

Extending a class of continuous estimation of distribution algorithms to dynamic problems

Bo Yuan · Maria Orlowska · Shazia Sadiq

Received: 28 April 2007 / Accepted: 13 November 2007
© Springer-Verlag 2007

Abstract In this paper, a class of continuous Estimation of Distribution Algorithms (EDAs) based on Gaussian models is analyzed to investigate their potential for solving dynamic optimization problems where the global optima may change dramatically during time. Experimental results on a number of dynamic problems show that the proposed strategy for dynamic optimization can significantly improve the performance of the original EDAs and the optimal solutions can be consistently located.

Keywords Evolutionary algorithms · Global optimization · Estimation of distribution algorithms · Dynamic optimization

1 Introduction

Estimation of Distribution Algorithms (EDAs) [7] refer to a class of Evolutionary Algorithms (EAs) [1] based on statistical modeling of the search space instead of traditional genetic operators such as crossover and mutation. In addition to sharing the robustness and global optimization ability of EAs, the unique feature of EDAs is that they are capable of building a principled statistical model of the distribution of

B. Yuan (✉)
Division of Informatics, Graduate School at Shenzhen, Tsinghua University,
Shenzhen 518055, People's Republic of China
e-mail: yuanb@sz.tsinghua.edu.cn

M. Orlowska · S. Sadiq
School of Information Technology and Electrical Engineering, The University of Queensland,
Brisbane, QLD 4072, Australia
e-mail: maria@itee.uq.edu.au

S. Sadiq
e-mail: shazia@itee.uq.edu.au

promising individuals and using this model to conduct highly efficient searching [9]. Like many other EAs, EDAs are often initially proposed with stationary optimization problems as the target. However, in the real-world, it is well known that many problems do come with certain level of non-stationary properties.

For example, in our previous work, a robot routing problem is proposed where the objective is to find the shortest path along which a mobile robot can download the data from all sensors in a wireless sensor network [19,20]. In order to communicate with each sensor, the robot must be physically within its effective range, which is specified by a disk. The size of the disk is determined by the battery level of the sensor and may change over time due to battery decaying or recharging, which requires the optimization techniques to be able to consistently locate the new global optima.

Motivated by the optimization problem in the above scenario, in this paper, we focus on investigating the possibility of adapting a class of continuous EDAs to the dynamic environments. One of the major characters of this class of EDAs is that they all employ a single Gaussian model as the statistical representation of promising individuals. The only difference is the complexity of the Gaussian model adopted (e.g., diagonal covariance matrix vs. full covariance matrix). Experimental research has shown that they are particularly suitable for optimization problems with the “big-valley” structure (the robot routing problem also has this type of structure). To the best of our knowledge, we are not aware of any existing work on extending this class of EDAs to the domain of dynamic optimization.

In Sect. 2, the framework of EDAs is presented to give an overview of their working mechanism. A brief review of some of the representative strategies for dynamic optimization is given in Sect. 3. The details of the proposed scheme for adapting the EDAs are shown in Sect. 4. Section 5 contains the experimental results on a number of dynamic optimization problems to empirically justify the performance of the dynamic EDAs. This paper is concluded in Sect. 6 with some discussions for future work.

2 Estimation of distribution algorithms

As shown in Table 1, similar to traditional EAs, the general framework of EDAs is also an iterative procedure. In each generation, a subset of the population is selected (Step 4), which typically consists of the best individuals in the current population. The most important part is in Step 5 where a statistical model $P(X)$ is built to

Table 1 The general framework of EDAs

Step 1: Initialize the probability model $P_0(X)$, $t = 0$
Step 2: Sample a population $O = \{X_1, \dots, X_n\}$ from $P_t(X)$
Step 3: Evaluate individuals in $O : F = \{f(X_1), \dots, f(X_n)\}$
Step 4: Select a subset of the population $O' \subset O$
Step 5: Update the model: $P_t(X) \rightarrow P_{t+1}(X)$
Step 6: $t = t + 1$
Step 7: Go to Step 2 until stopping criteria are met

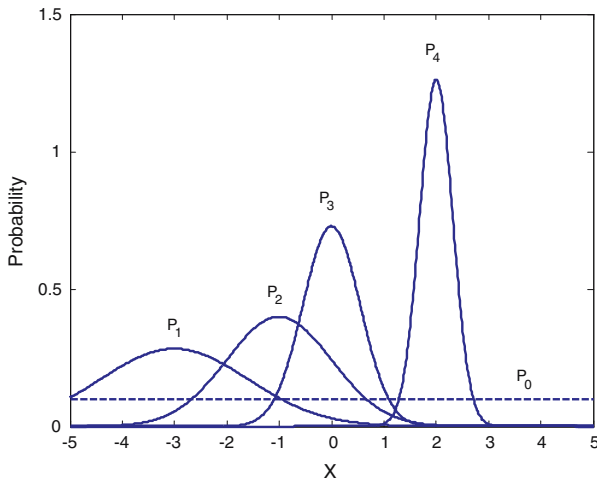


Fig. 1 An illustration of the evolution process of the Gaussian model during generations

approximate the probability distribution of the selected individuals (X is a vector of variables/parameters to be optimized). The population in the next iteration is then generated by sampling from the statistical model (Step 2).

In the binary domain, dependence chain models [3], dependence tree models [2] and Bayesian networks [10] are some of the typical statistical models used in the literature. For continuous optimization problems, Gaussian models are most widely employed mainly due to its computational efficiency [6,8]. To help understand the evolution process of EDAs with Gaussian models, Fig. 1 gives an illustration of the Gaussian model on a hypothesized 1D problem. Initially, the population is generated from a most general model (e.g., a uniform distribution). In subsequent generations, the parameters of the Gaussian model (mean and variance) are updated in a way to better represent the set of current promising individuals. The most basic version of Gaussian EDAs only uses a diagonal covariance matrix, which captures no dependences among variables. Alternatively, the dependence relationship can be partially or fully described by Gaussian networks.

In this paper, the focus is on the class of continuous EDAs based on a single Gaussian model. Since the capability of capture different orders of dependences is not directly related to the issue of dynamic optimization, for the sack of simplicity, only the basic version of Gaussian EDAs (referred to as EDA_G) is considered from now on and the techniques developed for dynamic optimization are equally applicable to other specific versions of Gaussian EDAs.

3 An overview of dynamic optimization

Recently, the area of dynamic optimization using EAs has attracted broad interest from the community. A number of strategies for adapting various traditional EAs to non-stationary environments have also been proposed. Although some successes have

been achieved, the research in this area is still far from mature and the results are often mixed, depending on the specific properties of the algorithms and the problems involved. Please refer to [14] for a comprehensive review of the related topics.

Generally speaking, there are two categories of strategies for dynamic optimization. The first one is based on the idea of maintaining a memory of good individuals that the algorithm has come across during the search [13]. Once the problem is changed, this memory will be merged with the main population. It is expected that transferring some good solutions of the previous problems to the current population may help the algorithm cope with the new problem.

There are a few issues that need to be considered in this strategy. Firstly, the memory needs to be reevaluated in every generation, which is inefficient in terms of computational cost. Secondly, additional heuristics need to be carefully designed to determine, for example, when and how to update this memory. Finally, this strategy requires the timely detection of the environmental changes. Although a change of environment can be detected whenever the fitness values of some individuals in the memory differ from their values in the last generation, this is only the sufficient condition instead of the necessary condition. In fact, it is possible for a problem to be updated in a way that all individuals in the memory will not be affected.

The second category of strategies is based on the idea of random immigrations, which means new individuals are introduced into the main population from time to time, mainly for the purpose of diversity maintenance. For example, it can be implemented by adding randomly generated individuals into the current population to maintain the effective search area of the algorithm. Obviously, there is a balance (tradeoff) between convergence speed and population diversity (dynamic optimization ability) that needs to be well managed.

4 EDA_G for dynamic optimization

The strategy proposed in this paper belongs to the second category, which is relatively simple compared to the first one. Also, it does not require the detection of environmental changes. EDA_G is different from many other EAs in that it often has faster convergence speed, which is potentially an advantage for quickly locating the global optimum. However, the downside is that once they converge to a certain area, it is very likely that they will get stuck there, losing the capability of exploring other areas of the search space [16–18].

One of the solutions to maintaining the diversity in EDA_G is to manipulate its Gaussian model to explicitly control the convergence speed [18]. However, if the convergence speed is too slow, EDA_G may not even be able to find the current global optimum before the problem is changed. On the flip side, once EDA_G converges to the global optimum, it will again get stuck there and have little ability of exploration. As a result, this method is not expected to be particularly helpful in practice.

A simple yet effective solution is to add a secondary model into EDA_G, which is responsible for generating individuals in a very general form (e.g., individuals following the random distribution or a Gaussian distribution with large variances). The

population thus contains individuals generated from two models. Note that the primary model is evolved in the usual way (i.e., updated by selected individuals from the population) while the secondary model is always fixed. Although this kind of strategy has been used in other EDAs before [15] and achieved certain level of success, it has some inherent issues as far as EDA_G is concerned.

Suppose that the primary model of EDA_G has converged to the global optimum $X^*(t)$ of the t th status of the problem. Next, the problem is changed and $X^*(t)$ is no longer optimal but still has reasonably high quality. In order to “reactivate” the primary model of EDA_G , the secondary model must be able to produce a large number of individuals with higher qualities compared to the individuals produced by the primary model in order to update the Gaussian model towards the new global optimum $X^*(t + 1)$. This is because, in EDA_G , only the best individuals are responsible for updating the model while all other inferior individuals are discarded. However, for any non-trivial search space, randomly generated individuals cannot be expected to have high fitness in general. As a result, simply adding the secondary model into EDA_G is considered insufficient for handling dynamic environments.

According to the above analysis, a novel strategy is proposed by incorporating an optional model into EDA_G , which only becomes active in certain circumstances. First of all, it is assumed that the secondary model is capable of producing at least a single individual X^{best} that is better than all individuals produced by the primary model. Note that it is not compulsory that X^{best} must be found immediately after the change of the problem. Instead, the secondary model can be allowed a few iterations to find X^{best} . It is easy to see that this assumption is very reasonable as in theory the secondary model can always find such an individual as long as the number of generations allowed is sufficiently large.

Once X^{best} is found, the optional model will come into play: its mean vector is set to X^{best} and its variances are fixed to some predefined values. The purpose of the optional model is to devote a certain portion of the search effort to the region identified by X^{best} in the hope to find more individuals with better qualities than the individuals produced by the primary model. In summary, the secondary model is responsible for keeping EDA_G exploring the whole search space while the optional model is employed to increase the probability of finding good individuals so that the primary model can be updated in the correct way.

An important decision that needs to be made is when to put the optional model into action? Based on the motivation of this model, it should be used when the primary model is converging towards somewhere different from the location of X^{best} , which means that EDA_G is making a mistake. Since it is well known that the probability of generating a sample point more than three standard deviations away from the mean of the Gaussian distribution is very low, in practice, if X^{best} does differ from the mean of the model more than three standard deviations, it is deemed that the evolution process of EDA_G is getting into trouble and the optional model should be triggered.

The full procedure of the dynamic version of EDA_G (referred to as Tri- EDA_G) is shown in Table 2. Note that the best individual from the current population is always carried to the next population to make sure that the best solution found so far will not be lost due to random factors.

Table 2 The framework of Tri-EDAG $P_t^1(X)$: primary model; $P_t^2(X)$: secondary model; $P_t^3(X)$: optional model

Step 1: Initialize $P_0^1(X), P_0^2(X), P_0^3(X), t = 0$
Step 2: Sample O_1 from $P_t^1(X)$
Step 3: Sample O_2 from $P_t^2(X)$
Step 4: Sample O_3 from $P_t^3(X)$ if $P_t^3(X)$ is active
Step 5: $O = \{O_1, O_2, O_3\}$
Step 6: Evaluate individuals in $O : F = \{f(X_1), \dots, f(X_n)\}$
Step 7: Select a subset of the population $O' \subset O$
Step 8: Update $P_t^1(X)$
Step 9: Determine whether to activate $P_t^3(X)$
Step 10: Update $P_t^3(X)$ if $P_t^3(X)$ is active
Step 11: $t = t + 1$
Step 12: Go to Step 2 until stopping criteria are met

5 Experiments

5.1 Test problems

For any experimental studies of EAs, a key decision that needs to be made in the first place is what kind of test problems should be included in the experiments. Since the number of all possible optimization problems is infinite, drawing any general conclusions about the performance of the algorithms based on a few arbitrarily selected test problems is always dangerous [4, 11, 12].

In this paper, we restricted our attention to a series of test problems with one global optimum and one local optimum. A test problem generator was employed [5] to generate the stationary test problems as the foundation. Each test problem consisted of two Gaussian functions. One Gaussian function was scaled by a factor so that its peak value was equal to f_{global} (the predefined fitness value of the global optimum) and consequently its mean vector was the global optimum. The other Gaussian function (representing the local optimum) was also scaled so that its peak value was equal to f_{local} (the predefined fitness value of the local optimum). The fitness value of an individual was determined by the Gaussian function that gave it the highest value.

Each time when the problem was due to change, the global optimum was moved to a randomly generated location while the local optimum was moved to the original location of the global optimum. By doing so, individuals converging around the previous global optimum would still have reasonably high fitness values in the new problem (better than most of randomly generated individuals in the search space), which may make it difficult for EDAG to reactivate its primary model. A 1D example is plotted below where Fig. 2 (right) shows the structure of the problem right after the previous problem shown in Fig. 2 (left).

5.2 Results

All test problems were defined in 10D spaces and were bounded within $[-5, 5]$ in each dimension. Both two Gaussian functions had a diagonal covariance matrix with

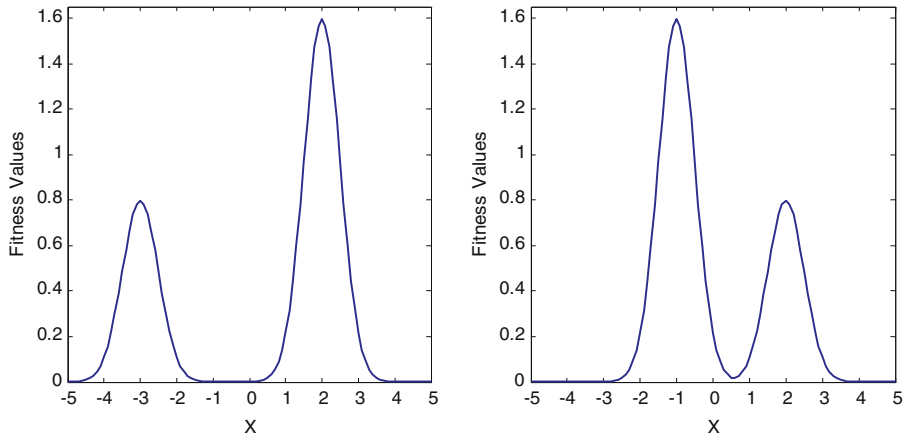


Fig. 2 An example of the structure of the test problem at different times

identical diagonal values 1.0 and were scaled to 1.0 (global optimum value) and 0.2 (local optimum value) respectively. The mean vectors of the two Gaussian functions were restricted with $[-2.5, 2.5]$ and were initialized randomly. In all the experiments, the test problems were updated with new locations of the two optima every 50 generations (certainly Tri-EDA_G was not informed in any way).

For Tri-EDA_G, the population size was 1000 and the truncation selection was employed, which selected the top 30% individuals in each generation to build the primary model. Normally, the primary model produced 80% of the individuals while the rest individuals in the population were produced by the secondary model (randomly generated). When the optional model (a Gaussian with identical variance value 1.0 in each dimension) was activated, it would account for 40% individuals in the population (the primary model would produce 40% individuals accordingly).

Fifty random problem sequences were tested each of which contained six different stationary test problems. Fig. 3 shows the typical performance of Tri-EDA_G on a certain problem sequence (Problem A). It is clear that Tri-EDA_G was always able to recapture the new global optimum (fitness value = 1.0) after each change of problem structure. The standard deviations of the primary model are plotted in Fig. 4, which shows that the diversity of the model increased dramatically (the transition from exploitation to exploration) a few generations after the problem was modified. By contrast, without the optional model, the capability of Tri-EDA_G in handling changing environments suffered significantly and the new global optimum was only found in one occasion. Similar conclusions can be also drawn from the results on another problem sequence (Problem B), as shown in Figs. 6, 7, and 8, respectively.

6 Conclusions

The major contribution of this paper is a novel triple-model strategy as the first attempt to extend the applicability of a class of continuous Gaussian EDAs to the domain of

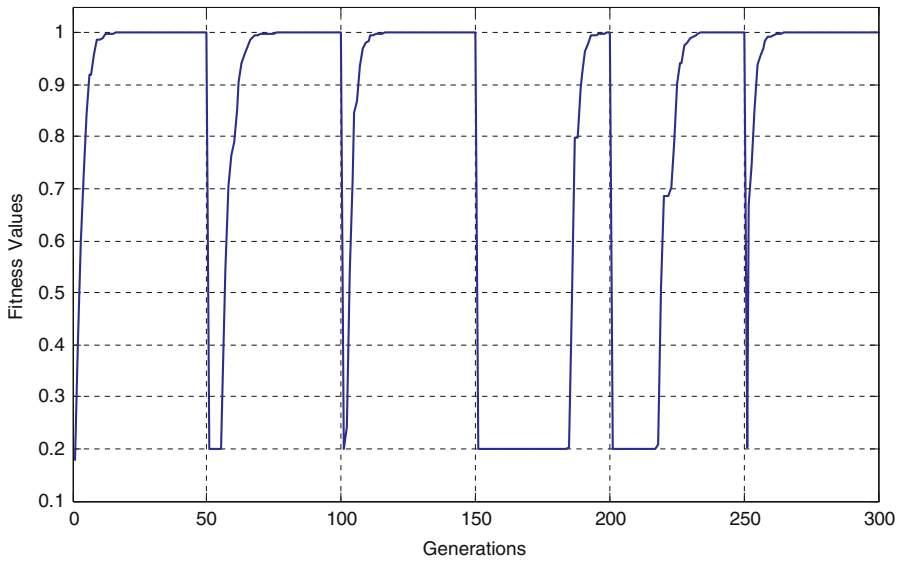


Fig. 3 The performance of Tri-EDA_G on Problem A

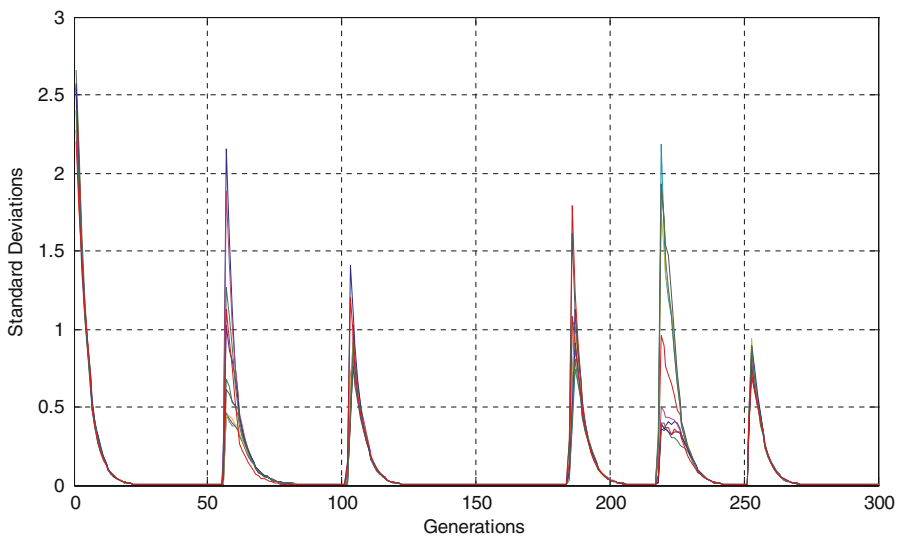


Fig. 4 The standard deviations of the primary model in Tri-EDA_G on Problem A

dynamic optimization. In addition to the primary model used in classical EDAs, a secondary model was employed to maintain their exploration capability in order to avoid premature convergence. Furthermore, an optional model was also incorporated into the algorithm framework, which may significantly improve the search efficiency of the secondary model. In the experimental studies, Tri-EDA_G performed reasonably well

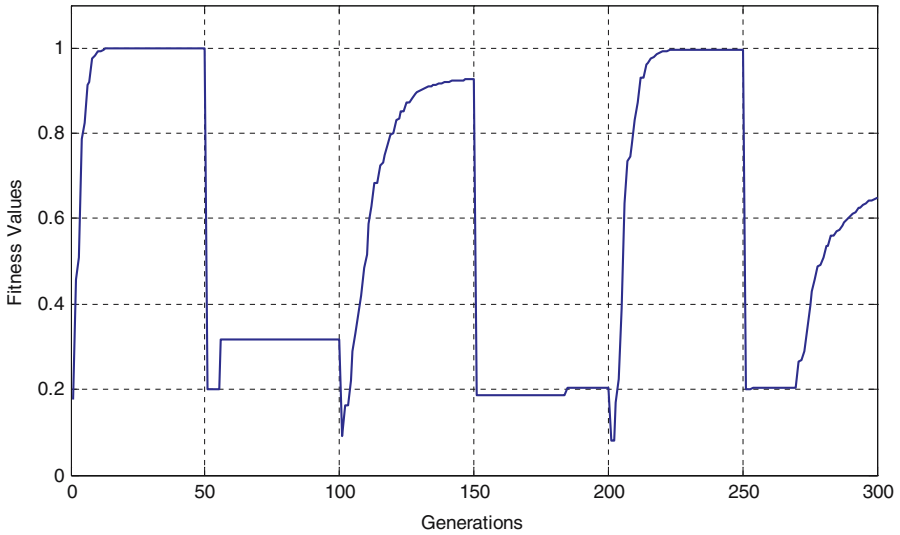


Fig. 5 The performance of Tri-EDA_G on Problem A without the optional model

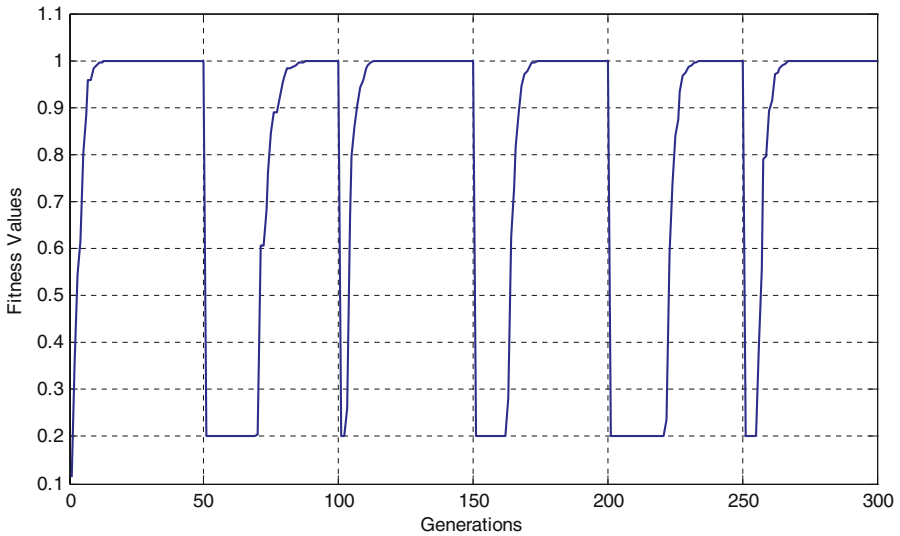


Fig. 6 The performance of Tri-EDA_G on Problem B

on a number of randomly generated problem sequences and consistently recaptured the new global optimum shortly after the problem was updated.

Certainly, the success of EAs/EDAs on dynamic optimization problems depends on a number of factors such as its parameter setting and the properties of the problems of interest. It is very likely that a strategy for dynamic optimization working well in one circumstance may not work well in others. The work presented in this paper was only

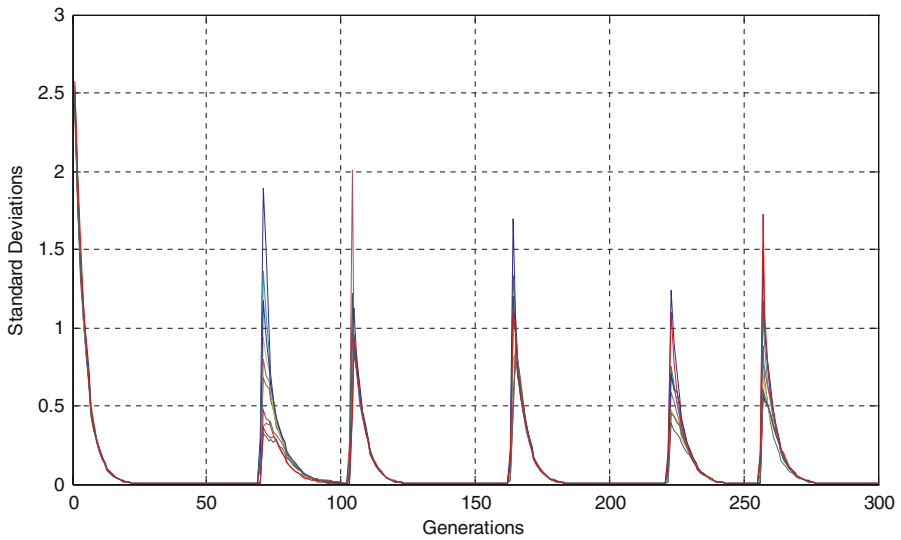


Fig. 7 The standard deviations of the primary model in Tri-EDA_G on Problem B

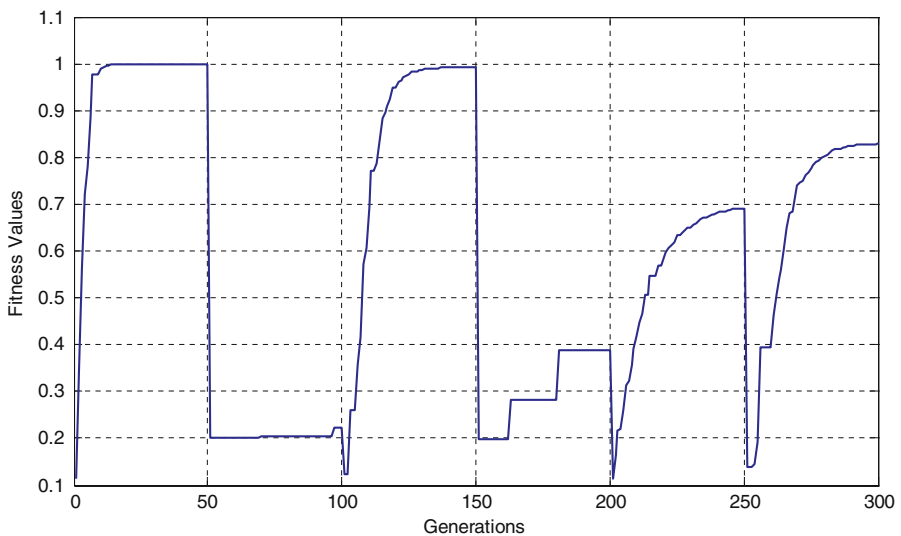


Fig. 8 The performance of Tri-EDA_G on Problem B without the optional model

conducted on a class of two-peak problems and more systematic studies are needed to fully investigate the advantages and weaknesses of Tri-EDA_G. It is expected that good dynamic optimization techniques should be designed by taking into account all these factors as well as having a good understanding towards the interactions among them.

References

1. Bäck, T., Fogel, D.B., Michalewicz, Z.: *Handbook of Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press, New York (1997)
2. Baluja, S., Davies, S.: Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space. In: *Fourteenth International Conference on Machine Learning*, pp. 30–38 (1997)
3. De Bonet, J.S., Isbell, C.L., Viola, P.: MIMIC: Finding Optima by Estimating Probability Densities. In: *Advances in Neural Information Processing Systems*, vol. 9. MIT, Cambridge, pp. 424–430 (1997)
4. Eiben, A.E., Jelasity, M.: A critical note on experimental research methodology in EC. In: *Congress on Evolutionary Computation*, pp. 582–587 (2002)
5. Gallagher, M., Yuan, B.: A general-purpose tunable landscape generator. *IEEE Trans. Evol. Comput.* **10**(5), 590–603 (2006)
6. Larrañaga, P., Etxeberria, R., Lozano, J.A., Pena, J.M.: Optimization by learning and simulation of Bayesian and Gaussian networks. Research Report EHU-kZAA-IK-4/99, University of the Basque Country (1999)
7. Larrañaga, P., Lozano, J.A.: *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Dordrecht (2001)
8. Larrañaga, P., Lozano, J.A., Bengoetxea, E.: Estimation of Distribution Algorithms based on multivariate normal and Gaussian networks. Technical Report KZZA-IK-1-01, University of the Basque Country (2001)
9. Pelikan, M.: Bayesian optimization algorithm: from single level to hierarchy. Ph.D. Thesis, University of Illinois at Urbana-Champaign (2002)
10. Pelikan, M.: *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*. Springer, Heidelberg (2005)
11. Whitley, D., Mathias, K., Rana, S., Dzubera, J.: Evaluating evolutionary algorithms. *Artif. Intell.* **85**(1–2), 245–276 (1996)
12. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**(1), 67–82 (1997)
13. Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: *The 2005 Genetic and Evolutionary Computation Conference*, pp. 1115–1122 (2005)
14. Yang, S., Ong, Y., Jin, Y.: Evolutionary computation in dynamic and uncertain environments. In: *Studies in Computational Intelligence*, vol. 51. Springer, Heidelberg (2007)
15. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comput.* **9**(11), 815–834 (2005)
16. Yuan, B., Gallagher, M.: Experimental results for the special session on real-parameter optimization at CEC 2005: a simple, continuous EDA. In: *Congress on Evolutionary Computation 2005*, pp. 1792–1799 (2005)
17. Yuan, B., Gallagher, M.: A mathematical modelling technique for the analysis of the dynamics of a simple continuous EDA. In: *The 2006 Congress on Evolutionary Computation*, pp. 1585–1591 (2006)
18. Yuan, B., Gallagher, M.: On the importance of diversity maintenance in estimation of distribution algorithms. In: *The 2005 Genetic and Evolutionary Computation Conference*, pp. 719–726 (2005)
19. Yuan, B., Orłowska, M., Sadiq, S.: Finding the optimal path in 3D spaces using EDAs—the wireless sensor networks scenario. In: *The 8th International Conference on Adaptive and Natural Computing Algorithms*, pp. 536–545 (2007)
20. Yuan, B., Orłowska, M., Sadiq, S.: On the optimal robot routing problem in wireless sensor networks. *IEEE Trans. Knowl. Data Eng.* **19**(9), 1252–1261 (2007)