e+02**	6.5472e+01

Table 6: The effect of changing p_a with no change in α on the performance of the ICS (N=10)

Function		$P_a = 0.5 - 0.05$	$P_a = 1 - 0.005$	$P_a = 0.5 - 0.005$
f_1	Mean	4.2599e-22	4.4653e-25	**1.0340e-26**
	SD	1.1657e-21	8.3606e-25	**2.5207e-26**
f_2	Mean	3.7763e+00	4.9089e+00	**2.3611e+00**
	SD	3.2777e+00	5.5049e+00	**3.1290e+00**
f_3	Mean	3.1597e+00	**1.8303e+00**	2.8445e+00
	SD	**9.1871e-01**	9.5162e-01	1.7087e+00
f_4	Mean	**3.4795e-02**	3.8763e-02	3.9614e-02
	SD	**1.6791e-02**	1.8304e-02	2.1058e-02
f_5	Mean	1.5947e-07	1.0920e-09	**3.6240e-10**
	SD	6.5468e-07	2.8637e-09	**1.7380e-09**
f_6	Mean	8.0449e-10	**8.2101e-13**	1.3079e-12
	SD	2.3225e-09	**1.2657e-12**	2.9891e-12
f_7	Mean	3.8381e+02	**2.5381e+02**	3.3117e+02
	SD	1.6208e+02	**1.0782e+02**	1.4635e+02
f_8	Mean	1.5240e-08	1.5016e-06	**1.3773e-09**
	SD	2.6462e-08	2.9948e-06	**2.1917e-09**
f_9	Mean	**-450**	**-450**	**-450**
	SD	1.8283e-14	**0**	1.4928e-14
f_{10}	Mean	**-450**	**-450**	**-450**
	SD	6.2205e-08	2.3677e-06	**3.3367e-08**
f_{11}	Mean	**-4.5000e+02**	-4.4999e+02	**-4.5000e+02**
	SD	4.2198e-04	1.1282e-02	**1.1264e-04**
f_{12}	Mean	3.9329e+02	3.9846e+02	**3.9247e+02**
	SD	2.9123e+00	1.1820e+02	**2.4899e+00**
f_{13}	Mean	**-1.8000e+02**	**-1.7996e+02**	**-1.7996e+02**
	SD	**1.4554e-02**	1.9431e-02	2.0279e-02
f_{14}	Mean	**-140**	-1.4000e+02	**-140**
	SD	2.9775e-07	1.3034e-03	**5.9855e-10**
f_{15}	Mean	-3.2698e+02	**-3.2831e+02**	-3.2726e+02
	SD	1.0785e+00	**7.4429e-01**	1.3477e+00

Table 7: The effect of changing p_a with no change in α on the performance of the ICS (N=30)

Function		$P_a = 0.5 - 0.05$	$P_a = 1 - 0.005$	$P_a = 0.5 - 0.005$
f_1	Mean	9.5438e-21	**4.6272e-25**	2.0677e-24
	SD	1.1279e-20	**5.1117e-25**	5.6379e-24
f_2	Mean	**2.6678e+01**	3.3602e+01	3.1223e+01
	SD	**1.3697e+01**	1.9880e+01	2.2979e+01
f_3	Mean	**2.2296e+01**	2.9557e+01	2.5750e+01
	SD	**4.1242e+00**	4.1747e+00	6.9751e+00
f_4	Mean	**3.1173e-09**	4.7222e-09	2.1770e-07
	SD	**1.1340e-08**	2.1088e-08	8.2739e-07
f_5	Mean	3.0880e-01	1.4052e-01	**3.8519e-02**
	SD	5.9632e-01	5.3836e-01	**2.1096e-01**
f_6	Mean	2.1058e-07	**1.0914e-12**	1.2310e-10
	SD	5.4655e-07	**3.1045e-12**	4.4570e-10
f_7	Mean	2.5541e+03	**2.2606e+03**	2.4286e+03
	SD	4.1141e+02	**3.6773e+02**	5.7437e+02
f_8	Mean	5.2848e+00	4.8976e+01	**4.7942e+00**
	SD	**3.4241e+00**	2.5553e+01	3.4457e+00
f_9	Mean	**-450**	**-450**	**-450**
	SD	1.0449e-13	**2.1111e-14**	6.9217e-14
f_{10}	Mean	-4.4034e+02	-3.9115e+02	**-4.4513e+02**
	SD	8.6941e+00	4.4652e+02	**3.3119e+00**
f_{11}	Mean	3.7542e+02	1.0171e+03	**2.8481e+02**
	SD	**3.3414e+02**	5.4128e+02	3.7925e+02
f_{12}	Mean	**4.1953e+02**	4.2623e+02	4.2345e+02
	SD	**2.1417e+01**	2.5360e+01	2.4905e+01
f_{13}	Mean	**-180**	**-180**	**-180**
	SD	**2.0596e-10**	9.7676e-07	1.6358e-07
f_{14}	Mean	-1.3978e+02	**-1.400e+02**	-1.3962e+02
	SD	8.2844e-01	**8.2856e-03**	2.1090e-01
f_{15}	Mean	**-3.0686e+02**	-3.0164e+02	-3.0661e+02
	SD	5.5904e+00	**5.4853e+00**	6.8482e+00

Table 8: The effect of changing p_a with no change in α on the performance of the ICS (N=50)

Function		$P_a = 0.5 - 0.05$	$P_a = 1 - 0.005$	$P_a = 0.5 - 0.005$
f_1	Mean	2.0675e-19	3.1272e-23	**9.0087e-24**
	SD	2.3478e-19	4.3418e-23	**1.0870e-23**
f_2	Mean	7.6080e+01	**6.7831e+01**	7.9552e+01
	SD	3.9611e+01	**2.9552e+01**	5.2299e+01
f_3	Mean	**5.3111e+01**	6.2762e+01	5.8441e+01
	SD	**1.0093e+01**	1.3800e+01	1.1241e+01
f_4	Mean	1.2358e-10	**8.9732e-14**	2.4653e-04
	SD	6.7400e-10	**4.0347e-13**	1.3503e-03
f_5	Mean	8.0832e-01	1.8687e+00	**3.5381e-01**
	SD	1.2424e+00	4.9738e+00	**9.2685e-01**
f_6	Mean	3.8818e-06	**7.4982e-12**	3.0294e-10
	SD	1.59140e-05	**1.1453e-11**	1.0090e-09
f_7	Mean	4.7645e+03	**4.6308e+03**	4.4703e+03
	SD	7.6395e+02	**6.1886e+02**	9.6839e+02
f_8	Mean	1.7395e+02	8.8711e+02	**1.1761e+01**
	SD	7.4926e+01	2.5878e+02	**6.0230e+01**
f_9	Mean	**-450**	**-450**	**-450**
	SD	2.2466e-13	**1.2035e-13**	1.2534e-13
f_{10}	Mean	-2.4841e+02	4.9511e+02	**-3.1568e+02**
	SD	**7.1120e+01**	2.6049e+02	7.8122e+01
f_{11}	Mean	4.1276e+03	7.1806e+03	**3.5103e+02**
	SD	**1.1680e+03**	2.4540e+03	1.3683e+02
f_{12}	Mean	4.6508e+02	**4.4468e+02**	4.6210e+02
	SD	4.2375e+01	**2.0388e+01**	3.8253e+01
f_{13}	Mean	**-180**	**-180**	-1.800e+02
	SD	2.7292e-12	**2.6177e-13**	2.5572e-03
f_{14}	Mean	-1.3897e+02	-1.3956e+02	**-1.3963e+02**
	SD	**2.0029e+00**	8.6614e+01	8.6484-01
f_{15}	Mean	-2.8149e+02	-2.6482e+02	**-2.8186e+02**
	SD	6.5472e+01	1.2553e+01	**9.0449e+00**

Table 9: The effect of changing α with no change in p_a on the performance of the ICS (N=10)

Function		$\alpha = 0.5 - 0.01$	$\alpha = 1 - 0.001$	$\alpha = 0.5 - 0.001$
f_1	Mean	4.2599e-22	**7.0038e-23**	8.9021e-20
	SD	1.1657e-21	**1.6336e-22**	3.5120e-19
f_2	Mean	**3.7763e+00**	5.9300e+00	8.6980e+00
	SD	3.2777e+00	1.2964e+01	**1.7134e+00**
f_3	Mean	**3.1597e+00**	3.3373e+00	3.9880e+00
	SD	**9.1871e-01**	1.3127e+00	1.3427e+00
f_4	Mean	3.4795e-02	4.0566e-02	**3.2937e-02**
	SD	1.6791e-02	1.8490e-02	**1.3853e-02**
f_5	Mean	**1.5947e-07**	3.9132e-06	3.7450e-06
	SD	**6.5468e-07**	1.8086e-05	1.9565e-05
f_6	Mean	8.0449e-10	**7.7525e-11**	2.7757e-08
	SD	2.3225e-09	**1.0322e-10**	8.0469e-08
f_7	Mean	3.8381e+02	**3.3703e+02**	4.0397e+02
	SD	1.6208e+02	1.6933e+02	**1.4764e+02**
f_8	Mean	**1.5240e-08**	8.0830e-08	1.1071e-07
	SD	**2.6462e-08**	1.5611e-07	1.4769e-07
f_9	Mean	**-450**	**-450**	**-450**
	SD	1.8283e-14	**1.0556e-14**	1.8283e-14
f_{10}	Mean	**-450**	**-450**	**-450**
	SD	**6.2205e-08**	4.9119e-07	1.7472e-07
f_{11}	Mean	**-4.5000e+02**	**-4.5000e+02**	**-4.500e+02**
	SD	**4.2198e-04**	3.5347e-02	1.4650e-03
f_{12}	Mean	**3.9329e+02**	3.9606e+02	4.0136e+02
	SD	**2.9123e+00**	7.4060e+00	2.0514e+01
f_{13}	Mean	**-1.8000e+02**	**-1.7996e+02**	**-1.7996e+02**
	SD	**1.4554e-02**	1.6862e-02	1.7038e-02
f_{14}	Mean	**-140**	-1.4000e+02	**-140**
	SD	**2.9775e-07**	4.6891e-05	1.1664e-06
f_{15}	Mean	**-3.2698e+02**	-3.2648e+02	-3.2606e+02
	SD	**1.0785e+00**	1.1100e+00	1.7840e+00

Table 10: The effect of changing α with no change in p_a on the performance of the ICS (N=30)

Function		$\alpha = 0.5 - 0.01$	$\alpha = 1 - 0.001$	$\alpha = 0.5 - 0.001$
f_1	Mean	**9.5438e-21**	4.6000e-20	1.2784e-17
	SD	**1.1279e-20**	1.3378e-19	2.1359e-17
f_2	Mean	2.6678e+01	**2.5784e+01**	3.0142e+01
	SD	**1.3697e+01**	1.5047e+01	2.5385e+01
f_3	Mean	**2.2296e+01**	2.4978e+01	2.4610e+01
	SD	**4.1242e+00**	5.4647e+00	4.6284e+00
f_4	Mean	**3.1173e-09**	2.4654e-04	1.2254e-06
	SD	**1.1340e-08**	1.3503e-03	4.7742e-06
f_5	Mean	**3.0880e-01**	2.3712e+00	5.8888e-01
	SD	**5.9632e-01**	3.8305e+00	1.1604e+00
f_6	Mean	2.1058e-07	**6.8962e-08**	6.9028e-06
	SD	5.4655e-07	**1.2212e-07**	1.9754e-05
f_7	Mean	**2.5541e+03**	2.6420e+03	2.6912e+03
	SD	**4.1141e+02**	4.3081e+02	4.2520e+02
f_8	Mean	**5.2848e+00**	6.1728e+00	1.2693e+01
	SD	**3.4241e+00**	4.6207e+00	1.2218e+01
f_9	Mean	**-450**	**-450**	**-450**
	SD	**1.0449e-13**	1.1221e-13	1.6656e-13
f_{10}	Mean	-4.4034e+02	-4.4102e+02	**-4.4207e+02**
	SD	8.6941e+00	**6.8546e+00**	7.1356e+00
f_{11}	Mean	3.7542e+02	**3.6437e+02**	4.0537e+02
	SD	**3.3414e+02**	3.4552e+02	4.5036e+02
f_{12}	Mean	**4.1953e+02**	4.2239e+02	4.2657e+02
	SD	**2.1417e+01**	2.5361e+01	2.4936e+01
f_{13}	Mean	**-180**	-1.8000e+02	-1.8000e+02
	SD	**2.0596e-10**	1.8763e-03	1.3503e-03
f_{14}	Mean	-1.3978e+02	-1.3649e+02	**-1.3985e+02**
	SD	8.2844e-01	5.3672e+00	**6.8264e-01**
f_{15}	Mean	-3.0686e+02	**-3.0725e+02**	-3.0540e+02
	SD	**5.5904e+00**	6.2390e+00	6.0573e+00

Table 11: The effect of changing α with no change in p_a on the performance of the ICS (N=50)

Function		$\alpha = 0.5 - 0.01$	$\alpha = 1 - 0.001$	$\alpha = 0.5 - 0.001$
f_1	Mean	**2.0675e-19**	6.8483e-19	2.2670e-16
	SD	**2.3478e-19**	1.2882e-18	4.8857e-16
f_2	Mean	7.6080e+01	7.6643e+01	**7.5390e+01**
	SD	3.9611e+01	**3.0751e+01**	3.6539e+01
f_3	Mean	**5.3111e+01**	5.9818e+01	5.3635e+01
	SD	1.0093e+01	**1.1159e+01**	**9.6758e+00**
f_4	Mean	**1.2358e-10**	9.8614e-04	3.8313e-10
	SD	**6.7400e-10**	2.5572e-03	1.6581e-09
f_5	Mean	**8.0832e-01**	7.7256e+00	1.4874e+00
	SD	**1.2424e+00**	7.2623e+00	1.8926e+00
f_6	Mean	3.8818e-06	**1.1599e-07**	1.0947e-04
	SD	1.5914e-05	**2.1111e-07**	1.9751e-04
f_7	Mean	**4.7645e+03**	5.0340e+03	4.9477e+03
	SD	**7.6395e+02**	9.6688e+02	7.9346e+02
f_8	Mean	**1.7395e+02**	2.2547e+02	2.1874e+02
	SD	**7.4926e+01**	9.8939e+01	8.1697e+01
f_9	Mean	**-450**	**-450**	**-450**
	SD	2.2466e-13	**2.1399e-13**	2.9366e-13
f_{10}	Mean	-2.4841e+02	-2.4571e+02	**-2.0653e+02**
	SD	**7.1120e+01**	7.5262e+01	8.7765e+01
f_{11}	Mean	**4.1276e+03**	4.2418e+03	4.2070e+03
	SD	**1.1680e+03**	1.5940e+03	1.5753e+03
f_{12}	Mean	4.6508e+02	4.6803e+02	**4.6397e+02**
	SD	4.2375e+01	3.7100e+01	**3.2449e+01**
f_{13}	Mean	**-180**	-1.8000e+02	-1.8000e+02
	SD	**2.7292e-12**	3.7436e-03	3.3796e-03
f_{14}	Mean	**-1.3897e+02**	-1.3338e+02	-1.3727e+02
	SD	**2.0029e+00**	7.1231e+00	3.88013+00
f_{15}	Mean	-2.8149e+02	-2.8409e+02	**-2.8475e+02**
	SD	6.5472e+01	1.0044e+01	**9.2662e+00**