Creating Robust Solutions by Means of Evolutionary Algorithms

Jürgen Branke

Institute AIFB
University of Karlsruhe
D-76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

Abstract. For real world problems it is often not sufficient to find solutions of high quality, but the solutions should also be robust. By robust we mean that the quality of the solution does not falter completely when a slight change of the environment occurs, or that certain deviations from the solution should be tolerated without a total loss of quality.
In this paper, a number of modifications to the standard evolutionary algorithm (EA) are suggested that are supposed to lead the EA to produce more robust solutions. Some preliminary experiments are reported where the proposed approaches are compared to a standard model. As it turns out, the EA's ability to create robust solutions can be greatly enhanced even without additional function evaluations.
Keywords: evolutionary algorithm, robust solution

1 Introduction

For real world problems, it is often important to create robust solutions to a problem, i.e. solutions that are insensitive to changes in the environment or noise in the decision variables.

Some examples for applications include:

- in scheduling problems, solutions are sought that allow e.g. slight variation in processing times, time for machine breakdowns or the incorporation of an additional urgent job without requiring a total reordering of the production plan.
- for many control problems, the environment may change slowly, e.g. machines can slowly wear out or the composition/quality of the raw material can change slightly. Often it is not possible to constantly monitor and control the process, thus it is advantageous to implement solutions that yield good results over a certain range of environmental conditions.
- often, certain tolerances have to be allowed for production. Solutions are sought that give good results over all settings within these tolerances.

From a more abstract perspective, this means that not only the solution should be good, but also that the (phenotypic) neighborhood of the solution should have a high average quality. Looking at the fitness landscape, a solution on a high plateau should be preferred over a solution on a thin peak: if the environment changes slightly (e.g. shifts in a random direction) or if it can not be guaranteed that the exact parameters of the solution are actually implemented (but a solution close to the original solution), the solution on the plateau will yield much better expected quality than the solution on the peak. This *effective* fitness function, $f_{eff}(x)$, depends on the distribution of the noise added to the input and could be calculated as $\int_{-\infty}^{\infty} p(\delta) \cdot f(x + \delta)\, d\delta$, with $p(\delta)$ being the probability density function for the disturbance δ. Of course, for problems of relevant complexity, this calculation will not be possible, thus $f_{eff}(x)$ has to be estimated.

In this paper, it is tried to tune evolutionary algorithms (cf. e.g. [3,7]) to produce robust solutions. All suggested approaches use a modified fitness value f_{mod} for selection that is designed to be a reasonable estimate of the effective fitness function f_{eff}.

Some related work, developed independently of the work presented here, has recently been published in [12]. Earlier related work includes [6,8–10] and [11].

We apply a rather simplistic setting for investigating the robustness of solutions: a simple multi-modal mathematical function is maximized by the EA. To determine the final solution's quality (i.e. to obtain a good estimate of its effective fitness), the average of 100 disturbed evaluations is used. For disturbance, a normally distributed noise is added to the phenotypic values of the solution. Assuming that these disturbances might actually be experienced in practice, the solution is evaluated according to its expected performance in that environment.

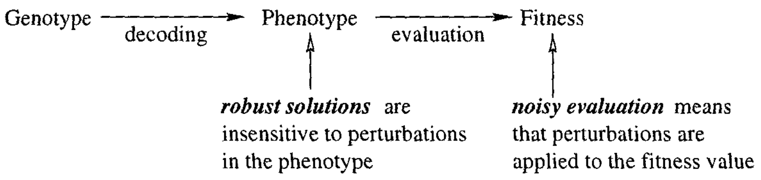
Genotype
decoding
Phenotype
evaluation
Fitness
robust solutions are insensitive to perturbations in the phenotype
noisy evaluation means that perturbations are applied to the fitness value

Fig. 1. Robust solutions vs. noisy fitness evaluation

Note that the problem of creating robust solutions as defined here has some similarities to optimizing noisy functions, which have already been examined in combination with EAs (e.g. [1,2,4]). However there are two main differences:

- with noisy functions, some noise is added to the output (quality) of the solution. In the settings regarded here, noise is added to the decision variables (or phenotype) of the solution. I.e. if $f(x)$ is the fitness function and δ is some (normally distributed) noise, then a noisy fitness function would mean $f'(x) = f(x)+\delta$, while in our case $f'(x) = f(x+\delta)$ (cf. Fig. 1 for illustration).
- noisy functions can not be evaluated without noise, the EA has to find good solutions despite the noise. In the settings considered in this paper, it is

assumed that only the decision variables of the final solution are disturbed, usual function evaluations during the EA run can be performed without disturbance. This is justified because that evaluation is usually done on the basis of a theoretical, computerized model, while the final solution is then actually implemented and has to face all the uncertainties present in reality.

The outline of the paper is as follows:
Section 2 describes in more detail the environments used for testing. In Section 3 we propose a number of modifications to the standard EA, aimed at delivering robust solutions. These approaches are evaluated empirically, the results obtained can be found in Section 4. The paper concludes with a summary and a number of ideas for future work.

2 Test Functions

For testing the approaches presented in Section 3, the following two 10-dimensional test functions have been used:

$$f_1(x) = \sum_{i=1}^{10} \sin\sqrt{|40x_i|} + \frac{20-|x_i|}{b} \qquad -20 \le x_i < 20 \tag{1}$$

and

$$f_2(x) = \sum_{i=1}^{10} g(x_i) \quad \text{with} \quad g(x_i) = \begin{cases} -(x_i+1)^2+1 & : \quad -2 \le x_i < 0 \\ c \cdot 2^{-8|x_i-1|} & : \quad 0 \le x_i < 2 \end{cases} \tag{2}$$

Plots of the two functions along one dimension are shown in Fig. 2. The parameters b resp. c allow to influence the height of the sharp peaks relative to the smoother hills. In the experiments, $b = 20$ and $c = 1.3$ were used.

Function f_1 has been designed to allow the EA to choose between a number of alternatives concerning the tradeoff between robustness and quality. Function f_2 is much simpler, having only one hill and one peak in each dimension, while the basin of attraction for both, hill and peak, is of equal size.

The final solution of the EA is then tested 100 times, perturbed before each evaluation. As perturbation, the individual is shifted slightly by adding to each decision variable (in our case equivalent to a gene) a randomly chosen value from a normal distribution with mean 0 and standard deviation 0.5 for function f_1 resp. a standard deviation of 0.1 for function f_2. The resulting effective fitness functions $f_{1,eff}(x)$ and $f_{2,eff}(x)$ are displayed in Fig. 3.

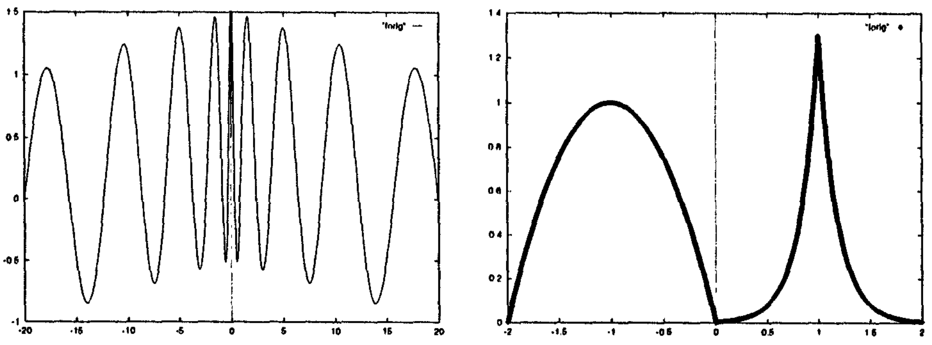

Fig. 2. Test function $f_1(x)$ with b=20 (left) and $f_2(x)$ with c = 1.3 (right)

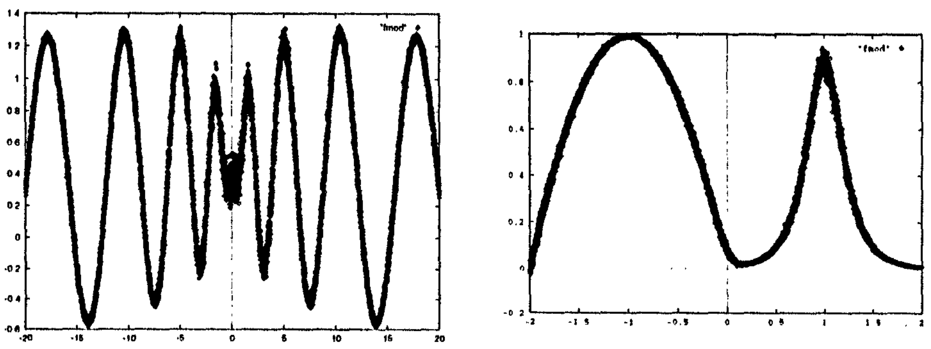

Fig. 3. Effective test functions $f_{1,eff}(x)$ (left) and $f_{2,eff}$ (right) when disturbed.

3 Tested Approaches

3.1 Simple EA

The standard EA by nature favors hills over peaks, since usually the probability that some individuals of the initial population are somewhere in the basin of attraction of a hill is higher, also the average fitness of individuals in the hill area is higher.

Thus, when for example in Fig. 2 (right) the two maxima are made equally high by setting c in Eq. (2) to 1.0, the standard EA will turn to the smooth hill rather than to the peak in 99% of the trials.

But as the experiments show, the higher the peak compared to the hill, the more often the standard EA will choose the peak, although this may not be advantageous in the environment considered in this paper. In this case, significant improvements can be obtained by slight modifications to the standard model.

As simple EA we use real-valued, direct encoding of the decision variables, generational replacement with elitism (i.e. the best individual survives), ranking selection, two-point-crossover, mutation rate of 0.1, and an island model[1] with 5 subpopulations, 50 individuals each, run for 500 generations. The same settings are used throughout all experiments unless stated otherwise.

3.2 Mean of several evaluations

To estimate the average quality of an individual's neighborhood, one possibility is to probe the fitness landscape at several random points in the neighborhood and calculate the mean. This value may then be used as that individual's fitness f_{mod}. The effect is a smoothing of the fitness landscape, with sharp peaks being flattened and smooth hills being almost non-affected.

Major drawback of this method is the increased computation time: since usually evaluation is the time determining factor, evaluating several points for one individual is very expensive. Furthermore, one has to decide how many probes in the neighborhood are necessary and how this neighborhood should be defined.

To investigate the first question, a number of experiments are conducted with different numbers of evaluations per individual. Concerning the definition of the neighborhood, it seems best to approximate the distribution of noise expected for the final solution, and use the same distribution for adding noise to the individuals during the run.

In the experiments reported below, it is assumed that the distribution of the noise finally applied to the solution is known, thus the same noise is applied to the decision variables when an individual is evaluated several times. The same assumption has been used in the approaches of Section 3.3 and Section 3.4.

3.3 Single disturbed evaluations

Instead of reevaluating several times with disturbed decision variables, one might as well just evaluate once at a disturbed position. If the input (i.e. the individual's phenotype as described by its genes) to the evaluation function is disturbed randomly for every evaluation, the *expected* returned value at every point in the search space is just equivalent to the mean over it's neighborhood. Since in EAs, promising areas are probed more often, this might be sufficient. In fact, Tsutsui and Gosh [12] showed, using the schema theorem, that given an infinitely large population size, an EA with disturbed evaluations is actually performing as if it would work on the effective fitness function.

Additionally, evaluation results in flat areas will be more consistent over time thus it could be easier for the EA to focus on these areas.

[1] from our experiences with EAs (e.g. [5]), the island model usually performs better than a single population model, thus we meanwhile use the island model as standard

3.4 Reevaluating only the best

As has been mentioned above, repeatedly evaluating the same (disturbed) individual may be extremely expensive. Thus it seems reasonable to try to achieve the same results by evaluating each individual once without disturbance, and then reevaluate only the best individuals several times. It may not be necessary to repeatedly evaluate mediocre individuals, since they won't be implemented in practice anyway.

In the experiments reported below, three settings were tested:

- the best individual is evaluated another 4 times
- the best 20% of the population (i.e. 10 individuals in our experiments) are evaluated another 2 times
- the best 20% individuals of the population are evaluated another 4 times, the next 20 % are evaluated another 3 times, etc. Note that the sum of function evaluations for this approach is just equal to the number of function evaluations necessary when each individual is evaluated 3 times (Section 3.2).

3.5 Looking at other individuals in the neighborhood

Even if only a part of the population is evaluated repeatedly, additional computational cost is incurred. To avoid additional evaluations, one might also use the information available anyway: the individuals evaluated so far.

Thus, the idea of this approach is to calculate a weighted mean of the fitness of the current individual and the fitnesses of other (previously evaluated) individuals in its neighborhood.

As individuals that could be taken into account, one might at least consider the current population, but it is certainly helpful to store a selection of individuals from previous generations as well. Appropriate strategies for selecting which individuals to keep in the memory are currently under investigation.

For this paper, two simple strategies have been implemented:

- use the current population only
- use a memory, in which the last 5000 individuals are stored.

Since both approaches suffer from the potential problem that when the population converges, there are too many equivalent individuals that do not provide any additional information, a variant with duplicate avoidance has also been tested. In these cases, an individual is mutated again when it already exists in the population.

Each individual x is compared with all individuals y in the population resp. the memory, and its modified fitness, $f_{mod}(x)$, is calculated as:

$$f_{mod}(x) = \frac{\sum_y w(y) \cdot f(y)}{\sum_y w(y)} \tag{3}$$

Ideally, the weight function $w(y)$ should reflect the probability that x is turned into y by a disturbance, i.e. $w(y) \propto p(\delta)$ such that $y = x + \delta$.

However in the experiments described in this paper, for reasons of computational simplicity, not the actual noise distribution has been used for weighting (which would have been the equivalent to the previous approaches). Instead, the following simple ad hoc neighborhood function was chosen.

$$w(y) = max\{0, 1 - d * b\} \tag{4}$$

with $w(y)$ being the weight of the individual y, b being a parameter that has been set to 4 in function f_1 and to 1 in function f_2 (see above), and d the distance from y to the reference individual x currently being evaluated.

Note that this approach is based on the assumptions that the distance between individuals can be measured and the evaluation of an individual is much more expensive than the calculation of the weighted mean.

4 Results and Discussion

In the first set of experiments, only the method from Section 3.2, where the modified fitness is calculated as the average from n disturbed evaluations, has been tested, and only on Function f_2. As expected, the resulting effective quality increases with the number of function evaluations, while the benefit from an additional evaluation decreases with the number of function evaluations. The result is depicted in Fig. 4.

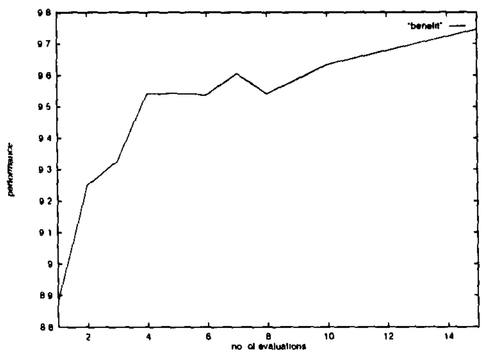

Fig. 4. Quality vs. number of trials per evaluation, averages over 10 runs

For comparisons between the different proposed approaches, extensive tests with both test function have been performed.

As performance measure, besides using the effective fitness from the final evaluation, we recorded the individual's distance from the origin in function f_1, with larger distance indicating that more flat hills were chosen, and the percentage of negative gene values in function f_2, with more negative values indicating that the hill was favored over the peak. All results reported are the averages over 10 runs with different random seeds.

Altogether, the following 10 approaches have been tested and compared:

1. a standard EA
2. a single evaluation with disturbed parameters
3. evaluate all individuals three times
4. evaluate the best individual another 4 times
5. evaluate the best 20% of the individuals another 2 times
6. evaluate the best 20% of the individuals another 4 times, the next 20% another 3 times etc.
7. use the current population to calculate a weighted fitness
8. same as 7., but with duplicate avoidance
9. use a memory of the last 5000 individuals to calculate weighted fitness
10. same as 9., but with duplicate avoidance.

The results are summarized in Table 1.

		Function 1		Function 2	
		best value	dist. origin	best value	genes < 0
1	standard EA	8.0798	7.692	8.9678	19%
2	single disturbed	8.8276	24.99	9.2360	69%
3	all 3x (avg)	9.5074	25.69	9.3260	79%
4	best 5x	8.4775	8.08	8.9064	50%
5	best10 3x	8.8699	10.37	9.4032	81%
6	decreasing	10.2495	13.31	9.8067	98%
7	pop weighted	10.2704	20.39	9.2730	47%
8	pop weighted, no doubles	10.2396	27.79	9.2541	50%
9	depot weighted	10.3613	39.46	9.8432	96%
10	depot weighted, no doubles	10.3233	41.95	9.7866	95%

Table 1. Results, each value is averaged over 10 runs

Figure 5 displays the tradeoff between cost (measured in the number of function evaluations) and performance (final effective quality achieved) of the different approaches for function f_1 (left) resp. function f_2 (right).

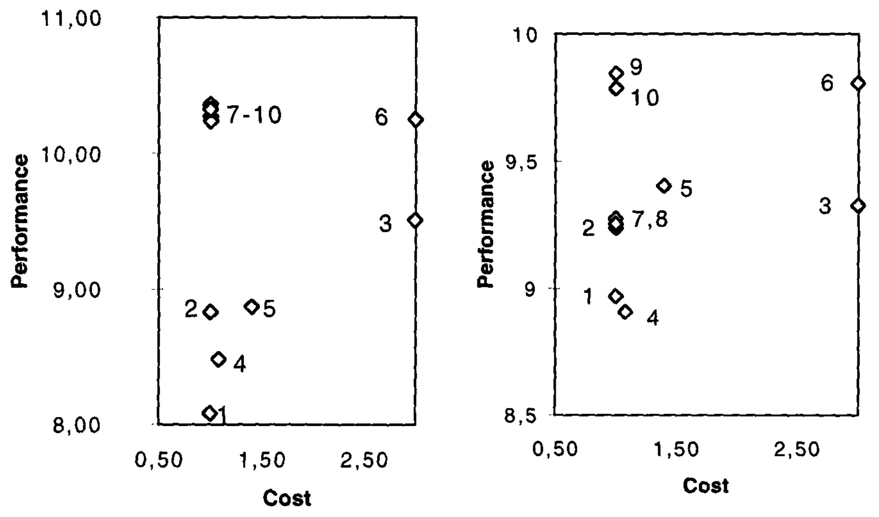

Fig. 5. Cost vs. performance comparison of different approaches on Function f_1 (left) resp. function f_2 (right)

From these preliminary results, the following conclusions may be drawn:

- Method 6 clearly outperforms Method 3, so the idea to distribute more function evaluations on the better individuals seems to give a significant advantage.
- the duplicate avoidance strategy does not seem to help much in the weighting approach (Methods 7 and 8 resp. 9 and 10 are almost equivalent).
- just disturbing the input for one evaluation (Method 2) performs comparatively well, at no additional cost.
- reevaluating just the best (Method 4) does not seem to be sufficient. At least the best fifth of the population (Method 5) has to be reevaluated to yield a noticeable effect.
- the idea to use individuals from previous evaluations in order to compute a weighted fitness vastly improves performance without requiring additional function evaluations. Of course the large memory base used in Method 9 and 10 incurs other computational overhead that has to be considered unless the function evaluations are the determining time factor. Anyway, it seems to be worth to investigate good strategies concerning which individuals to keep in the memory in order to obtain maximum benefit with minimum cost for weighting.

5 Conclusion

This paper pointed out the importance of creating robust solutions and compared a number of novel approaches to help the EA take into account the robustness of a solution.

The experiments clearly indicate that although the standard EA has an inherent tendency to favor robust solutions, this can be improved significantly by various modifications. Among the modifications proposed here, the idea to use individuals from previous evaluations to weight the fitness value show the greatest promise. Also, if re-evaluations are considered, more re-evaluations should be made of the fitter individuals than of poor individuals.

The paper raises a number of open issues, that are currently under investigation in our research group. Among those are:

- transferring these results to real world problems like e.g. scheduling
- designing a good strategy which and how many individuals to keep in the depot for future comparisons
- trying combinations of the proposed approaches, e.g. making extra evaluations for good individuals only when there are no old individuals in the neighborhood, or starting with single perturbed evaluations and later move towards the mean of several disturbed evaluations as fitness values.

References

1. A. N. Aizawa and B. W. Wah. Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, pages 97–122, 1994.
2. J. Michael Fitzpatrick and John J. Greffenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1988.
3. D. E. Goldberg. *Genetic Algorithms.* Addison-Wesley, 1989.
4. U. Hammel and T. Bäck. Evolution strategies on noisy functions, how to improve convergence properties. In Y. Davidor, H. P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, number 866 in LNCS. Springer, 1994.
5. U. Kohlmorgen, H. Schmeck, and K. Haase. Experiences with fine-grained parallel genetic algorithms. *Annals of Operations Research*, to appear.
6. M. McIlhagga, P. Husbands, and R. Ives. A comparison of search techniques on a wing-box optimisation problem. In H.-M. Voigt, editor, *Parallel Problem Solving from Nature 4*, number 1141 in LNCS, pages 614-623. Springer Verlag, 1996.
7. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, 3rd edition, 1996.
8. I. C. Parmee. Cluster-oriented genetic algorithms for the identification of high-performance regions of design spaces. In *EvCA96*, 1996.
9. C. R. Reeves. A genetic algorithm approach to stochastic flowshop sequencing. In *IEE Colloquium on Genetic Algorithms for Control and Systems Engineering*, number 1992/106 in Digest, pages 13/1–13/4. IEE, London, 1992.
10. R. Roy, I. C. Parmee, and G. Purchase. Integrating the genetic algorithm with the preliminary design of gas turbine blade cooling systems. In *ACEDC'96*, 1996.
11. A.V. Sebald and D.B. Fogel. Design of fault tolerant neural networks for pattern classification. In D.B. Fogel and W. Atmar, editors, *1st Annual Conf. on Evolutionary Programming*, pages 90–99, 1992.
12. S. Tsutsui and A. Ghosh. Genetic algorithms with a robust solution searching scheme. *IEEE Transactions on Evolutionary Computation*, 1(3):201–208, 1997.