COPYRIGHT NOTICE

Federation UNIVERSITY • AUSTRALIA

FedUni ResearchOnline
http://researchonline.ballarat.edu.au

This is the submitted for peer-review version of the following article:

Long, Q., & Huang, J., (2012). A new hybrid method combining genetic algorithm and coordinate search method. *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI).* P. 1072-1077

Which has been published in final form at:
http://dx.doi.org/10.1109/ICACI.2012.6463337

A New Hybrid Method Combining Genetic Algorithm and Coordinate Search Method

Qiang Long and Junjian Huang

***Abstract*— This paper proposed a new hybrid method combining genetic algorithm(GA) and coordinate search method (CSM). Genetic algorithm is good at global exploration but bad at accuracy and local search. Whereas, coordinate search method is good at local exploitation, and its accuracy is reliable when searching in a local area. Thus we combine those two methods in this paper to design a hybrid method called *genetic algorithm with coordinate search* (GACS). Experimental tests shows that this method are good at both global search and local accuracy.**

I. Introduction

In this paper, we consider a global optimization problem

$$\begin{cases} \text{Minimize} & f(x) \\ \text{Subject to} & \\ & x \in X = [lb, ub], \end{cases} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function, and $X \in \mathbb{R}^n$ is a box set. Note that f could be a nonsmooth nonconvex function, so gradient at some point may not available or hard to obtain. And furthermore, nonconvex property of f makes global minimum hard to achieve. Therefore, globally solving problem (1) is quite a challenge problem.

In the last decades, different categories of global optimization methods have been developed to solve this problem. Among them, *stochastic algorithms* attracted a great deal of attention. Some stochastic algorithms designed from behaviors of insects or birds, such as *ant colony algorithm*[4], [5], [3] which simulates ants' strategy of searching food, *artificial bee colony algorithm*[18], [19], [17], [16] which simulate bee colony's process of searching honey and *particle swarm optimization*[20], [23], [25] which designed from flying behaviors of a group of birds. Some stochastic algorithms involve physical processes, such as *simulate annealing algorithm*[1], [21], [24] which simulate the annealing process of metal. *Evolutionary algorithm* simulate the evolutionary process of nature where the population evolute to the next generation though crossover, mutation and selection. Similarly, in evolutionary algorithms, one designs crossover operator, mutation operator and selection operator, and the population which constitutes by some randomly generated points moves to the next generation by applying those operators. A typical evolutionary algorithm is *genetic algorithm*[7], [26], [2].

Qiang Long: School of Science, Information Technology and Engineering, University of Ballarat, VIC 3350, Australia. E-mail: qianglong@students.ballarat.edu.au

Junjian Huang: Department of Computer Science, Chongqing University of Education, Chongqing 400067, China. E-mail: hmomu@sina.com

An advantage of those stochastic algorithms is their ability of global exploration, thus they are quite commonly used in global optimization. However, since they just explore the new search area in a probability point of view, their numerical performance is not very stable, they cannot guarantee that a global solution or an approximation global solution can be obtained every time. And furthermore, the accuracy of those algorithms is a big problem.

In order to overcome the disadvantage of stochastic algorithms and make good use of their global search ability, a hybrid of stochastic algorithms and local search methods could be a good idea. Some authors have made some contribution to this idea, such as Durand[6] who combined Nelder-Mead simplex method and genetic algorithm, Hedar[11] who combined simulated annealing method and direct search method and still [14] which combined tabu search method and direct search method. In this paper, we will provide a hybrid method which combines a derivative-free method, say coordinate search method, and an evolutionary algorithm, say genetic algorithm. The idea is that in the process of generating the next population, except crossover operator, mutation operator and selection operator, we add another operator called *accelerate operator* which is designed from coordinate search method. the function of accelerate operator is that, through applying coordinate search method to some randomly chosen chromosomes, we add some outstanding genes into the new generation which, in return, generate better points. In this way, we accelerate the convergence rate and improve the accuracy of genetic algorithm. The new hybrid method is called *genetic algorithm with coordinate search*(GACS).

The following contents of this paper are arranged as follows. In section 2, we propose a review of genetic algorithm and coordination search method. In section 3, we design an accelerate operator based on coordinate search method and provide the hybrid method GACS. In section 4, some numerical tests are investigated and their results are compare with some other hybrid methods. Finally, section 5 concludes this paper.

II. Preliminaries

In this section we first review the general process of genetic algorithm, and then the coordinate search method.

A. Genetic algorithm

Genetic Algorithm is one of the most important Evolutionary algorithms in mathematical programming. It was firstly introduced by John Holland in 1960s, and then developed by his students and colleagues at the University of Michigan between 1960s and 1970s [15]. In the last two decades, genetic algorithm was increasingly enriched by plenty of literatures, such as [9], [8], [10], [22]. And now various genetic algorithms are applied in different areas, such as math programming, combinational optimization, automatic control, image processing, and so on.

The main idea of genetic algorithm is based on biological natural selection and genetic mechanism. And different from traditional deterministic methods, it is a stochastic algorithm. The earliest structure of genetic algorithm was provided by Glodberg in [7]. It firstly randomly generates a series of solutions which is called *initial population*, and one individual from the population is called a *chromosome*. The number of chromosomes in a population is defined as *population size*. In numerical computation, Chromosomes are expressed as binary code, Gray code or real number code. the population generate their offspring by two different means: crossover and mutation. Crossover randomly exchanges some *genes* (which constitute chromosomes) between two selected individuals. Mutation changes some randomly selected genes of an individual in a certain way. Then the next population is constructed by selecting population size of best chromosomes from the last population and its offspring. The criterion for selecting the next generation is the performance of each chromosome according to a fitness function which is normally the objection function value. Those chromosomes whose fitness is better are kept and whose fitness is worse are eliminated. In this way, as the generation iteration goes on, the algorithm will converge to the best chromosome, which probably is the optimal solution or suboptimal solution of the original optimization problem. In practical computation, we set beforehand a *maximal generation time* which further plays a role of stopping criterion.

Suppose that $P(t)$ and $O(t)$ represent the parents and offspring of the t^{th} generation, respectively. Then the general structure of genetic algorithm can be written in the following pseudo code.

General Structure of Genetic Algorithm

1 Initialization
- 1.1 Generate the initial population $P(0)$,
- 1.2 Set crossover rate, mutation rate and maximal generation time,
- 1.3 Let $t \leftarrow 0$.

2 While the maximal generation time is not reached, do
- 2.1 Crossover operator: generate $O(t)$,
- 2.2 Mutation operator: generate $O(t)$,
- 2.3 Evaluate $P(t)$ and $O(t)$: compute the fitness function,
- 2.4 Select operator: build the next population,
- 2.5 $t \leftarrow t + 1$, go to 2.1

  end

end

From the pseudo code, we can see that there are three important operators in general genetic algorithm. Based on different encoding, those operators can be various. In this paper we use real number encoding and operators are arithmetic crossover operator, nonuniform mutation operator and best chromosomes selection operator, respectively.

B. Coordinate search method

The coordinate search method (also known as the coordinate descent method or the alternating variables method) cycles through the n coordinate directions $e_1, e_2, \ldots, e_n$, obtaining better points by performing a line search along each direction in turn. It includes an inner iteration and an outer iteration. This method is simple and somewhat intuitive, but it works quite well for some problems especially for small scale problems.

The advantage of genetic algorithm is its power of global exploration. Given the randomness of populations, the search is bestrewed all over the search space. This provides us more opportunities to obtain the global basin of a objective function, but it is the randomness which leads the terrible accuracy of genetic algorithm. Coordinate search method, on the other hand, is good at local exploitation. If the starting point of coordinate search method is in the global basin, then it can give a global solution with an excellent accuracy. Therefore, combine both the advantages of genetic algorithm and coordinate search method should be a good idea.

III. Acceleration operator and GACS method

In this section we propose a new hybrid method combining genetic algorithm and coordinate search method. By doing this, we first design the acceleration operator based on coordinate search method.

Acceleration operator

Step 1: Input the acceleration rate $acce\_rate$, the population size $popu\_size$. Set a counter $k = 1$.

Step 2: If $k > popu\_size$, stop the loop; otherwise, randomly generate a number $\beta \in [0, 1]$, if $\beta < acce\_rate$, then let $x_0 = x_k$ (which is the k^{th} chromosome in the current population) and go to step 3; otherwise, let $k = k+1$ and go back to step 2.

Step 3: Starting from x_0, Do the coordinate search and store the obtained optimal point as an offspring. Go back to step 2.

Line search is essential in coordinate search method. Normally, the optimal linear search is applied in each coordinate direction. But this may cause some problems, first of all, optimal line search needs gradient information which is not acceptable for some engineering problems, even the gradient is available, the cost for computation could be a big problem. Second of all, like steepest descent direction method, for some problem, such as quadratic problems, the so-called zigzag may happen (see Figure 1) which brings a problem for convergence.

In this paper, we apply inexact line search method, and even simpler than that, we use a double strategy for step size in line search. The idea is that we start from trying a

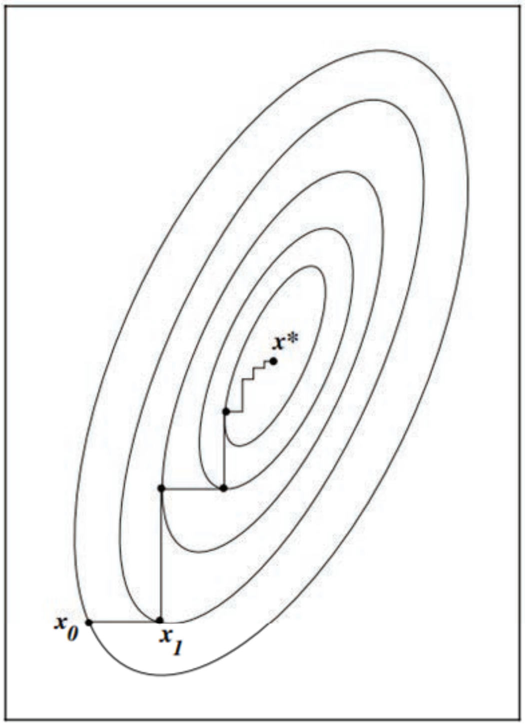

Fig. 1. Zigzag in coordinate search method

TABLE I

MAIN PROPERTIES OF TEST PROBLEMS

$Pro.$	Dim.	No.	Property	f^{**}
Ackley	10	several	Nonlinear	0
Beale	2	several	Quadratic	0
Bh1	2	1	Quadratic	0
Bh2	2	1	Quadratic	0
Bh3	2	1	Quadratic	0
Booth	2	several	Quadratic	0
Branin	2	1	Quadratic	0.397887
Colville	4	several	Quadratic	0
Dp	10	several	Quadratic	0
Easom	2	several	Exponential	-1
Gold	2	several	Quadratic	3
Griewank	10	several	Nonlinear	0
Hart3	3	4	Exponential	-3.86278
Hart6	6	6	Exponential	-3.32237
Hump	2	1	Quadratic	0
Levy	2	several	Quadratic	0
Matyas	2	1	Quadratic	0
Mich	2	several	Nolinear	-1.8013
Perm	4	several	Quadratic	0
Perm0	4	several	Quadratic	0
Powell	10	several	Quadratic	0
Powerl	4	1	Quadratic	0
Rast	10	several	Nonlinear	0
Rosen	10	several	Quadratic	0
Schw	10	several	Nonlinear	0
Shekel	4	10	Nonlinear	-10.1532
Shub	2	several	Nonlinear	-186.7309
Sphere	10	1	Quadratic	0
Sum	10	1	Quadratic	0
Trid	10	1	Quadratic	-210
Zakh	10	1	Quadratic	0

previously set step size, if it makes the objective function value decrease then we accept it and try the one which doubles it; otherwise, if it does not reduce the objective function value, then we use the last accepted step size as the result for line search.

Adding the accelerate operator to the general process of genetic algorithm, we can add some better chromosomes to the offspring, which, in return, generates more outstanding points in the next generation. And for the selection operator, we, on the one hand, try to keep those better chromosomes to be in the next generation, on the other hand, guarantee some new global exploration. So instead of choosing the population size number of best chromosomes, we build the next generation by half chosen from the best chromosomes and half chosen randomly. In the following we propose the pesudocode of genetic algorithm with coordinate search.

Genetic algorithm with coordinate search (GACS)

1 Initialization
- 1.1 Generate the initial population $P(0)$,
- 1.2 Set crossover rate, mutation rate, accelerate rate and maximal generation time,
- 1.3 Let $t \leftarrow 0$.

2 While generation counter does not reach the maximal generation number , do
- 2.1 Arithmetic crossover operator: generate $O(t)$,
- 2.2 Nonuniform mutation operator: generate $O(t)$,
- 2.3 Accelerate operator: generate $O(t)$,
- 2.4 Evaluate $P(t)$ and $O(t)$: compute their value of fitness function,
- 2.5 Selection operator: choose half population size of best chromosome from $P(t)$ and O(t), the other half is chosen randomly.
- 2.6 $t \leftarrow t + 1$, go to 2.1
  end

end

IV. NUMERICAL TESTS

In this section, we test GACS by some famous global test problems which are cited from website

http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm.

Main properties (like number of variables, global minimal value and number of local minimum) of those test problems are illustrated in table I. From table I, we can see that those test functions enjoy very different properties (quadratic ,nonliner and exponential) and some of them do have several local minimums (No.) except a global one. In the last column, global minimal value (f^{**}) of the test problems are provided.

Unfortunately, there is no mature theory of how to adjust parameters for evolutionary algorithms. So, we can just set the parameters of GACS by a great deal of experimental tests. Empirically, if the dimension of the problem is n, then the population size is $2n \sim 5n$, maximal generation number is $20n \sim 50n$. crossover rate, mutation rate and accelerate rate are $0.4 \sim 0.5$, $0.2 \sim 0.3$ and $0.1 \sim 0.2$, respectively.

In order to measure the success rate of all test algorithms,

TABLE II

COMPARISON OF ACCURACY BETWEEN GA AND GACS

$Pro.(n)$	$\bar{f}$		f^*		f^{**}
	GA	GACS	GA	GACS	
Ackley(10)	1.0882e-4	1.1738e-4	5.2899e-5	6.7253e-5	0
Beale(2)	6.2309e-9	7.1775e-8	7.3832e-10	5.7100e-13	0
Bh1(2)	6.52489e-9	2.5951e-8	7.1939e-10	1.4181e-10	0
Bh2(2)	5.6003e-9	2.8859e-8	8.1967e-10	2.0006e-11	0
Bh3(2)	3.7422e-9	6.4279e-8	4.3685e-10	5.3739e-12	0
Booth(2)	5.2070e-9	9.6581e-9	5.1714e-10	3.1795e-12	0
Branin(2)	3.9788e-1	3.9788e-1	3.9788e-1	3.9788e-1	0.397887
Colville(4)	5.6385e-5	4.2757e-5	1.0718e-9	8.5737e-8	0
Dp(10)	2.1194e-8	3.7873e-7	3.1574e-9	3.4527e-8	0
Easom(2)	–	-1.0000	–	-1.0000	-1
Gold(2)	3	3	3	3	3
Griewank(10)	–	3.0811e-3	–	7.5525e-10	0
Hart3(3)	-3.8627	-3.8627	-3.8627	-3.8627	-3.86278
Hart6(6)	-3.3223	-3.3223	-3.3223	-3.3223	-3.32237
Hump(2)	5.2402e-8	5.4034e-8	4.7194e-8	4.6514e-8	0
Levy(2)	3.8252e-9	7.5267e-10	4.8869e-10	1.3792e-13	0
Matyas(2)	4.2675e-9	9.0509e-10	5.5671e-10	1.8293e-12	0
Mich(2)	-1.8036	-1.8013	-1.8843	-1.8013	-1.8013
Perm(4)	4.1457e-3	4.6400e-3	1.7909e-8	2.7555e-4	0
Perm0(4)	3.6127e-4	3.9217e-4	2.1966e-10	2.9097e-7	0
Powell(10)	1.0520e-7	6.3157e-6	1.4659e-9	3.1229e-8	0
Powerl(4)	1.8818e-4	1.2383e-3	3.6822e-9	2.6439e-7	0
Rast(10)	8.5126e-9	1.6723e-6	5.2740e-9	8.1787e-7	0
Rosen(10)	2.7445e-8	7.2265e-6	3.0088e-9	1.2679e-6	0
Schw(10)	1.2727e-4	–	1.2727e-4	–	0
Shekel(4)	-10.5368	-10.5360	-10.5369	-10.5360	-10.1532
Shub(2)	-186.7309	-186.7309	-186.7309	-186.7309	-186.7309
Sphere(10)	1.5144e-8	8.8349e-9	3.5837e-9	3.9039e-9	0
Sum2(10)	1.5510e-8	4.8324e-8	4.5229e-9	2.0105e-8	0
Trid(10)	-210	-210	-210	-210	-210
Zakh(10)	1.6485e-8	1.0730e-6	2.3243e-9	6.5966e-8	0

we introduce the follow criteria,

$$\frac{f^* - f^{**}}{|f^{**}| + 1} < \epsilon,$$

where f^* and f^{**} stand for the obtained optimal solution and the current known best optimal solution, respectively. And ϵ is a threshold number which, in our test problems, is $10^{-2} \sim 10^{-3}$. In order to see the stability of algorithms, for each test problems, we calculate 100 times by each solver and record the time of successful calculation. And the analysis of average time of objective function evaluation, average time spent for each calculation and average optimal value are all based on the successful calculation.

All test problems are calculated in a environment of MATLAB(2010a) installed on an ACER ASPIRE4730Z laptop with a 2G RAM and a 2.16GB CPU. Before the results are showed, we illustrate some signs which are used in the following table.

- $Pro.(n)$ ... name for test problems, n is dimension of the problem;
- $\bar{f}$ ... average optimal solution over the successful calculations;
- f^* ... the best optimal solution over 100 times of execution.
- f^{**} ... the current known optimal value.

Example 4.1: **Comparison between GA and GACS**

In this example, we first compare the accuracy between GA and GACS to see if the presentation of accelerate operator improves the ability of local search. Table II shows the mean solutions and the best solutions of GA and GACS over 100 execution, from column of f^* we can see that GACS indeed improves the accuracy for most of the test problems. Second of all, Table III illustrate the success rate of GA and GACS. It can be seen that they are really neck and neck except GA failed at Easom and Griewank, as well as GACS at Schw.

TABLE III

SUCCESS RATE OF GA AND GACS

$Pro.(n)$	GA	GACS	$Pro.(n)$	GA	GACS
Ackley(10)	11	100	Matyas(2)	100	100
Beale(2)	92	83	Mich(2)	72	96
Bh1(2)	83	72	Perm(4)	85	43
Bh2(2)	80	62	Perm0(4)	75	71
Bh3(2)	87	79	Powell(10)	100	100
Booth(2)	100	100	Powerl(4)	100	89
Branin(2)	100	100	Rast(10)	4	30
Colville(4)	98	100	Rosen(10)	84	85
Dp(10)	31	88	Schw(10)	91	0
Easom(2)	0	71	Shekel(4)	36	38
Gold(2)	99	92	Shub(2)	77	93
Griewank(10)	0	8	Sphere(10)	100	100
Hart3(3)	95	89	Sum2(10)	100	100
Hart6(6)	57	67	Trid(10)	100	100
Hump(2)	100	100	Zakh(10)	100	100
Levy(2)	79	99			

Example 4.2: **Comparison between GACS and other solver**

In this example, we compare numerical performance of GACS with SAHPS[13], DTS[14] and DSSA[12]. The results are analyzed through the following four indexes: success rate, average time of objective function evaluation, average time consumption and the best solution over 100 times of executions, which are illustrated from Table IV to Table VII, respectively. It can be seen from Table IV that the success rate of GACS is neck and neck with the other three methods, except GACS failed to solve problem Schw. For the term of average objective function value evaluation time, Table V shows that GACS is better than DSSA but worse than SAHPS and DTS for most of the test problems. However, GACS spent less time on each execution than DTS and DSSA which is illustrated in Table VI. Still Table VII provides that GACS can achieve a better accuracy than other methods, such as problem Beale, Levy and Matyas.

V. CONCLUSION

In this paper, we proposed a new hybrid method combining genetic algorithm and coordinate search method which shortly named GACS. In the iteration process of general genetic algorithm, except crossover operator, mutation operator and selection operator, we add another operator called accelerate operation which is based on coordinate search method. From the numerical results, GACS performances better than GA both in terms of success rate and accuracy. And when compared with other hybrid method, GACS is still reliable and efficient.

REFERENCES

[1] E.H.L. Aarts and P.J.M. Laarhoven. Simulated annealing: an introduction. *Statistica Neerlandica*, 43(1):31–52, 1989.

TABLE IV

SUCCESS RATE OF SAHPS, DTS, DSSA AND GACS

$Pro.(n)$	SAHPS	DTS	DSSA	GACS
Ackley(10)	11	29	77	91
Beale(2)	92	87	94	76
Bh1(2)	83	82	97	23
Bh2(2)	80	85	97	31
Bh3(2)	87	92	96	40
Booth(2)	100	100	100	100
Branin(2)	100	100	100	100
Colville(4)	98	99	100	100
Dp(10)	31	42	94	80
Easom(2)	0	3	1	35
Gold(2)	99	98	97	77
Griewank(10)	0	2	43	2
Hart3(3)	95	99	95	87
Hart6(6)	57	75	94	71
Hump(2)	100	99	93	100
Levy(2)	79	100	85	100
Matyas(2)	100	100	100	100
Mich(2)	72	93	53	88
Perm(4)	85	78	93	35
Perm0(4)	75	83	100	77
Powell(10)	100	100	100	100
Powerl(4)	100	100	100	86
Rast(10)	4	0	36	4
Rast(10)	4	0	36	4
Rosen(10)	84	94	100	89
Schw(10)	91	15	72	0
Shekel(4)	36	46	66	32
Shub(2)	77	63	83	78
Sphere(10)	100	100	100	100
Sum2(10)	100	100	100	100
Trid(10)	100	100	100	100
Zakh(10)	100	100	100	100

TABLE V

AVERAGE TIME OF OBJECTIVE FUNCTION VALUE EVALUATION

$Pro.(n)$	SAHPS	DTS	DSSA	GACS
Ackley(10)	2212	12992	6125	12387
Beale(2)	243	244	336	566
Bh1(2)	224	261	283	411
Bh2(2)	229	257	279	406
Bh3(2)	227	257	274	563
Booth(2)	236	242	287	421
Branin(2)	261	242	300	344
Colville(4)	405	1079	1563	2440
Dp(10)	1002	5035	11277	6113
Easom(2)	–	181	271	997
Gold(2)	237	261	309	403
Griewank(10)	–	4706	7624	5786
Hart3(3)	376	491	635	599
Hart6(6)	798	1652	2547	1961
Hump(2)	215	244	281	368
Levy(2)	276	235	282	508
Matyas(2)	244	235	220	439
Mich(2)	311	263	304	311
Perm(4)	424	1285	3030	2128
Perm0(4)	402	1006	1546	1607
Powell(10)	1002	4503	8951	6081
Powerl(4)	413	1115	2553	1605
Rast(10)	2089	–	7510	6997
Rosen(10)	1002	6862	27287	8320
Schw(10)	1015	6355	17810	–
Shekel(4)	740	852	941	1167
Shub(2)	331	264	328	536
Sphere(10)	1003	4072	5749	4895
Sum2(10)	1002	4249	7335	4962
Trid(10)	1002	5051	12007	7745
Zakh(10)	1002	4583	16055	13844

TABLE VI

AVERAGE TIME COMSUMPTION FOR EACH EXECUTION

$Pro.(n)$	SAHPS	DTS	DSSA	GACS
Ackley(10)	0.1928	2.8324	1.1797	0.7471
Beale(2)	0.0177	0.0461	0.0289	0.0411
Bh1(2)	0.0177	0.0455	0.0299	0.0353
Bh2(2)	0.0174	0.0454	0.0291	0.0360
Bh3(2)	0.0178	0.0453	0.0290	0.0434
Booth(2)	0.0173	0.0450	0.0249	0.0367
Branin(2)	0.0190	0.0437	0.0252	0.0342
Colville(4)	0.0755	0.2076	0.2014	0.1450
Dp(10)	0.3877	1.9636	2.2302	0.4708
Easom(2)	–	0.0321	0.0258	0.0604
Gold(2)	0.0198	0.0462	0.0273	0.0377
Griewank(10)	–	1.8835	1.5041	0.4943
Hart3(3)	0.0408	0.1048	0.0749	0.0708
Hart6(6)	0.1121	0.5078	0.4280	0.2249
Hump(2)	0.0179	0.0451	0.0267	0.0372
Levy(2)	0.0241	0.0456	0.0282	0.0483
Matyas(2)	0.0180	0.0440	0.0217	0.0395
Mich(2)	0.0278	0.0496	0.0301	0.0359
Perm(4)	0.1220	0.2559	0.4084	0.1662
Perm0(4)	0.0744	0.2122	0.2243	0.1388
Powell(10)	0.2689	1.9398	1.8962	0.5960
Powerl(4)	0.1032	0.2272	0.3427	0.1345
Rast(10)	0.2210	–	1.4142	0.5395
Rosen(10)	0.8455	2.4377	6.5660	0.5808
Schw(10)	0.4820	2.2685	4.9787	–
Shekel(4)	0.0805	0.2007	0.1393	0.1261
Shub(2)	0.0277	0.0462	0.0390	0.0442
Sphere(10)	0.1483	1.7980	1.3329	0.4492
Sum2(10)	0.1874	1.8493	1.7222	0.4318
Trid(10)	0.3785	1.9795	2.5160	0.5414
Zakh(10)	0.3070	1.9101	3.4809	0.7948

TABLE VII

THE BEST SOLUTION OVER 100 EXECUTIONS

$Pro.(n)$	SAHPS	DTS	DSSA	GACS
Ackley(10)	0.0529e-3	0.1974	0.1902	0.0817
Beale(2)	0.7383e-9	0.6148	0.5537	0.0001
Bh1(2)	0.7194e-9	0.5934	0.4328	0.2422
Bh2(2)	0.0820e-8	0.1026	0.0582	0.0521
Bh3(2)	0.4369e-9	0.4821	0.3328	0.7545
Booth(2)	0.5171e-9	0.7391	0.5788	0.0216
Branin(2)	0.3979	0.3979	0.3979	0.3979
Colville(4)	0.0011e-6	0.0011	0.0003	0.3178
Dp(10)	0.0032e-6	0.0036	0.0027	0.1274
Easom(2)	–	-1.0000	-1.0000	-1.0000
Gold(2)	3	3	3	3
Griewank(10)	–	0.0000	0.0000	0.0000
Hart3(3)	-3.8628	-3.8628	-3.8628	-3.8628
Hart6(6)	-3.3224	-3.3224	-3.3224	-3.3224
Hump(2)	0.4719e-7	0.4706	0.4699	0.4651
Levy(2)	0.4887e-9	0.6093	0.3692	0.0000
Matyas(2)	0.5567e-9	0.5486	0.2966	0.0001
Mich(2)	-1.8844	-1.8013	-1.9962	-1.8013
Perm(4)	0.0000e-3	0.0000	0.0000	0.2278
Perm0(4)	0.0000e-5	0.0001	0.0001	0.1323
Powell(10)	0.0147e-7	0.0337	0.0101	0.7074
Powerl(4)	0.0000e-4	0.0000	0.0000	0.1297
Rast(10)	0.0000	–	0.0000	0.0000
Rosen(10)	0.0003e-5	0.0001	0.0002	0.1762
Schw(10)	-0.0661e5	-0.0063	-1.0008	–
Shekel(4)	-10.5364	-10.5364	-10.5364	-10.5364
Shub(2)	-186.7309	-186.7309	-186.7309	-186.7309
Sphere(10)	0.3584e-8	0.3562	0.2531	0.3631
Sum2(10)	0.0452e-7	0.0259	0.0247	0.1857
Trid(10)	-210	-210	-210	-210
Zakh(10)	0.0232e-7	0.0359	0.0231	0.8621

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
[3] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006.
[4] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2):243–278, 2005.
[5] M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997.
[6] N. Durand and J.M. Alliot. A combined nelder-mead simplex and genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, volume 99, pages 1–7, 1999.
[7] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley, 1989.
[8] D.E. Goldberg. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, 4(4):445–460, 1990.
[9] D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(5):493–530, 1989.
[10] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128, 1986.
[11] A.R. Hedar and M. Fukushima. Hybrid simulated annealing and direct search method for nonlinear global optimization. *Department of Applied Mathematics & Physics Kyoto University*, pages 2001–013, 2001.
[12] A.R. Hedar and M. Fukushima. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *Optimization Methods and Software*, 17(5):891–912, 2002.
[13] A.R. Hedar and M. Fukushima. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optimization Methods and Software*, 19(3-4):291–308, 2004.
[14] A.R. Hedar and M. Fukushima. Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170(2):329–349, 2006.
[15] J.H. Holland. *Adaptation in natural and artificial systems*. Number 53. University of Michigan press, 1975.
[16] D. Karaboga. Artificial bee colony algorithm. *Scholarpedia*, 5(3):6915, 2010.
[17] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009.
[18] D. Karaboga and B. Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. *Foundations of Fuzzy Logic and Soft Computing*, pages 789–798, 2007.
[19] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.
[20] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.
[21] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
[22] H. Kitano. Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms. *Physica D: Nonlinear Phenomena*, 75(1-3):225–238, 1994.
[23] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.
[24] R.A. Rutenbar. Simulated annealing algorithms: An overview. *Circuits and Devices Magazine, IEEE*, 5(1):19–26, 1989.
[25] Y. Shi. Particle swarm optimization. *IEEE Connections*, 2(1):8–13, 2004.
[26] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *Intelligent Systems and Their Applications, IEEE*, 13(2):44–49, 1998.