DEAL: A Direction-Guided Evolutionary Algorithm

Cuong C. Vu[1], Lam Thu Bui[1], and Hussein A. Abbass[2]

[1] Le Quy Don Technical University, Vietnam
{cuongvcc,lam.bui07}@gmail.com
[2] University of New South Wales, Australia
h.abbass@adfa.edu.au

Abstract. In this paper, we propose a real-valued evolutionary algorithm being guided by *directional information*. We derive direction of improvement from a set of elite solutions, which is always maintained overtime. A population of solutions is evolved over time under the guidance of those directions. At each iteration, there are two types of directions that are being generated: (1) *convergence direction* between an elite solution (stored in an external set) and a second-ranked solution from the current population, and (2) *spreading direction* between two elite solutions in the external set. These directions are then used to perturb the current population to get an offspring population. The combination of the offsprings and the elite solutions is used to generate a new set of elite solutions as well as a new population. A case study has been carried out on a set of difficult problems investigating the performance and behaviour of our newly proposed algorithm. We also validated its performance with 12 other well-known algorithms in the field. The proposed algorithm showed a good performance in comparison with these algorithms.

Keywords: direction of improvement, evolutionary algorithms.

1 Introduction

Evolutionary algorithms (EAs) have been popular tools for approximating solutions of optimization problems. For real-parameter EAs such as real-parameter GA, differential evolution (DE), evolutionary strategies (ES) and evolutionary programming (EP), a real-valued representation of genes is used. In the literature of evolutionary computation, the use of elitism has been the most popular one. For it, usually a number of good solutions, *the elite set*, is (either implicitly or explicitly) maintained over time. Our motivation is that this set can contribute information to the evolutionary process much more than just storing some solutions to the next generation.

Our proposal is as follows: we can derive from the elite set some directions of improvement and use these directions to guide the evolutionary process. Note that when designing an optimization algorithm, it is desirable to find a good

L.T. Bui et al. (Eds.): SEAL 2012, LNCS 7673, pp. 148–157, 2012.

direction that guides the search process; the *steepest gradient descent* method is the most typical example of getting advantage from a good direction to guide the search. However, in general, defining a good direction is not a trivial task, especially in the case of non-linear or black-box functions. For evolutionary computation, the use of directions has been shown quite promising. An example is the case of Differential Evolution (DE), which uses the direction between two randomly-selected parents to guide the newly-generated offsprings [17]; DE has been very effective in solving continuous optimization problems.

There are several ways to maintain the elite set (in short, ETS). In this proposal, we select a number of best solutions for ETS. Our main design will focus on how to use and refine ETS. Its size is set as half of the main population. ETS is used not only to contribute solutions to the next generation, but also to generate directions for offspring production. At each generation, a pool of offsprings (having the same size as the main population) is produced. This is done via a perturbation process in which randomly-selected parents (from the main population) are perturbed by directions of improvement. There are two types of directions presented in this work: (1) convergence direction that is defined as the direction between an elite solution from ETS and a second-ranked solution from the main population and (2) spreading direction between two elite solutions in ETS. This pool of offsprings is combined with ETS in order to generate the new content of ETS and the next generation. The combined population is then sorted and copied to ETS as well as the main population. To validate the newly proposed algorithm, we carried out a case study on 6 benchmark problems with high modality. The results from these problems showed that our algorithm performed quite well. Further, we also obtained the results from 12 other well-known algorithms. The results indicated that our algorithm was very competitive with these algorithms.

The paper is organized in seven sections. Section 2 is dedicated to background literature and followed by the methodology section. A number of test problems are presented, solved and studied in Section 4 to demonstrate the concept, and the paper concludes in Section 5.

2 Background

Evolutionary Algorithms (EAs) have been well recognized as a major class of heuristic techniques in computational optimization and machine learning. They have been applied widely in many aspects of human life from social and engineering problems to security and military domains. In principle, an EA is a process that imitates evolution in Nature. It works with a population of solutions in searching for the problem's optima where the population undergoes an evolutionary process of many cycles using nature-mimicking crossover, mutation and/or selection operators. After each cycle, a new population is formed and is called a *generation*. With this population-based computational approach, EAs offer a potential paradigm for solving global optimization problems [9,1,15,16,10]. Originally, there were four classes of EAs; including Genetic Algorithms (GAs),

Evolutionary Strategies (ES), Evolutionary Programming (EP), and Genetic Programming (GP). To date, there are several paradigms that have emerged as alternatives for the conventional EAs, such as Particle Swarm Optimization (PSO) [12], Ant Colony Optimization (ACO) [7], Differential Evolution (DE) [17], Estimation of Distribution Algorithms (EDA) [14], and Artificial Immune Systems (AIS) [3]. For them, mutation and crossover operators might be replaced by some specific operator inspired by a different phenomenon in nature.

Real-parameter evolutionary algorithms can be classified into different streams. In general, the difference between these streams is in the way to employ and implement the evolutionary operators. The first stream is the real parameter GA that has the same framework as the binary-coded GA with a focus on crossover. As an example, Deb et al [5] introduced a version of the real parameter GA using the SBX crossover operator that simulates the binary crossover operator. A reasonable overview of real parameter GA can be found in [4]. Meanwhile, ES [18] and EP [8,20] concentrate more on the mutation operator. The child is generated by disturbing a selected solution with Gaussian or Cauchy distributed random deviations. After this phase, all solutions have to undergo a selection process.

In the case of simple DE, it uses one main parent and two supportive parents [19,2] for generating a child. Basically, the main parent is disturbed by adding a step length multiplied by the difference between the two supportive parents. The resultant solution is called the trial/protoype solution. The prototype is then crossed-over with another pre-selected solution to generate a child. Elitism is implemented implicitly in the way that the child is inserted into the population if it outperforms the pre-selected solution. By using difference vectors, DE takes into account direction information. In some cases, good directions will be generated and DE will generate good solutions. In other cases, bad directions will be generated which will deteriorate the solution quality. This poses a question on whether or not we can systematically maintain good directions?

Several other real-valued versions can be listed here such as covariance matrix adaptation evolution strategy (CMAES) [11] being implemented with an adaptive covariance matrix to model a second-order approximation of the objective function, and generalized generation gap model with generic parent-centric recombination operator (G3PCX) [6] using elite-preservation and the parent-centric recombination operator. Recently, real-coded version of chemical reaction optimization emerges as a new paradigm for real-valued optimization [13].

3 Methodology

3.1 Overview

It has been demonstrated that elitism is useful for an EA. However, the issue is how to use it effectively? Therefore, we will focus our work on this issue when designing a new EA; especially we will address: (1) *Interaction between an ETS and the main population* and (2) *updating the ETS*.

Our methodology proposes to maintain an ETS during the optimization process. This ETS will contribute to the evolutionary process not only the elitist

solutions, but also the directional information (which we call as direction of improvement). A direction of improvement is considered as a useful piece of information during optimization process. Our proposal is that at every generation, the main population will be perturbed by these directions in order to produce offsprings. Theses offsprings are then combined with the current ETS to form a temporary population, called "*the combined population*". This combined population is used subsequently to fill in the new version of ETS. Based on this merit, we call our algorithm as Direction-guided Evolutionary ALgorithm or DEAL.

3.2 Directional Information

We propose to use two types of directional information: convergence and spreading.

- *Convergence direction*: It is defined as the direction from a solution to a better one. We consider it as the direction between second-ranked solution and an elite one. If elite solutions are maintained globally, it is considered as the global direction of convergence. If a solution is guided following this direction, it will find a better area.
- *Spreading direction*: It is defined as the direction between two peers. In this context, it is the direction between two elite solutions. If solutions are perturbed along these directions, a better spreading within the population will be obtained.

3.3 General Structure

A step-wise structure of the proposed algorithm is given as follows:

- **Step 1:** Initialize the main population P with size N
- **Step 2:** Evaluate the population P
- **Step 3:** Copy elite solutions to ETS (that has the half size of population P)
- **Step 4:** Report the elite solutions in ETS
- **Step 5:** Generate a mixed population M with size of N, and set $index = 0$
- Loop {
  - Copy $P(index)$ to $M(index)$
  - Select a random parent P_r
  - Generate a convergence direction d_1 from a randomly-selected low-rank solution in the population P to a randomly-selected solution from ETS.
  - Generate a spreading direction d_2 between two randomly selected solutions in ETS.
  - Generate two offspring solutions S_1 and S_2 by perturbing the parent solution using two newly generated directions.
    - For each dimension i
      - If $U(0,1) < pc$ then $S_1(i) = P_r(i) + \sigma_1 * d_1(i)$
      - Else $S_1(i) = P_r(i)$
      - If $U(0,1) < pc$ then $S_2(i) = P_r(i) + \sigma_2 * d_2(i)$

· Else $S_2(i) = P_r(i)$
* End for
Where $U(0,1)$ is the random function returning values between 0 and 1, pc is crossover rate, $\sigma_1 = U(0,1)$; σ_2= a small constant (i.e 0.5).
- Evaluate S_1
- S_1 is better than $M(index)$ then replace $M(index)$ by S_1
- Mutate S_2 with a predefined rate pm
- Evaluate S_2
- S_2 is better than $M(index+1)$ then replace $M(index+1)$ by S_2

$index = index + 2$
– } Until (the mixed population is full)
– **Step 6:** Combine the mixed population M with ETS to form a combined population C (or M+A $\rightarrow$ C)
– **Step 7:** Sort C using fitness values
– **Step 8:** Determine the new members of ETS by copying first $N/2$ elite solutions from the combined population C)
– **Step 9:** Determine the new population P by copying solutions from M)
– **Step 10**: Go to Step 4 if stopping criteria is not satisfied

Note that ETS can be maintained implicitly (without an explicit data structure). The current main population can be classified into two parts: the first half is the elite solutions copied from the combined population C and this part is actually ETS and the second half is the lower ranked (or second-ranked) solutions. σ is the step length for perturbation. For σ_1, our finding is that the best strategy is $\sigma_1 = U(0,1)$ and σ_2 is 0.5 (that basically reduces half of the vector's magnitude). The step 5 is the main element in this structure. It shows that for the offsprings, half of them are created for convergence purpose (exploitation) while the other half to make it more diverse (exploration).

The main computational cost comes from the task of filling ETS. Filling solutions requires sorting the combined population C. In general, a sorted procedure requires complexity of $O(NlogN)$. So, the overall complexity of the algorithm is $O(NlogN)$.

4 A Case Study

4.1 Testing Problems

We considered to test a set of 6 popular continuous test problems with high-dimensionality and high modality [13]. The only reason for us to select these problems is that these problems illustrate the highest difficulty facing optimization algorithms: multi-modality. They are reported in Table 1.

4.2 Experimental Setup

We selected other well-known algorithms for validating ours with settings used by authors in [13]: Real-coded version of Chemical Reaction Optimization (RCCRO), Genetic Algorithm (GA), Fast evolutionary programming(FEP), Classical evolutionary programming (CEP), Fast evolutionary strategy (FES), Conventional evolutionary strategy (CES), Particle Swarm Optimization (PSO),

Table 1. Lists of test problems used for experiments in this paper

ID	n	Description	Name	Range	f_{min}
F1	30	$f(x) = -\sum_{i=1}^{n}(x_i sin(\sqrt{\lvert x_i \rvert}))$	Generalized Schwefel's problem	$[-500, 500]$	-12569.5
F2	30	$f(x) = \sum_{i=1}^{n}(x_i^2 - 10cos(2PIx_i) + 10)$	Generalized Rastrigin's problem	$[-5.12, 5.12]$	0
F3	30	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^2)})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}(cos(2PIx_i)) + 20 + e$	Ackley's problem	$[-32, 32]$	0
F4	30	$f(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i^2) - \prod_i^n cos(\frac{x_i}{\sqrt{i}}) + 1$	Generalized Griewank's problem	$[-600, 600]$	0
F5	30	$f(x) = \frac{PI}{n}\{10sin^2(PIy_1) + \sum_{i=1}^{n-1}(y_i - 1)^2$ $[1 + 10sin^2(PIy_{i+1})] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & \text{otherwise.} \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	Generalized Penalized problem	$[-50, 50]$	0
F6	30	$f(x) = 0.1\{sin^2(3PIx_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 +$ $sin^2(3PIx_{i+1})] + (x_n - 1)^2[1 + sin^2(2PIx_n)]\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Generalized Penalized problem	$[-50, 50]$	0

Group search optimizer (GSO), Real-coded biogeography-based optimization (RCBBO), Differential evolution (DE), Covariance matrix adaptation evolution strategy (CMAES), and Generalized generation gap model with generic parent-centric recombination operator (G3PCX).

The experiments for our algorithm are carried out on all 6 test problems and with the following parameters: The population size was also 100 solutions, the number of evaluations are 150000, 250000, 150000, 150000, 150000, and 150000 for all problems respectively, the mutation rate was kept at the same small rate of 0.01, and the crossover rate was 0.9. Further, there were 100 runs with different random seeds for testing each problem.

4.3 Results and Discussion

Behavior Analysis: To analyze the behavior of DEAL, we first tested it on a spherical problem, the easiest problem: $f(x) = \sum_{i=1}^{n} x_i^2$, with $n = 30, x \in [-100, 100]$ (the optimal point is at the origin and we call it as the zero point). We recorded the objective value of the best solution found over time. After 150000 evaluations, DEAL obtained a near-zero average objective value (in different 100 runs) of 1.558E-12 (standard deviation is 6.158E-12).

From Figure 1, it is obvious that DEAL stably converged towards the optimal solution among all 100 runs. After 250 generations, solutions found by DEAL's 100 runs were almost close to the optimal point (the left graph). The right graph is magnified by logarithmic transformation and shows the constant convergence towards the zero point. In the decision space, all the points have x_i values being around the zero point with radius of 10e-8.

Another look at can be seen at Figure 2 where we displayed the behavior of DEAL on a multi-modal problem: the Akley problem (F3). The difficulty

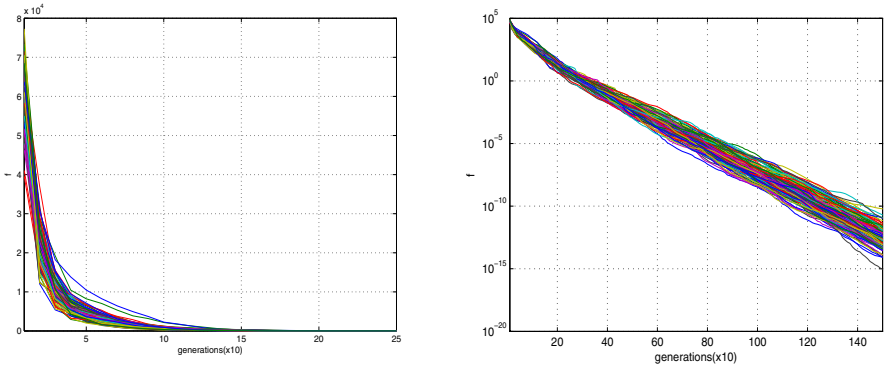

Fig. 1. Visualization of the best solution for the spherical problem found by DEAL in all 100 runs. Left graph: the convergence curve during the first 250 generations. Right graph: the convergence curve with log-transformation during all 1500 generations.

of multi-modality clearly made DEAL longer to converge. In contrast the case of the spherical problem, at generation 25rd (where DEAL converged for the spherical problem), DEAL stilled far from the optimal point. In all 100 runs, it converged almost at generation 500rd. After that the optimization still refining its best solution until the end (see the right logarithmic-transformed graph)

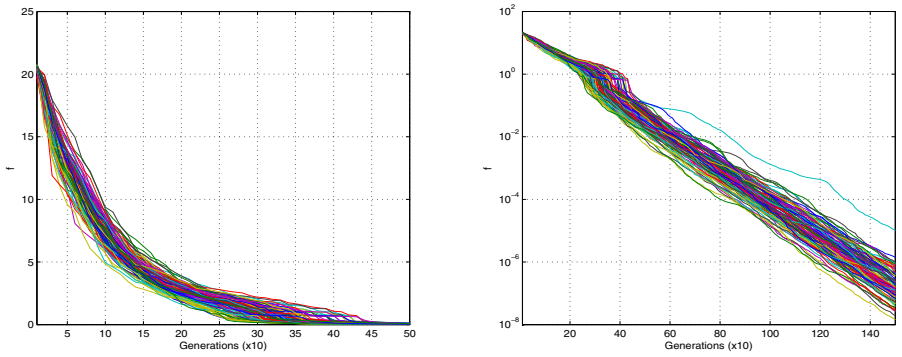

Fig. 2. Visualization of the best solution for Akley problem found by DEAL in all 100 runs. Left graph: the convergence curve during the first 500 generations. Right graph: the convergence curve with log-transformation during all 1500 generations.

Comparison with Others: In comparison with other approaches, we recorded the best objective value obtained by approaches for all problems in Table 1 and then calculated the mean and standard deviation. They are reported in Table 2 together with results obtained from [13]. From the table we can see that there is no clear winer among all 13 approaches. However, it indicates that 8 approaches (GA, FEP, CEP, FES, CES, PSO, RCBBO, and G3PCX)were inferior in all 6

test problems. The remaining 5 algorithms are seemed better in which each had at leat the result on one problem with 1st rank. Within this set of approaches, DEAL and DE emerged with more competitive results. While DEAL has two problems ranked No1 (the most case), DE has 1 problem ranked 1st and 3 problems ranked 2rd. The interesting note here is that both DEAL and DE used direction explicitly: while DE used direction between two randomly-selected parents, DEAL used direction of improvement.

Table 2. Obtained results for all test problems (Mean, Standard deviation and rank)

		F1	F2	F3	F4	F5	F6
DEAL	Mean	-1.085E+04	3.482E-15	3.097E-07	1.305E-02	3.619E-07	1.099E-04
	Std	3.996E+02	5.888E-15	8.720E-07	1.721E-02	1.268E-06	1.099E-03
	Rank	9	1	1	5	3	5
RCCRO1	Mean	-1.257E+04	9.077E-04	1.944E-03	1.117E-02	2.074E-02	7.048E-07
	Std	2.317E-02	2.876E-04	4.190E-04	1.622E-02	5.485E-02	5.901E-07
	Rank	2	3	5	3	7	1
GA	Mean	-1.257E+04	6.509E-01	8.678E-01	1.004E+00	4.372E-02	1.681E-01
	Std	2.109E+00	3.594E-01	2.805E-01	6.755E-02	5.058E-02	7.068E-02
	Rank	2	7	9	3	10	10
FEP	Mean	-1.255E+04	4.600E-02	1.800E-02	1.600E-02	9.200E-06	1.600E-04
	Std	5.260E+01	1.200E-02	2.100E-02	2.200E-02	6.140E-05	7.300E-05
	Rank	8	5	7	6	4	6
CEP	Mean	-7.917E+03	8.900E+01	9.200E+00	8.600E-02	1.760E+00	1.400E+00
	Std	6.345E+02	2.310E+01	2.800E+00	1.200E-01	2.400E+00	3.700E+00
	Rank	11	12	12	9	12	12
FES	Mean	-1.256E+04	1.600E-01	1.200E-02	3.700E-02	2.800E-02	4.700E-05
	Std	3.253E+01	3.300E-01	1.800E-03	5.000E-02	8.100E-11	1.500E-05
	Rank	7	6	6	8	8	4
CES	Mean	-7.550E+03	7.082E+01	9.070E+00	3.800E-01	1.180E+00	1.390E+00
	Std	6.314E+02	2.149E+01	2.840E+00	7.700E-01	1.870E+00	3.330E+00
	Rank	12	11	11	11	11	11
PSO	Mean	-9.660E+03	2.079E+01	1.340E-03	2.323E-01	3.950E-02	5.052E-02
	Std	4.638E+02	5.940E+00	4.239E-02	4.434E-01	9.142E-02	5.691E-01
	Rank	10	9	4	10	9	9
GSO	Mean	-1.257E+04	1.018E+00	2.655E-05	3.079E-02	2.765E-11	4.695E-05
	Std	2.214E-02	9.509E-01	3.082E-05	3.087E-02	9.167E-11	7.001E-04
	Rank	2	8	2	7	1	3
RCBBO	Mean	-1.257E+04	2.620E-02	2.510E-02	4.820E-01	3.280E-05	3.720E-04
	Std	2.200E-05	9.760E-03	5.510E-03	8.490E-02	3.330E-05	4.630E-04
	Rank	2	4	8	12	5	7
DE	Mean	-1.257E+04	7.261E-05	7.136E-04	9.054E-05	1.886E-07	9.519E-07
	Std	2.333E-05	3.376E-05	6.194E-05	3.402E-05	4.266E-08	2.021E-07
	Rank	2	2	3	1	2	2
CMAES	Mean	-9.873E+07	4.950E+01	4.607E+00	7.395E-04	5.167E-03	1.639E-03
	Std	8.547E+08	1.229E+01	8.725E+00	2.389E-03	7.338E-03	4.196E-03
	Rank	1	10	10	2	6	8
G3PCX	Mean	-2.577E+03	1.740E+02	1.352E+01	1.127E-02	4.593E+00	2.349E+01
	Std	4.126E+02	3.199E+01	4.815E+00	1.310E-02	5.984E+00	2.072E+01
	Rank	13	13	13	4	13	13

Effect of the Step length: In this section, we will discuss the effect of the step length σ on the performance of our proposed approach. We call the above version of DEAL is as Option 1 where $\sigma_1 = U(0,1)$ and $\sigma_2 = 0.5$. We tested 3 other options as follows:

- Option 2: $\sigma_1 = 1$ and $\sigma_2 = 0.5$
- Option 3: $\sigma_1 = U(0,1)$ and $\sigma_2 = U(0,0.5)$
- Option 4: $\sigma_1 = 1$ and $\sigma_2 = U(0,0.5)$

Table 3. Obtained results for all test problems from all options of DEAL

		F1	F2	F3	F4	F5	F6
Option1	Mean	-1.085E+04	3.482E-15	3.097E-07	1.305E-02	3.619E-07	1.099E-04
	Std	3.996E+02	5.888E-15	8.720E-07	1.721E-02	1.268E-06	1.099E-03
Option2	Mean	-1.064E+04	8.419E-05	3.977E-05	1.546E-02	1.063E-03	6.594E-04
	Std	3.479E+02	5.599E-04	3.433E-04	1.737E-02	1.037E-02	2.622E-03
Option3	Mean	-1.097E+04	1.785E-14	4.213E-06	1.350E-02	6.433E-06	8.790E-04
	Std	3.028E+02	1.594E-14	4.641E-06	1.423E-02	3.494E-05	2.996E-03
Option4	Mean	-1.064E+04	8.419E-05	3.977E-05	1.546E-02	1.063E-03	6.594E-04
	Std	3.479E+02	5.599E-04	3.433E-04	1.737E-02	1.037E-02	2.622E-03

We can observe from Table 3 that there were no large change in the results obtained by all options, except the slightly better results of Option 1. This indicates that the use of either random or fixed value of σ does not significantly effect on the performance of DEAL.

5 Conclusion

In this paper, we introduced a novel technique for employing directions of improvement for EAs; we call it *a direction-guided evolutionary algorithm*. With this new algorithm, a population of solutions is evolved over time under guidance of directions of improvement. At each generation, there are two types of directions are generated: (1) convergence direction between an elite solution and a solution from the current population, and (2) spreading direction between two elite solutions in ETS. These directions are then used to perturb the current population to get a temporary population of offsprings. The combination (combined population) of this offspring population and the current ETS is used to generate the next content of ETS and the main population.

A case study has been carried out to investigate the performance and behaviour of our newly proposed algorithm. We also validated its performance with 12 other well-known algorithms in the field. Our algorithms showed a good performance in comparison with these algorithms.

Acknowledgement. The authors gratefully acknowledge the financial support from the Vietnam Institute for Advanced Study in Mathematics (VIASM) and the University of New South Wales at Australian Defence Force Academy.

References

1. Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
2. Corne, D., Dorigo, M., Glover, F.: New Ideas in Optimization. McGraw Hill, Cambridge (1999)
3. Dasgupta, D.: Artificial Immune Systems and Their Applications. Springer, Berlin (1998)
4. Deb, K.: Multiobjective Optimization using Evolutionary Algorithms. John Wiley and Son Ltd., New York (2001)

5. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
6. Deb, K., Anand, A., Joshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evolutionary Computation 4, 371–395 (2002)
7. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, USA (2004)
8. Fogel, L.J., Angeline, P.J., Fogel, D.B.: An evolutionary programming approach to self-adaptation in finite state machines. In: McDonnell, J.R., Reynolds, R.G., Fogel, D.B. (eds.) Proc. of Fourth Annual Conference on Evolutionary Programming, pp. 355–365. MIT Press, Cambridge (1995)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
10. Goldberg, D.E.: The design of innovation: lessons from and for competent genetic algorithms. Kluwer Academic Publishers, Massachusetts (2002)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9, 159–195 (2001)
12. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
13. Albert, Y.S., Lam, V.O.K.: Li, and James J.Q. Yu. Real-coded chemical reaction optimization. IEEE Transactions on Evolutionary Computation (accepted for publication, 2012)
14. Larraanaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Norwell (2002)
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, London (1996)
16. Mitchell, T.: Machine Learning. McGraw Hill, Singapore (1997)
17. Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin (2005)
18. Rudolph, G.: Evolution strategy. In: Handbook of Evolutionary Computation. Oxford University Press (1997)
19. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report tr-95-012. Technical report, ICSI (1995)
20. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2), 82–102 (1999)