A Scouting-Inspired Evolutionary Algorithm

Jeffrey O. Pfaffmann
Dept. of Computer Science
Lafayette College
Easton, PA 18042
pfaffmaj@cs.lafayette.edu

Konstantinos Bousmalis
Dept. of Computer Science
Lafayette College
Easton, PA 18042
bousmalk@cs.lafayette.edu

Silvano Colombano
Computation Sciences Division
NASA-Ames Research Center
MS 269-2
Moffett Field, CA 94035.
scolombano@mail.arc.nasa.gov

Abstract—**The goal of an Evolutionary Algorithm (EA) is to find the global optimum in a state space of potential solutions. But these systems can become trapped in local optima due to the EA having only generational information. Using the Scouting Algorithm (SA) it is suggested that a cross-generation memory mechanism can be added to modulate fitness relative to how well a region has previously been sampled. Thus, the goal is to allow the Scouting-inspired EA (SEA) to leave well explore regions to find the global optimum more quickly. It will be shown that the SEA does achieve this goal for the problem domain of nonlinear programming (NLP).**

I. Introduction

An Evolutionary Algorithm (EA) in its most basic form generates a population of potential solutions for a given task that are ranked, selected based on rank, and varied to produce a new population of potential solutions. Over time this system converges on a solution that can be the global optimum, but the system can also become trapped in one of many different local optima. One solution to this issue is to add a cross-generational memory mechanism for regulating how future populations search the state space based on previous experience. Thus, over time the system will accumulate domain knowledge that can be used for a more effective search.

This idea has been previously explored by Reynold's with Cultural Algorithms (CAs), where an anthropological view is taken on EAs [1, 2]. CAs maintain both population and cultural knowledge, with the population knowledge contributing to the cultural knowledge that is available for future generations. CAs use cultural knowledge to form generalizations about previously encountered phenomena, providing a belief space for facilitating a population's exploration of the state space. CAs were initially created to explore archaeological findings of developing cultures [3]. CAs have also been used to solve nonlinear constrained parameter optimization problems [4], which is the same problem domain used for these explorations.

The work presented here takes inspiration from the Scouting Algorithm (SA) to introduce a cross-generational memory mechanism to an Evolutionary Algorithm, creating the Scouting-inspired EA (or SEA). But unlike the CA's belief space, the SA confers the ability to determine what regions have already been explored and to what degree. This will allow the SEA to avoid being trapped in local optima and free the system to pursue the global optimum.

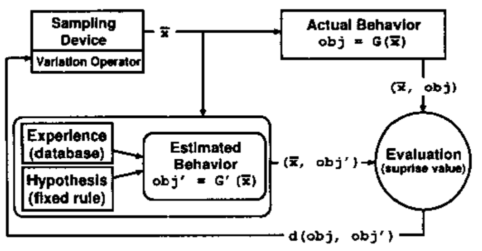


Figure 1. Surprise is the difference between actual and estimated behavior.

The SA was originally developed to perform automatic experimentation [5] where the resources to conduct experiments are limited or the system being examined is too complex for an exhaustive search to be performed. Thus, initially it was the SAs goal to efficiently search the state space by looking for interesting behavior that the experimentalist can later examine. It should be emphasized that this exploration is not to find optimal behavior, but instead to find interesting or "surprising" behavior.

To determine surprising behavior, the SA builds an experience database from previously performed experiments to calculate an estimated result for a yet unperformed experiment. The estimate is compared to the actual result, with the difference being the surprise value. Thus, an experiment where the difference between actual and estimated behavior is quite large will generate a great surprise. To calculate the estimate value, the k-nearest neighbors in the experimental space are averaged together, with each neighbor weighted by its distance to the current point being estimated relative to all other averaged neighbors.

The SA uses the surprise value in two ways, as the fitness value and to modulate the mutation operator. Allowing individuals with a great surprise to be predominately selected and mutated to remain in the region of previously great surprise. As a region becomes well sampled the surprise value will become smaller, allowing newly generated individuals to move farther from the current region until a new surprising result is located and the system focus moves appropriately.

The SA has been applied to variety of experimental systems [6, 7] where it has been shown to perform well. In the initial SA, population size and mutation strength was specified by parameters, while for automatic experimentation it would be more advantageous to allow these parameters be set dynamically based on the previous system behavior. Recently, the modulation of both aspects are explored in the Self-adaptive Scouting implementation [8], by modifying the initial SA in two ways. The first modification modulates the mutation operator so that the average surprise up to that point is used to scale the mutation strength relative to the difference between averaged and current surprise. The second modification regulates the number of offspring a parent has according to the performance of the offspring. It should be noted that this is an unusual concept, as typically parent fitness will influence the number of generated children.

Using Self-adaptive Scouting as inspiration, the SEA creates a mapping from the current surprise value to the normal distribution parameter σ for a gaussian distribution. The mapping is a simple inverse relationship, so that a high surprise value will produce a low σ to generate variates near zero. When the surprise value is low a high σ is generated and will produce a more even distribution of random variates. Thus, the enhancement will regulate the randomness exhibited by the EA, allowing the system to focus on a specific region when the underlying state space is poorly understood, and become more random when that local region is well understood.

II. The Problem Domain

The goal of the presented work is not to solve a specific problem, rather it is to determine the potential for enhanced performance by incorporating surprise, from the SA, into a traditional EA. Thus, a test-case generator was chosen that could produce a variety of problems of sufficient complexity to challenge both the EA and SEA. Additionally, two other desirable traits were looked for: the ability to generate problems with specified levels of complexity and the capability to change the problem dimensionality. These qualities were found in the TCG-2 package by Schmidt and Michalewicz [9], which is a C++ class for generating nonlinear constrained parameter optimization tasks (also known as nonlinear programming (NLP) problems).

Briefly, NLP problems are n-dimensional real-valued tasks consisting of an objective function that is constrained to produce a feasible solution space. Potential solutions are specified as a n-dimensional vector $\bar{x} = (x_1, \ldots, x_n) \in \Re^n$, which is bounded by a specified search space. For the TCG-2 package the search space is contained to $0 \le x_i \le 1$, where $1 \le i \le n$. For a more detailed account of NLP problems and their relationship to the TCG-2 see [9].

The fitness function used here follows Schmidt and Michalewicz's suggestion of a static penalty approach, as follows:

$$Fit(\bar{x}) = G(\bar{x}) - W \times CV(\bar{x}), \qquad (1)$$

where the objective function $G(\bar{x})$ is penalized with the constraint violation function $CV(\bar{x})$ value weighted by the penalty constant W. The given fitness function requires that $W > 0$, which is fixed to $W = 10$ for all experiments.

Different NLP problems were examined by manipulating the TCG-2 parameters and examining the resulting objective landscape. This process was facilitated by the capability of the TCG-2 package to provide the different landscapes as files containing the numerical information that later can be visualized.

The goal while manipulating the TCG-2 package parameters was to find a very rugged landscape with many regions of local optima that have a similar magnitude as the global optimum. The final parameter

TABLE I

TCG-2 PARAMETER SET.

problem dimensionality	:	n	$=2$
feasible components	:	m	$=10$
search space feasibility	:	ρ	$=0.5$
search space complexity	:	c	$=0$
active constraints at global optimum	:	a	$=0$
objective function peak count	:	p	$=50$
peak width	:	σ	$=0.1$
peak decay	:	α	$=0.1$
component min. distance	:	d	$=0.01$

choices are given in Table I producing the following landscapes for the objective function (Figure 2), constraint violation (Figure 3), and fitness function (Figure 4).

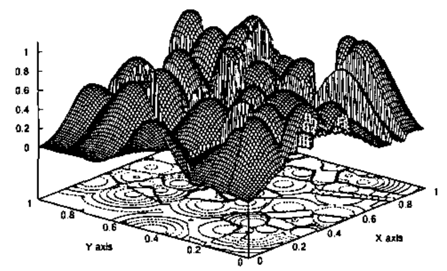


Figure 2. Objective function landscape $G(\bar{x})$.

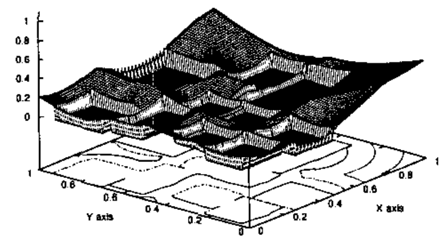


Figure 3. Constraint Violation function landscape $CV(\bar{x})$.

Schmidt and Michalewicz's paper provide a detailed description of the listed parameters (in Table I) and their effect on the resulting NLP problem. One notable parameter is the peak count count that directly translates into the number of peaks in the actual objective function. The global optimum peak was generated by the TCG-2 package near the center of the search space, $\bar{x}_g = (0.480704, 0.495582)$, with a peak height of 1. Within the search space there are 5 local optima with a peak height of 0.9 and greater, with the greatest peak within this group at 0.982. For peaks that range: from 0.8 to 0.9 there are 5 and from 0.7 to 0.8 there are 6. The active constraints on the parameters at the global optimum specify whether the global fitness point is placed in a feasible area of the fitness landscape or near the edge of such an area. For the problem used here, the global optimum is placed in a feasible area and not on the edge such an area.

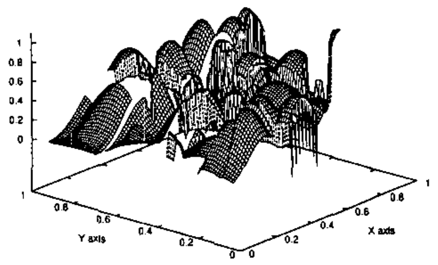


Figure 4. Fitness function landscape $Fit(\bar{x})$.

III. Scouting-inspired Evolutionary Algorithm

As indicated previously the design of the EA took a traditional approach, capturing the essence of the evolutionary optimization technique. An initial population of vectors are generated, each vector is evaluated using the fitness function (Equation 1), these individuals are selected using *roulette wheel* sampling, and each selected individual is mutated by varying a single gene within the vector that represents the individual. Individuals are mutated using a gaussian distribution with the standard deviation parameter σ set to provide a specific range of random variates biased around a median of 0. The process of variation and selection continues until a specific generation is reached.

From one generation to the next there is a total population turnover, allowing the EA the maximum number of new samples each generation. To ensure that convergence is achieved, in the presence of a new population each generation, roulette wheel sampling is used. This type of sampling will in general retain the best performing individuals to contribute their genetic material to the next generation while still allowing for lesser performers to also contribute.

Pseudocode for the Scouting-inspired EA is given below with the first part indicating the parameters and the second part providing the code.

```
PARAMETERS:
    $Pop$ = population size
    $n$ = problem dimensionality; number of genes
    $\sigma_{min}$ = minimum standard deviation
    $\sigma_{max}$ = maximum standard deviation
    $gen_{max}$ = maximum number of generations
BODY:
01 initialize random number generator.
02 initialize scouting database.
03 generate initial population : $\bar{x}_p = \{\bar{x}_1, \ldots, \bar{x}_{Pop}\}$
04 $gen = 0$
05 repeat
06     for $i = 1$ to $Pop$ do
07         calculate fitness : $fit_i = Fit(\bar{x}_i)$
08         calculate estimate : $fit'_i = Fit'(\bar{x}_i)$
09         calculate surprise : $sup_i = |fit'_i - fit_i|$
10         store fitness value in experience database
11     end for
12     for $i = 1$ to $Pop$ do
13         select $\bar{x}_p$ using roulette selection
14         select gene to mutate : $x_{p,g}$ for $g = 1$ to $n$
15         calculate mutation modulator :
               $\sigma_p = \sigma_{max} - (sup_p \times (\sigma_{max} - \sigma_{min}))$
16         mutate gene : $x_{p,g} = x_{p,g} + gaussianRand(\sigma_p)$
17     end for
18     increment $gen$ by 1
19 until $gen = gen_{max}$
```

Conceptually, the surprise calculation processes in parallel to the EA by looking over the EA's shoulder and helping to guide the search behavior without changing the basic EA functionality. Examining the previous pseudocode the parallel functionality is contained in lines 8–10, 15, and 16. The first for-loop (starting line 6) calculates the fitness, evaluates the current population by determining surprise from previous fitness values, and stores the actual behavior in the scouting database. The calculation of the surprise value (sup_i) uses the actual (fit_i) and estimated (fit'_i) fitness, with the estimate generated by the weighted k-nearest neighbor technique described previously for the SA.

Once the fitness (fit_i) and surprise (sup_i) values have been generated for each individual of the current generation, the algorithm enters the variation/selection-loop (lines 12–17). In the second for-loop the EA enhancement is seen only on lines 15 and 16 where the standard deviation (σ_p) is calculated and applied for each parent to be mutated.

Integrating the surprise calculation in this way allows the EA to have an additional perspective on the fitness space, providing a finer granularity of control. Thus, not only is the fitness used to select an individual to be varied, the information from all previous samples can be used to further refine the search mechanism by regulating the variation. So a high surprise value indicates an unexplored region where children should be varied only slightly. While a low surprise value indicates a well explored region and the resulting children should be varied greatly so that new regions can be searched out. It should be noted, variates producing a mutation that falls outside of the range $0 \leq x_{p,g} \leq 1$ are rejected and a new variate is generated as a replacement. This has the effect of rejecting a larger number of variates when σ is high. In future implementations the variate can be scaled to reduce the rejection frequency, which will also reduce the quantity of calls to the random number generator.

IV. System Performance

Since the focus of the presented work is to perform a comparison of an EA with and without the surprise modulated variation enhancement, the choice of random generators was critical. Thus, an established numerical package was selected, the GNU Scientific Library (gsl) [10], and from that three high quality random number generators were chosen. The first generator was Matsumoto and Nishimura's "Mersenne Twister" [11], which is known to generate long periods with low correlation. The second and third generators use Lüscher's ranlux algorithm, or the "luxury random number generator" [12]. The first of these generators is the original implementation, while the second is a double precision variant that runs a quarter times slower. Like the first generator, the ranlux generators create very long periods and produce provably de-correlated numbers at different levels of randomness. For the three generators 50 different random seeds were generated using dice, further ensuring a high level of randomness, to produce 150 different random sequences for experimentation. This is to indicate that during an experiment only one of the three different generators is used as part of the $gaussianRand(\sigma_p)$ function (see line 16 of the previous algorithm). The goal of using the three random generators is to eliminate any biases that might occur from the different techniques and implementations.

To determine what population size should be used, again the work by Schmidt and Michalewicz was referred to. The TCG-2 package was initially evaluated using an evolutionary algorithm to empirically explore arbitrarily derived problem spaces generated by the package for given parameter sets. In this work, Schmidt and Michalewicz used a population size of 100 individuals with a very high problem dimensionality ($n = 30$). Additionally, their evolutionary algorithm used a crossover mutation operator, which is not true of the EA used here.

From Schmidt and Michalewicz's studies, it was determined that the three most important parameters were dimensionality, number of peaks, and peak width (affecting the peak slope). The work here uses both a large number of peaks and a small peak width or 50 peaks with a width of 0.1. In general, Schmidt and Michalewicz used 30 peaks at a width of 0.5, while varying a single parameter to examine the system behavior along that dimension in the parameter space. In this regard, our chosen problem is more complex than theirs, where our task is less complex in the dimensionality. As mentioned above, Schmidt and Michalewicz used 30 dimensions compared to our 2 dimensions, with the motivation being the ability to visualize the system on the landscape. In future work the dimensionality will be increased to determine if the SEA functionality similarly scales.

Since the problem here is less complex, in terms of dimensionality, the size of our populations were chosen to be smaller for the majority of the experiments, using population sizes of 10, 20, 30, and 100. As seen in Figure 5-A the averaged EA fitness values over 150 experiments for 5000 generations produces high levels of fitness that scale as the number of individuals increase. But when compared to the SEA (Figure 5-B) the EA (Figure 5-A) performs poorly, as the SEA consistently achieves higher fitnesses more quickly. Given 5000 generations all SEA experiments reached a fitness level of 0.99 or greater, seen in Figure 5-B.

Figure 6 plots the number of individual experiments to reach a level of 0.99 or higher at a given generation out of a total 150 experiments, for both the EA (Figure 6-A) and SEA (Figure 6-B). The fitness level of .99 is significant because it indicates that the regions containing the global optimum fitness has been reached (as the next highest peak is 0.982). Figure 6-B shows the SEA achieved a fitness of 0.99 or greater within 5000 generations for all 150

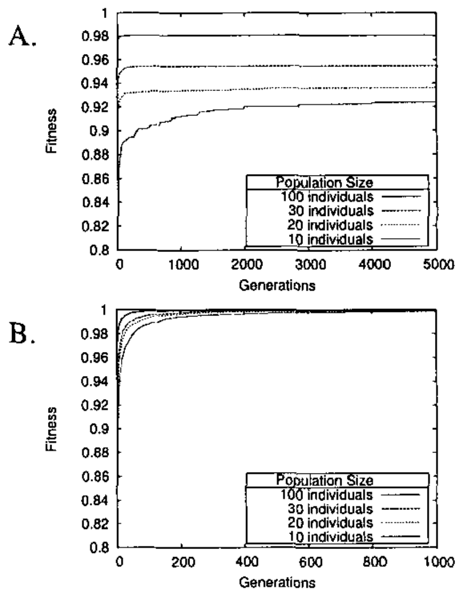


Figure 5. Average fitness of 150 experiments for different population sizes, showing the: A. EA and B. SEA. Note, only 1000 generations of the SEA results were plotted for increased graph clarity.

experiments, indicating that the region containing the global optimum was always reached. While Figure 6-A indicates a lesser but still significant number of EA experiments reached this region. This advantageous behavior can be attributed to the ability of the SEA to perform a much broader search of all possible local optima.

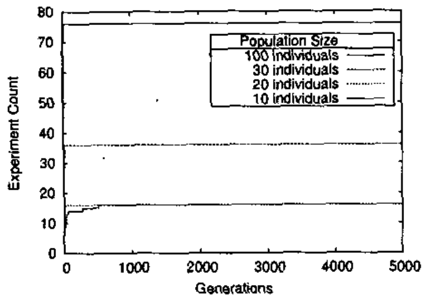
TABLE II

EXPERIMENTS THAT FOUND GLOBAL OPTIMUM.

Population	Found Optimum Count	
Size	EA	SEA
100	60	2
30	13	0
20	6	1
10	3	1

But as can be seen in Table II that shows the number of experiments that found the global optimum, the EA exceeded the capability of the SEA to find the optimum solution. This indicates that the SEA has lost its ability to converge, which is an important aspect of evolutionary algorithms. This convergence

A.



B.

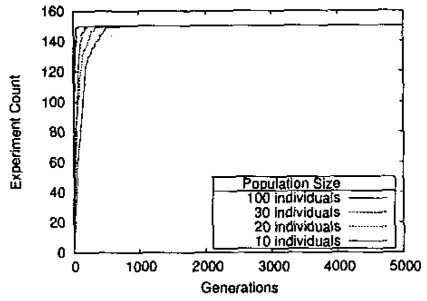
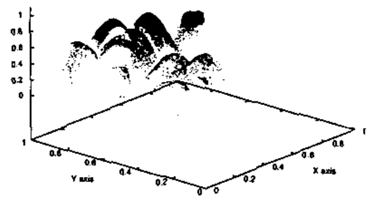


Figure 6. Count of experiments achieving a fitness of 0.99 or greater: A. showing the EA and B. showing the SEA.

A.



B.

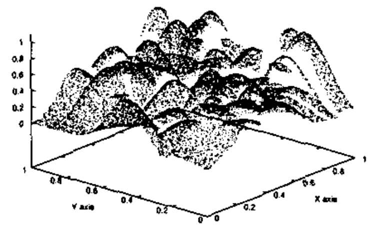


Figure 7. Typical state space sampling patterns for an EA (A) and SEA (B).

loss is due to the effect of surprise always pushing the system from a well explored region to another less explored region, based on an estimate, and will rarely find the optimum. However, the SEA always finds the region near optimum for the system explored here.

This behavior can be visualized by plotting how the EA and SEA sample their state space over the 5000 generations, shown in Figure 7. The EA is an inherent local search mechanism that examines where it currently is, and thus more slowly explores the larger state space with the possibility of well explored regions to continued to be explored. The SEA also retains this local search aspect, but the local search is modulated by the effects of surprise that is continuously pushing the system into more interesting regions. Figure 7-A shows a single EA experimental state space sampling, which is clearly localized to specific regions due to the issue of local optima trapping. While the SEA (Figure 7-B) can more freely move throughout the space, paying the price of losing the ability to converge.

V. Conclusion

Often when Evolutionary Algorithms (EAs) are discussed in relation to a random search it is in terms of performance, in other words, can the EA outperform a purely random search. In the case of this work it is better to think of the EA as a constrained random search, with the two search mechanisms forming a continuum. Obviously, a purely random search is a waste of resources, but so is examining regions of the fitness landscape that have been throughly explored particularly if that region only contains a local optimum and the EA is trapped. The question becomes how can the EA be made less constrained, or can the level of constraint be modulated, allowing for the more effective use of randomness.

Presented is a potential solution to how the level of randomness can be modulated by taking inspiration from the Scouting Algorithm (SA) to enhance an EA, producing the SEA. This enhancement exists purely to determine when the SEA has explored a space to such a level that there is little new, or surprising, information to discover. This information is derived from previously generated samples to compute an estimate about a given location in the state space. This estimate is compared to the actual behavior to determine a level of surprise, using the difference. Great differences produce large surprises, indicating that the system should remain in that region. The surprise level is used to modulate the gaussian random variate distribution so that when the SEA is in a region that little is known about a high surprise is generated, causing the variate to more likely be generated near 0. If the surprise is low, this causes a more even variate distribution, allowing children to be mutated so they are more likely to leave a well understood region.

It has been shown that this enhancement both quickly and consistently moves the SEA into regions containing solutions that are near the global optimum, while at the same time sacrificing convergence. The traditional EA retains the ability to converge, at the cost of possibly being trapped in local optima. Both systems have their advantages and clearly the next step in this work is to resolve this issue by modulating the effects of the scouting surprise value in relation to the current fitness level. The idea is to reduce the effect of surprise if fitness is high. The opposite will be performed if the fitness is low. Another avenue of exploration is the mapping from the surprise value to the standard deviation parameter. Here a simple mapping is used, but due to the nature of the gaussian distribution the effect of changing the standard deviation parameter is a non-linear relationship and it is not clear that this is the most effective mapping. The third and most challenging avenue of exploration is to see whether the behavior of the SEA can be scaled to solve high dimensional problems.

Overall, this work shows initial steps to providing an enhancement to many evolutionary algorithms that does not drastically change the essential spirit of this optimization technique.

ACKNOWLEDGMENT

This research was conducted with support from the Lafayette College Excel Scholar program. Also, the authors would like to thank Schmidt and Michalewicz for the contribution of the TCG-2 package, allowing exploration of the SEA to be conducted in a timely fashion.

REFERENCES

[1] R. G. Reynolds, "An introduction to cultural algorithms," in *Evolutionary Programming III: Proceedings of the Third Annual Conference, February 24-26 San Diego, CA*, L. J. Fogel and A. V. Sebald, Eds.River Edge, NJ: World Scientific Publishing Co., Inc., 1994, pp. 131–139.

[2] R. G. Reynolds, "Cultural algorithms: Theory and applications," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, England: McGraw-Hill, 1999, chapter 24, pp. 367–377.

[3] R. G. Reynolds, "The impact of raiding on settlement patterns in the northern valley of oaxaca: An approach using decision trees," in *Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes*, Santa Fe Institute Studies in the Sciences of Complexity, T. Kohler and G. Gummermann, Eds. New York, NY: Oxford University Press, Inc., 2000, pp. 251–274.

[4] R. G. Reynolds and X. Jin, "Using knowledge-based evolutionary computation to solve non-linear optimization problems: A cultural algorithm approach," in *Proceedings of the Congress of Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds. 1999, vol. 3, pp. 1672–1678.

[5] J. O. Pfaffmann and K.-P. Zauner, "Scouting context-sensitive components," in *The Third NASA/DoD Workshop on Evolvable Hardware EH-2001*, D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, Eds. Los Alamitos, CA: IEEE Computer Society Press, 2001, pp. 14–20.

[6] N. Matsumaru, S. Colombano, and K.-P. Zauner, "Scouting enzyme behavior" in *Proceedings of the 2002 World Congress on Computational Intelligence, May 12–17 Honolulu, Hawaii*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, Eds. Piscataway, NJ: IEEE Press, 2002, pp. CEC 19–24.

[7] F. Centler, P. Dittrich, L. Ku, N. Matsumaru, J. Pfaffmann, and K.-P. Zauner, "Artificial life as an aid to astrobiology: Testing life seeking techniques," in *Advances in Artificial Life — Proceedings of the 7th European on Artificial Life, ECAL 2003, September 14–17 Dortmund, Germany*, vol. 2801 of *Lecture Notes in Computer Science*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, Eds. Heidelberg, Germany: Springer-Verlag, 2003, pp. 31–40.

[8] N. Matsumaru, F. Centler, K.-P. Zauner, and P. Dittrich, "Self-adaptive scouting—autonomous experimentation for systems biology," in *Proceedings of 2nd European Workshop on Evolutionary Bioinformatics, EvoBIO 2004, Lecture Notes in Computer Science*. Heidelberg, Germany: Springer-Verlag, 2004.

[9] M. Schmidt and Z. Michalewicz, "Test-case generator TCG-2 for nonlinear parameter optimization," in *Parallel Problem Solving from Nature – PPSN VI, Proceedings of the 6th International Conference, September 18-20 Paris, France*, vol. 1917 of *Lecture Notes in Computer Science*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervós, and H.-P. Schwefel, Eds. Heidelberg, Germany: Springer-Verlag, 2000, pp. 539–548.

[10] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi, *GNU Scientific Library Reference Manual*. Bristol, UK: Network Theory Ltd., 2003.

[11] M. Matsumoto and T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, vol. 8, num. 1, pp. 3–30, 1998.

[12] M. Lüscher, "A portable high-quality random number generator for lattice field theory calculations," *Computer Physics Communications*, vol. 79, pp. 100–110, 1994.