Efficient Sampling When Searching for Robust Solutions

Juergen Branke(✉) and Xin Fei

Warwick Business School, University of Warwick, Coventry, UK
juergen.branke@wbs.ac.uk, xin.fei.14@mail.wbs.ac.uk

Abstract. In the presence of noise on the decision variables, it is often desirable to find *robust* solutions, i.e., solutions with a good *expected* fitness over the distribution of possible disturbances. Sampling is commonly used to estimate the expected fitness of a solution; however, this option can be computationally expensive. Researchers have therefore suggested to take into account information from previously evaluated solutions. In this paper, we assume that each solution is evaluated once, and that the information about all previously evaluated solutions is stored in a memory that can be used to estimate a solution's expected fitness. Then, we propose a new approach that determines which solution should be evaluated to best complement the information from the memory, and assigns weights to estimate the expected fitness of a solution from the memory. The proposed method is based on the Wasserstein distance, a probability distance metric that measures the difference between a sample distribution and a desired target distribution. Finally, an empirical comparison of our proposed method with other sampling methods from the literature is presented to demonstrate the efficacy of our method.

1 Introduction

Many practical real-world problems involve uncertainty on decision variables. For example, in engineering, the actual product often does not correspond to the original design because of manufacturing tolerance. In such cases, the solutions should not only be good, but also *robust*. If $\xi \in \Xi$ are the possible disturbances to the decision variables, then the solution's expected fitness (which in the following, due to consistency with previous publications, we call *effective* fitness) is

$$f_{eff}(x) = \int_{\Xi} f(x+\xi) dP(\xi) \tag{1}$$

where $P(\xi)$ is the probability distribution of disturbance ξ. The effective fitness can be estimated by sampling as $\hat{f}_{eff}(x) = \sum_n f(x+\xi_n), \xi_n \in \Xi$. However, this is computationally expensive. Several researchers have thus attempted to speed up the search for robust solutions, surveys on these topics can be found in [2,8].

A previous study [11] has suggested that, for evolutionary algorithms (EAs), a single disturbed sample $f(x+\xi)$ that is used to evaluate a solution may actually

© Springer International Publishing AG 2016
J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 237–246, 2016.
DOI: 10.1007/978-3-319-45823-6_22

be sufficient. The same study has reported that, in the case of infinite population size, an evolutionary algorithm with a single disturbed sample employed to evaluate each individual behaves in an identical manner to an evolutionary algorithm that operates directly on the effective fitness function. In the context of evolution strategies, [1] propose a mechanism to adaptively increase the population size over a run, along with a mechanism that adjusts mutation to account for the noise on the decision variables. To improve the estimate of an individual's effective fitness, previous studies have proposed to compute for the average of multiple samples, preferably based on Latin Hypercube Sampling [3,5].

EAs are population-based iterative search methods; hence, they usually converge to a promising region of the search space and then evaluate many samples in this area. Hence, at least towards the end of the optimisation run, when the EA would like to evaluate a solution, information about many other solutions in the neighbourhood is likely to be available if it is stored in a memory. This information can be exploited when estimating the robustness of a solution. Two questions arise:

1. How should the fitness values from the memory be weighted to yield a good (i.e., accurate and unbiased) estimate of the effective fitness of an individual?
2. If new information in terms of additional fitness evaluations can be collected, at what location(s) should this information be collected?

In [3], a new sample is taken at $\xi = 0$, and all the previous fitness values are weighted with the probability that a disturbance might actually result in the corresponding decision vector. However, this may result in a rather biased estimate if the distribution of memory samples is quite different from the distribution of expected disturbances. [9] propose to generate several candidate disturbances ξ_n, and then select the one that has the maximal minimum distance to any of the existing memory samples. This aims to fill in gaps in the distribution of memory samples; however, it is a rather simple heuristic and often results in extreme solutions being evaluated that are close to the disturbance boundary. [10] uses surrogate models to estimate the effective fitness.

In this paper, we propose a new method based on the Wasserstein distance to address the above two questions mentioned above. The Wasserstein distance measures the distance between two probability measures. The idea is to derive a large-sample target distribution from the known probability distribution of disturbances, and then collect new information and reallocate weight values such that the Wasserstein distance between the used samples and the large-sample target distribution is minimised.

The paper is structured as follows. Section 2 describes our proposed method and the mathematical foundation. Section 3 reports on several empirical experiments and a comparison with other methods from the literature. Finally, the paper concludes with a summary and some ideas for future work.

Algorithm 1. EA with ASA

Set $t \leftarrow 1$, initialise population P^t.
while Termination criterion is not met **do**
  Generate offspring population O from P^t.
  Generate N disturbance samples $x_n^t \in \Xi^t, n = 1, \ldots, N$
  for each solution $x^m \in O, m = 1, \ldots, \lambda$ **do**
    Compute approximate target $z_n = x^m + \xi_n^t$
    Identify memory solutions in neighbourhood $\mathcal{A}(x^m)$
    Construct N approximate set candidates $\mathcal{Y}^n(x^m) = \mathcal{A}(x^m) \cup z_n^t$
    Compute the Wasserstein distance value of each approximate set $\mathcal{Y}^n(x^m)$.
    Select the best approximate set $\mathcal{Y}^*(x^m)$ with the minimum distance value.
    Compute the optimal weight values $P(\mathcal{Y}^*(x^m))$.
    Compute $\hat{f}_{eff}(x^m) = \sum_{k=1} P(y_k) f(y_k), y_k \in \mathcal{Y}^*(x^m)$.
  end for
  Set $t \leftarrow t + 1$, update population P^t according to $\hat{f}_{eff}(x^m), m = 1, \ldots, \lambda$
end while

2 Proposed Method

This section describes our proposed method called archive sample approximation (ASA). The following notations are used throughout the paper:

- $\xi_n \in \Xi, n = 1, \ldots, N$: the underlying disturbance on decision variables.
- $z_n = x + \xi_n \in \mathcal{Z}(x), n = 1, \ldots, N$ (*approximation target*): one realisation of disturbed solution x.
- $y_k \in \mathcal{Y}(x), k = 1, \ldots, K (K \leq M)$ (*approximation set*): the set is used to approximate $\mathcal{Z}(x)$.
- $P(\mathcal{Z}), P(\mathcal{Y})$: the probability measure over approximation sets $\mathcal{Z}(x), \mathcal{Y}(x)$.
- $W(P(\mathcal{Y}), P(\mathcal{Z}))$: the Wasserstein distance between $P(\mathcal{Y})$ and $P(\mathcal{Z})$.
- $\mathcal{A} = \{a_1, \ldots, a_L\}$: the archive of previous fitness evaluations.
- $\mathcal{A}(x)$: a subset of $\mathcal{A}$ that contains locations in the "disturbance neighbourhood"of solution x.

Algorithm 1 describes how ASA can be integrated into an evolutionary algorithm. First, we generate a set of disturbances Ξ from the underlying noise distribution. Note that the disturbance set changes in every generation, but we use the same disturbances for all the individuals within one generation. With disturbance set Ξ, we can generate the approximation target set $\mathcal{Z}(x^m)$ for solution x^m that would, if evaluated, allow us to compute a good-enough estimate of the individual's effective fitness.

Next, we search $\mathcal{A}(x^m)$ in the "disturbance neighbourhood" of solution x^m from archive $\mathcal{A}$, and include all available memory solutions into approximation set $\mathcal{Y}(x^m)$. Meanwhile, we would like to add one additional disturbance realisation z_n for solution x^m into its approximation set. As candidates, we consider all the points in the target set $\mathcal{Z}(x)$, and try inserting each one, resulting in N approximation sets $\mathcal{Y}^n(x^m) = \mathcal{A}(x^m) \cup z_n$. The goal of the ASA procedure is to

find the best approximation set $\mathcal{Y}^*(x^m)$ with probability measures $P(\mathcal{Y}^*(x^m))$ to well approximate the target $\mathcal{Z}(x^m)$.

Our algorithm uses the Wasserstein distance [6] to decide which approximation set is the best option (i.e., where to sample) and how to weigh the samples. This paper implements the L_1 Wasserstein distance to quantify the error in the approximation set, which can be computed by solving the Kantorovich-Rubinstein transportation problem, as follows.

$$\begin{aligned} W(P(\mathcal{Y}^n), P(\mathcal{Z})) = \min_{\mu} \quad & \sum_k \sum_n d(y_k, z_n)\mu(y_k, z_n) \\ \text{s. t.} \quad & \sum_k \mu(y_k, z_n) = P(z_n), \quad \forall n \\ & \mu \geq 0 \end{aligned} \tag{2}$$

Once we obtain the optimal "transportation plan" μ^* in (2) is obtained, the optimal weights (probability measure $P(\mathcal{Y}^n)$ can be determined immediately by using

$$P(y_k) = \sum_n \mu^*(y_k, z_n), \quad \forall k. \tag{3}$$

To identify the best candidate, we simply add, one by one, each target point to the set of relevant memory locations $\mathcal{A}(x)$, and compute their Wasserstein distance values $W(P(\mathcal{Y}^n), P(\mathcal{Z}))$. The ASA algorithm aims to find the approximation set $\mathcal{Y}^*$ with the minimum distance value given by

$$W(P(\mathcal{Y}^*), P(\mathcal{Z})) = \min_n (W(P(\mathcal{Y}^n), P(\mathcal{Z}))). \tag{4}$$

Finally, we discuss the computation issue of linear program (2) and its efficient solution method. For the Kantorovich-Rubinstein transportation problem, the computation complexity is significantly influenced by the size of the approximation target. In this paper, we apply duality theory to reduce the computational effort. We assume that η_n is the dual decision variable for the n^{th} constraint in (2), then we have

$$\begin{aligned} W(P(\mathcal{Y}^n), P(\mathcal{Z})) = \max_{\eta} \quad & \sum_n P(z_n)\eta_n \\ \text{s.t.} \quad & \eta_n \leq d(y_k, z_n), \quad \forall n, \quad \forall k \end{aligned} \tag{5}$$

The optimal value is found if η_n satisfies

$$\eta_n^* = \min_n \; d(y_n, z_k) \tag{6}$$

Hence, the Wasserstein value can be computed by using

$$W(P(\mathcal{Y}^n), P(\mathcal{Z})) \leq \sum_n P(z_n)\eta_n^*. \tag{7}$$

The equality will hold if the linear program exhibits *strong duality.* By nature of the transportation problem, the optimal decision algorithm should select the

closest starting point for each destination to minimise the total transportation cost. Therefore, the optimal weight value for each element in the approximation set can be computed as

$$P(y_k) = \frac{\sum_n \lambda^{kn} P(z_n)}{N} \tag{8}$$

with

$$\lambda^{kn} = \begin{cases} 1 & if\ y_k\ is\ the\ closest\ sample\ to\ z_n \\ 0 & otherwise \end{cases}$$

where λ^{kn} is an index function that is used to count the number of times y_k is the closest sample to z_n.

3 Numerical Experiments

3.1 Test Functions

We demonstrate the performance of our proposed method on three 5-D test problems listed in Table 1. A 1-D visualisation of each test function is shown in Fig. 1. TP 1 has a single asymmetric peak and has been adapted from [10]. It will allow to examine an algorithm's ability to precisely identify the location of the robust solution. TP 2 has been taken from [4] and is multi-modal. The original fitness function has its optimum at $x = 1$, whereas the optimum of the effective fitness is at $x = -1$, which allows to test whether an algorithm is able to correctly identify the robust optimum. TP 3 combines both characteristics and has been adapted from a function used in [10].

Table 1. Test function description

	Formulation	Noise
TP 1	$\min \quad 0.9d + \sum_{i=1}^{d} Q_1(x_i),$ $Q_1(x_i) = \begin{cases} -(8-x_i)^{0.1} e^{-0.2(8-x_i)} & x_i < 8 \\ 0 & otherwise \end{cases}$ $x \in [0, 10]$	$U(-1, 1)$
TP 2	$\min \quad \sum_{i=1}^{d} Q_2(x_i)$ $Q_2(x_i) = \begin{cases} -(x_i+1)^2 + 1.4 - 0.8\,\lvert sin(6.283x_i)\rvert & -2 < x_i < 0 \\ 0.6 \cdot 2^{-8\lvert x_i - 1\rvert} + 0.958887 - 0.8\,\lvert sin(6.283x_i)\rvert & 0 \le x_i < 2 \\ 0 & otherwise \end{cases}$ $x \in [-2, 2]$	$U(-0.2, 0.2)$
TP 3	$\min \quad \sum_{i=1}^{d} Q_3(x_i)$ $Q_3(x_i) = 2sin(10e^{(-0.2x_i)}x_i)e^{(-0.25x_i)}$ $x \in [0, 10]$	$U(-1, 1)$

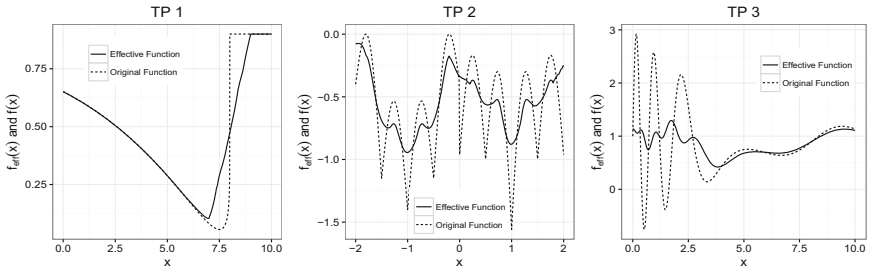

Fig. 1. 1-D visualisation of test functions

3.2 Evolutionary Algorithm

Our method can be combined with any metaheuristic. In this paper, we use a standard CMA-ES [7] for our experiments, with $\mu = 4, \lambda = 8$, initial $\sigma_0 = \frac{1}{4} Search\,Interval\,Width$ and equal weighting of the four individuals to determine the next centre of the mutation distribution.

3.3 Final Solution Selection and Performance Measure

Due to the noise, the effective fitness estimated by the algorithm is likely to deviate from the true effective fitness. For this reason, we use the barycenter of the selected parents as the solution that would be returned to the user.

$$x_{final} = \sum_{i=1}^{\mu} w_i x_i, \quad w_i = \frac{1}{\mu}$$

The effective fitness of final solution is evaluated using Monte-Carlo simulation with $N = 10,000$ samples.

$$f_{eff}(x_{final}) = \frac{1}{N} \sum_{i=1}^{N} f(x_{final} + \xi_i)$$

In order to better understand the quality of the effective fitness estimation, we furthermore report on the average absolute error AE_t by calculating the mean squared error between the true and approximate effective fitness as follows.

$$AE_t = \frac{1}{\lambda} \sum_{j=1}^{\lambda} \left| f_{eff}(x_m) - \hat{f}_{eff}(x_m) \right|, \quad x_m \in P_t$$

3.4 Target Samples Generation

We test the following three techniques for generating target samples.

1. Monte Carlo sampling (MC).
2. Latin hypercube sampling (LHS).
3. Equidistant sampling (ES) which places all samples on a regular grid with three points in each dimension.

3.5 Experimental Setup

We compare our method with three alternative approaches:

1. SEM: take one random sample for each solution. This is the approach proposed in [11].
2. SEM+AR: take one additional random sample for each solution, but also take into account all memory points in the area of disturbance. The new sample and all memory points are equally weighted. This is the approach proposed in [3].
3. ABRSS: A method that uses Latin Hypercube Sampling as reference points, and then includes for each reference point the closest memory point. To add a new sample, some random samples are generated and the one furthest from any memory point is selected. This method has been proposed in [9].

All methods are incorporated into CMA-ES, the size of target samples or reference points is $3^5 = 243$ for all methods. All reported results are averaged over 30 runs. The computational budget for each run was 2,500 fitness evaluations.

3.6 Results on Convergence Rate and Average Approximation Error

Figures 2 and 3 compare the convergence rate and approximation error of different methods. The effective fitness of the final solution is also reported in Table 2. As can be seen, ASA has the best convergence behaviour and smallest approximation error in all three test problems. The use of the Wasserstein distance effectively controls the approximation error, and thus it has a fast convergence rate. On TP 1, SEM+AR works almost as good as ASA. Because this problem is unimodal, all algorithms converge to the correct peak, and a lot of memory samples accumulate there, leading to a very small approximation error also for SEM+AR. ABRSS is much worse, probably because its sampling mechanism tries to sample away from existing memory samples, which in this case means at less relevant points and introducing a bias. The increase in approximation error for SEM and ABRSS can be explained by their focusing on the peak area, which has a very large gradient.

ABRSS is the second best method for TP 2 and TP 3. Since SEM and SEM+AR draw samples randomly, they are more prone to "lucky" over evaluations of individuals. As a consequence, they always discover new presumably good solutions, move there, and then realise after some time that the solution was actually not really very good, leading to a jumping behaviour from one local optimum to another.

3.7 Influence of the Target Sample Generation Mechanism

ASA requires a set of target samples to start with. To better understand the influence of the target sample generation mechanism, we compare the influence of different sample generation methods in Figs. 4 and 5. LHS (which we also

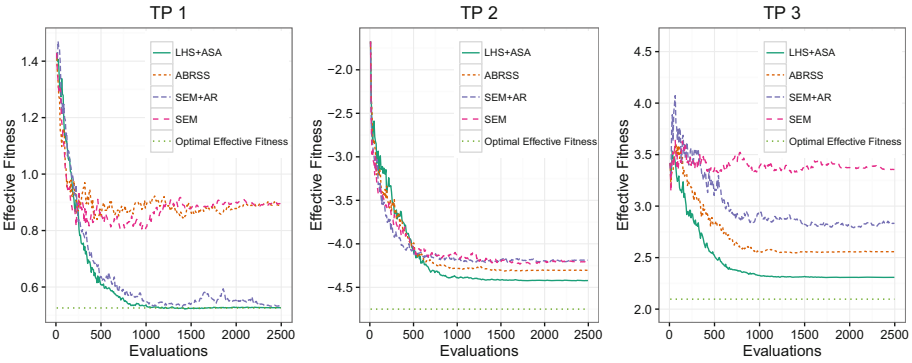

Fig. 2. Convergence rate of different methods

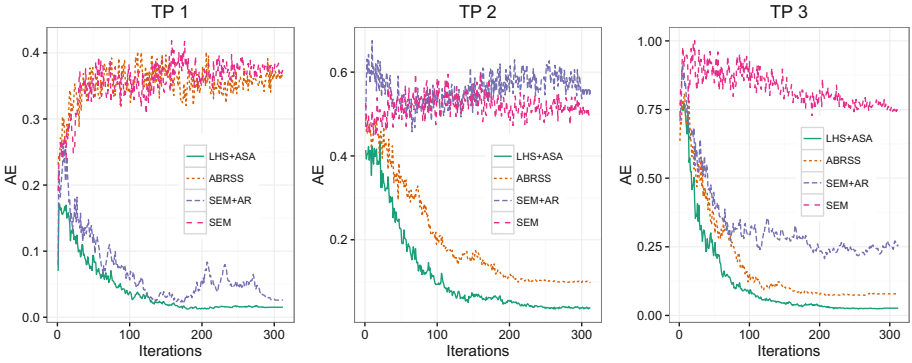

Fig. 3. Average absolute error of different methods

Table 2. Effective fitness after 2,500 evaluations

Method	$mean \pm s.e.$		
	TP 1	TP 2	TP 3
SEM	0.8988 ± 0.0314	-4.1279 ± 0.0434	3.3569 ± 0.0945
SEM + AR	0.5388 ± 0.0069	-4.1845 ± 0.0420	2.8368 ± 0.0939
ABRSS	0.8893 ± 0.0451	-4.3293 ± 0.0388	2.5574 ± 0.0717
LHS + ASA	0.5269 ± 0.0013	-4.4231 ± 0.0362	2.3083 ± 0.0297

used in the previous experiment) performs well in all test functions. Interestingly, equidistant sampling outperforms other sampling methods in TP 3, but produces a bad solution in TP 1. This is probably because the boundary of the disturbance region has a particular large influence for TP 1, and the way we chose the grid structure that boundary region was never sampled.

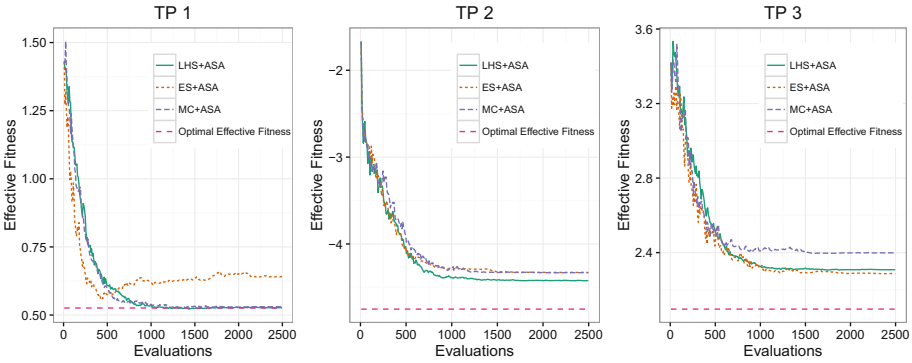

Fig. 4. Convergence rate of different target sample generation schemes

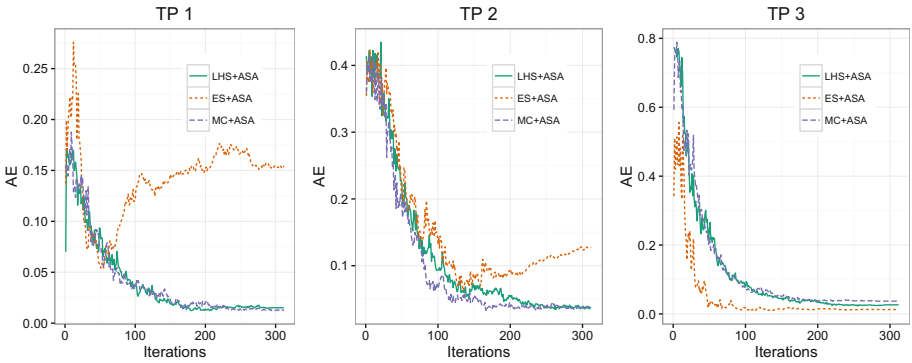

Fig. 5. Average absolute error of different target sample generation schemes

Table 3. Effective fitness after 2,500 evaluations

Target samples	$mean \pm s.e.$		
	TP 1	TP 2	TP 3
MC+ASA	0.5304 ± 0.0030	-4.3308 ± 0.0625	2.3988 ± 0.0583
ES+ASA	0.6416 ± 0.0076	-4.3042 ± 0.0466	2.2872 ± 0.0439
LHS+ASA	0.5269 ± 0.0013	-4.4231 ± 0.0362	2.3083 ± 0.0297

4 Conclusion

We have looked at the problem of searching for robust solutions, where robust means a good expected fitness over a given distribution of disturbances to the decision variables. In particular, we have re-considered the idea of estimating a solution's effective fitness by making use of previous fitness evaluations stored in the memory. We proposed a methodology based on the Wasserstein distance to decide at what location we should evaluate the fitness in order to gain the most

useful additional information about a solution's effective fitness, and also how to weigh the different samples in the neighborhood for estimating the effective fitness. Empirical comparisons with several previous methods for this problem on three test functions demonstrates the superiority of our new approach.

Future work will include developing other distance metrics and moving from an individual based view to a population based view when determining where to evaluate fitness.

References

1. Beyer, H.-G., Sendhoff, B.: Evolution strategies for robust optimization. In: World Congress on Computational Intelligence, pp. 4489–4496. IEEE (2006)
2. Beyer, H.-G., Sendhoff, B.: Robust optimization - a comprehensive survey. Comput. Methods Appl. Mech. Eng. **196**(33), 3190–3218 (2007)
3. Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 119–128. Springer, Heidelberg (1998)
4. Branke, J.: Evolutionary Optimization in Dyamic Environments. Kluwer, Boston (2001)
5. Branke, J.: Reducing the sampling variance when searching for robust solutions. In: Genetic and Evolutionary Computation Conference, pp. 235–242. Morgan Kaufmann, San Francisco (2001)
6. Dudley, R.M.: Real Analysis and Probability, vol. 74. Cambridge University Press, Cambridge (2002)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
8. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. Evol. Comput. **9**(3), 303–317 (2005)
9. Kruisselbrink, J., Emmerich, M., Bäck, T.: An archive maintenance scheme for finding robust solutions. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 214–223. Springer, Heidelberg (2010)
10. Paenke, I., Branke, J., Jin, Y.: Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. IEEE Trans. Evol. Comput. **10**(4), 405–420 (2006)
11. Tsutsui, S., Ghosh, A.: Genetic algorithms with a robust solution searching scheme. IEEE Trans. Evol. Comput. **1**(3), 201–208 (1997)