Improving Evolutionary Algorithms with Scouting

Konstantinos Bousmalis[1], Gillian M. Hayes[2], and Jeffrey O. Pfaffmann[3]

[1] Department of Informatics, The University of Edinburgh, Edinburgh, UK
K.Bousmalis@sms.ed.ac.uk

[2] Institute of Perception, Action and Behaviour(IPAB), Department of Informatics, The University of Edinburgh, Edinburgh, UK
gmh@inf.ed.ac.uk

[3] Department of Computer Science, Lafayette College, Easton, PA 18042, USA
pfaffmaj@cs.lafayette.edu

Abstract. The goal of an Evolutionary Algorithm(EA) is to find the optimal solution to a given problem by evolving a set of initial potential solutions. When the problem is multi-modal, an EA will often become trapped in a suboptimal solution(premature convergence). The Scouting-Inspired Evolutionary Algorithm(SEA) is a relatively new technique that avoids premature convergence by determining whether a subspace has been explored sufficiently, and, if so, directing the search towards other parts of the system. Previous work has only focused on EAs with point mutation operators and standard selection techniques. This paper examines the effect of scouting on EA configurations that, among others, use crossovers and the Fitness-Uniform Selection Scheme(FUSS), a selection method that was specifically designed as means to avoid premature convergence. We will experiment with a variety of problems and show that scouting significantly improves the performance of all EA configurations presented.

1 Introduction

1.1 Evolutionary Algorithms

Evolutionary Algorithms are a family of optimization techniques that attempt to solve a given problem by evolving a set of solutions. A typical EA randomly initializes a population of potential solutions-individuals that are subsequently *(a)* assigned a measure of merit(fitness value), *(b)* selected for reproduction based on that merit, and *(c)* varied via crossover(exchange of genes), mutation, and deletion, to produce a new generation of individuals. The cycle of fitness assignment, selection, reproduction and deletion usually continues for a preset number of generations, or until the global optimum of a certain objective function is reached. The goal is to find this global optimum, but often, a typical EA gets trapped in local optima, a problem this paper suggests a solution for.

Generation-based EAs replace the entire population in each generation, whereas steady-state EAs replace only one or two individuals. The Fitness-Uniform Selection Scheme, which will be one of the focal points of the work presented here, has been tested only on steady-state EAs.[5] This work focuses on the use of generation-based EAs, with future work potentially expanding into the area of steady-state EAs.

1.2 Scouting-Inspired Evolutionary Algorithm: Previous Work

Scouting was originally introduced as a mechanism for automated exploration of complex phenomena, using a conservative number of samples. It uses an evolutionary technique, which, instead of focusing on finding an optimum, searches for regions of the search space that exhibit "surprising" behavior.[2] "Surprise" is defined as the difference between an estimated sampling result and the actual returned value. It was apparent from the introduction of the technique, that an experience database, namely a database of observations, could create a simple model of a system.

Figure 1 provides an illustration of a Scouting Algorithm(SA), which consists of an Evolution Strategy(ES) that evaluates individuals, creates new generations solely through mutation, and selects only the best individual for reproduction.[2] As observations are made on a given system, the results are saved in an experience database. A characteristic of an ES is self-adaptation of the mutation strength.[8] The way the SA achieves that is by adapting the Gaussian distribution of the mutation range based on the surprise value of each individual. As already mentioned, surprise is defined as the absolute difference of the estimated and the actual fitness values. The estimate is calculated via a weighted k-nearest neighbor algorithm, which uses the experiences stored in the database. When the surprise is high, the search continues with a small mutation range and vice versa. Hence, the technique will explore a subspace until the results gained are no longer "surprising," according to the experience the search has already had in the system at hand. Scouting has also been the focus of two more recent papers, which presented an SA as means to automated experimentation in biological systems. [3][4]

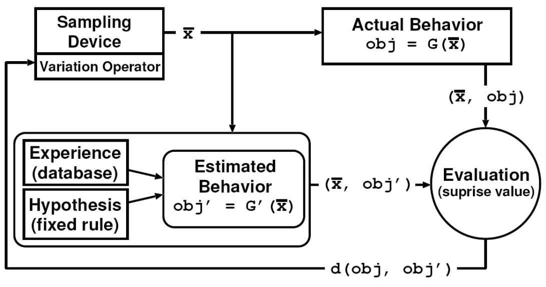


Fig. 1. A Scouting Algorithm varies individuals based on their surprise value. $\bar{x}$ is a new individual, $G(\bar{x})$ the actual, and $G'(\bar{x})$ the estimated objective values. d is their difference.[1]

Pfaffmann et al.[1] defined the Scouting-Inspired Evolutionary Algorithm as an evolutionary technique that uses scouting to model a given search space, and provide a simple way to avoid premature convergence in deceptive and multi-modal problems. It was defined as an EA that uses the Roulette Wheel selection scheme, scouting-driven mutation as the only genetic operator, and deletion of the entire population in each generation (generation-based EA). For each mutation operation, the mutation strength is randomly chosen from a Gaussian distribution, the standard deviation(σ) of which changes based on the surprise value of the parent. Given minimum and maximum standard deviation, σ_{min} and σ_{max}, the active σ for a given parent with surprise s_{ind}, scaled to [0,1], is chosen by the modulator function

$$\sigma(s_{ind}) = \sigma_{max} - s_{ind} \times (\sigma_{max} - \sigma_{min}) \ . \tag{1}$$

As one can easily determine in (1), when the surprise is minimal, σ approaches σ_{max}, whereas when the surprise is close to its maximum value of one, it approaches σ_{min}.

1.3 Fitness-Uniform Selection Scheme (FUSS)

The SEA was designed to promote genetic diversity in the overall pool of individuals via regulating mutation. It was hypothesized that this would help find fitter solutions faster, in multi-modal and deceptive problems. The Fitness-Uniform Selection Scheme(FUSS), a relatively new selection method, has also been proved successful in maintaining genetic diversity, and helping an EA perform better in such problems.[5]

The way FUSS achieves high genetic diversity, is by allowing only a small number of fitness-similar individuals in a population. Similarity between two individual fitness values is simply defined as their absolute difference. This particular selection technique works in two stages. Firstly, a random fitness value f is selected uniformly from the interval $[f_{min}, f_{max}]$, where f_{min} and f_{max} are the minimum and maximum fitness values of the given population. Subsequently, the individual with fitness nearest to f is selected. FUSS, as opposed to traditional selection techniques, does not have an inherent goal to achieve populations with the highest average fitness possible. Hutter et al.[5] show that by focusing selection pressure towards less represented fitness values, premature convergence can be avoided and the path towards fitter solutions can remain open.

2 A Closer Look to the SEA

2.1 A New Standard Deviation(σ) Modulator

Reproduction and analysis of the results presented in [1] showed that the vast majority of all surprise values are in the lowest 10% of the surprise range. Table 1 shows the number of individuals in each surprise level, from the $29,950,012$

Table 1. The number of individuals in each surprise level throughout a set of experiments. (10 individuals per generation, 5000 generations per experiment)

Surprise Level	Number of Individuals	Percentage(%)
[0, 0.1]	29,064,234	97.04
(0.1, 0.2]	479,374	1.6
(0.2, 0.3]	211,509	0.71
(0.3, 0.4]	146,481	0.49
(0.4, 0.5]	41,365	0.14
(0.5, 0.6]	5,953	0.02
(0.6, 0.7]	911	0.003
(0.7, 0.8]	128	0.0004
(0.8, 0.9]	48	0.0001
(0.9, 1]	8	0.00002

individuals created throughout a set of experiments that used scouting-driven mutation.

As one can see in Fig.2, the linear mapping from surprise to standard deviation in (1) makes the SEA behave similarly to a traditional EA with a mutation standard deviation of σ_{max}. In order to achieve a fairer comparison with an EA that uses σ_{min}, we ought to regulate the effect of scouting such that when $s_{ind} = 0$, $\sigma = \sigma_{max}$, when $s_{ind} = 0.10$, σ approaches σ_{min}, and slowly decreases, to equal σ_{min} when $s_{ind} = 1$. We will therefore introduce a new standard deviation modulator, which, as seen in Fig.2, meets the requirements set above.

$$\sigma(s_{ind}) = \sigma_{max} - (s_{ind})^{\gamma} \times (\sigma_{max} - \sigma_{min}) \ , \quad 0 < \gamma \leq 1 \ . \qquad (2)$$

Based on Table 1, $\gamma = 0.01305$ is the optimal value for the new parameter. The reason is that when $s_{ind} = 0.1$, $s_{ind}^{0.01305} = 0.9704$, namely the technique will use 97.04% of the σ range on 97.04% of the individuals. Note that (1) is a case of (2) for $\gamma = 1$.

2.2 Crossovers

A traditional EA usually uses both crossovers and mutation as its genetic operators. As mentioned earlier, previous work has focused on EAs that only use point mutation. We want to show that scouting can improve an EA, and any evidence for this claim would not be complete without the examination of its effect on a configuration that uses crossovers.

In this paper, we examine the effect scouting has on EA configurations with and without single-point crossovers. Crossover and mutation are applied with certain crossover and mutation rates. When an EA configuration uses the crossover operator, and crossover is not chosen for a given pair of parents, mutation is always applied, in order to ensure that the parents are not cloned.

The introduction of crossover does not significantly affect the way scouting controls mutation standard deviation. When scouting-driven mutation is used

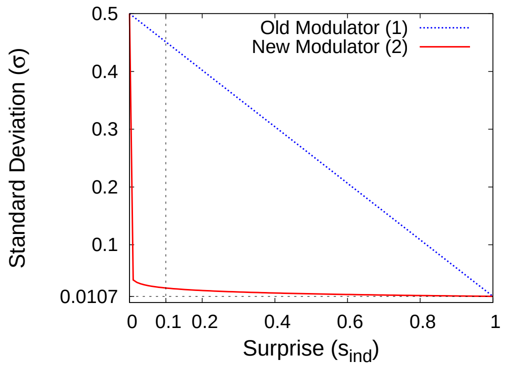


Fig. 2. Standard deviation vs. surprise using the old (1) and new (2) modulators with $\gamma = 0.01305$, $\sigma_{min} = 0.0107$ and $\sigma_{max} = 0.5$. The majority of the individuals created throughout an experiment has surprise $s_{ind} \leq 0.1$.

together with the crossover operator, there arises the issue of determining a s_{ind} to use in (2), without calculating fitness and surprise for intermediate individuals. In the techniques presented here, we use the average surprise values of the two parents used for the crossover.

We will therefore examine the effect of scouting on EAs with crossover and show that the improvement incurred is equivalent to the one observed in a mutation-only EA.

2.3 FUSS and the SEA

Another focal point of this paper is FUSS, mainly because of its similarities to the SEA. Firstly, both techniques aim to avoid premature convergence, by maintaining genetic diversity. Secondly, they have both been designed for deceptive and multi-modal domains, where traditional evolutionary techniques tend to become trapped in suboptimal solutions. Finally, the goal of both FUSS and the SEA is not to explicitly find higher fitness individuals, but to avoid premature convergence allowing for greater fitness levels to be achieved. The first technique succeeds by favoring the least-represented individuals in a population, whereas the latter succeeds by directing the search towards surprising parts of the search space of interest via mutation strength regulation.

One of the goals of this paper is to examine the effect a combination of the two techniques could have on a traditional EA when attempting to solve deceptive and multi-modal problems. We will show that scouting can improve an EA that also uses FUSS, and that even higher fitness levels could potentially be reached when these two techniques work together.

3 Implementation

The EA framework was written in C/C++. The random number generators used are the "Mersenne Twister"[9], and the two versions of the "luxury random number generator" algorithm[10], which are all included in the GNU Scientific Library (gsl)[7]. The problems used to test and compare the techniques were generated by Schmidt and Michalewicz's TCG-2 test-case generator.[6]

3.1 Test Cases and the TCG-2 Package

The problem domain and the reasons for using TCG-2 are fully outlined in [1]. Briefly, TCG-2 is a very configurable C++ software package that can generate a vast variety of nonlinear constrained parameter optimization problems with different levels of complexity. The fitness function used here follows the suggestion by Schmidt and Michalewicz[6] for a static penalty approach:

$$Fit(\bar{x}) = G(\bar{x}) - W \times CV(\bar{x}) \ , \tag{3}$$

where $G(\bar{x})$ is the objective function, W the static penalty, and $CV(\bar{x})$ is the constraint violation function for the given test case.

The test cases generated were varied in the number and width of peaks, in an attempt to get a better idea of the kind of problems the SEA is particularly effective at. We will refer to the width of peaks as σ_{peak} to differentiate it from the standard deviation σ used in mutation. We used the TCG-2 parameters shown in Table 2, and created the test landscapes by setting the parameter σ_{peak} to 0.02, 0.1 and 0.2, and the number of peaks p to 10, 50, 100 and 150. Consequently, twelve different two-dimensional test-case problems were created, as illustrated in Fig. 3.

The number of dimensions was mainly kept to two, for visualization purposes and easier understanding of the behavior of the new techniques. After analysis of the results on the two-dimensional problems, a set of experiments was also run in three dimensions with the parameters outlined in Table 2, for the case of 10 peaks and $\sigma_{peak} = 0.02$, as an example of a higher-dimension, multi-modal and deceptive problem.

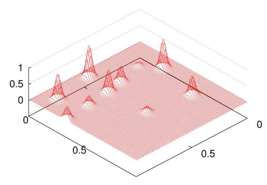
3.2 The Evolutionary Algorithm

The implemented Evolutionary Algorithm follows the guidelines provided in the introductory SEA paper, with the necessary changes for the goals of this paper. Selection for parenthood uses either Roulette Wheel or FUSS, and the parents are varied via crossover and/or mutation to create the next generation of individuals. The process loops for a set number of generations—5000 for all experiments presented here.

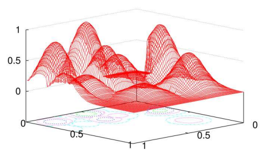
Mutation varies individual genes with a Gaussian distribution centered around mean $\mu = 0$. The number of genes changed for each individual during mutation is

Table 2. TCG-2 parameters for the two-dimensional experiments.

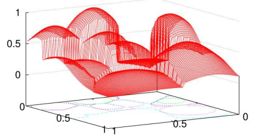
Number of dimensions (n)	: 2
Number of feasible components (m)	: 10
Search space feasibility (ρ)	: 0.5
Search space complexity (c)	: 0
Active constraints at global optimum (a)	: 0
Number of peaks (p)	: 10, 50, 100, 150
Peak width (σ)	: 0.02, 0.1, 0.2
Peak decay (α)	: 0.1
Component minimum distance (d)	: 0.01
Penalty (W)	: 10



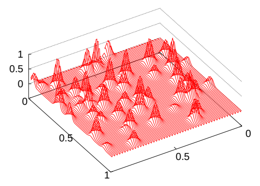
(a) 10 peaks, σ_{peak}=0.02



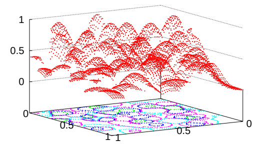
(b) 10 peaks, σ_{peak}=0.1



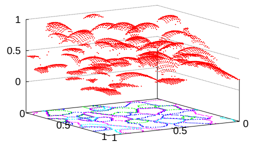
(c) 10 peaks, σ_{peak}=0.2



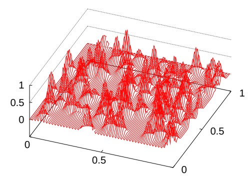
(d) 50 peaks, σ_{peak}=0.02



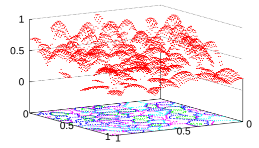
(e) 50 peaks, σ_{peak}=0.1



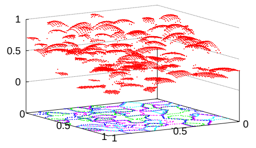
(f) 50 peaks, σ_{peak}=0.2



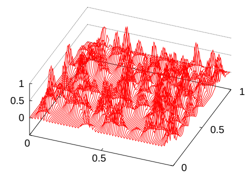
(g) 100 peaks, σ_{peak}=0.02



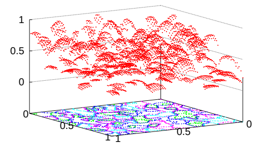
(h) 100 peaks, σ_{peak}=0.1



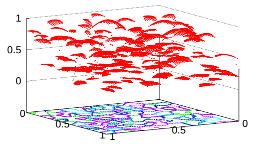
(i) 100 peaks, σ_{peak}=0.2



(j) 150 peaks, σ_{peak}=0.02



(k) 150 peaks, σ_{peak}=0.1



(l) 150 peaks, σ_{peak}=0.2

Fig. 3. The objective functions for each two-dimensional test case

configurable, but all genes of an individual are varied during mutation in the experiments presented. New individuals that are created out of range are rejected, and mutation recreates individuals, until one is within bounds.

Both mutation and crossover rates are set to 0.5 for all experiments, due to the same configuration in the FUSS paper [5]. This is also in accordance with the basis of the original SA, the traditional ES, which uses crossover and mutation with equal importance.[8]

Minimum standard deviation for scouting-driven mutation is set equal to the standard deviation for Random Mutation, $\sigma_{min} = 0.0107$, whereas the maximum one is $\sigma_{max} = 0.5$. The constant parameter γ is set, for reasons explained in the previous section, to $\gamma = 0.01305$ for all experiments.

4 Results and Analysis

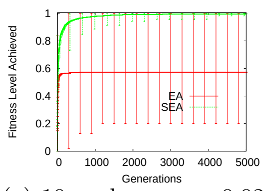
4.1 Two-dimensional test cases

We ran all experiments with eight different EA configurations, as shown in Table 3, for each of the twelve different two-dimensional test cases generated by TCG-2. The experiments were repeated 150 times per set (50 seeds with 3 random-number generator techniques, as suggested in [1]). Different experiment sets were run for a population size of 10, 20, 30 and 100 individuals per generation and it was found, similarly to the results presented in [1], that the results scaled accordingly, as the population size increased.

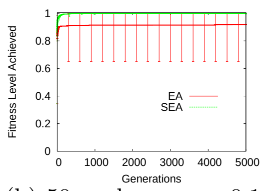
Table 3. The eight different EA configurations.

Configuration	Selection	Mutation	Crossover
EA	Roulette Wheel	Random	None
SEA	Roulette Wheel	Scouting-driven	None
EAC	Roulette Wheel	Random	Single-point
SEAC	Roulette Wheel	Scouting-driven	Single-point
EAF	FUSS	Random	None
SEAF	FUSS	Scouting-driven	None
EAFC	FUSS	Random	Single-point
SEAFC	FUSS	Scouting-driven	Single-point

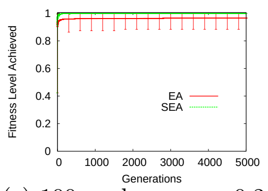
The results obtained clearly show that scouting improved all EA configurations used. The more deceptive the problem, the bigger the improvement of the performance exhibited by the SEAxx methods. More specifically, it was observed that there was a larger average improvement of performance, as the landscape contained less and narrower peaks. Figures 4 and 5 display the fitness level reached by generation, for, due to lack of space, only a few representative sets of experiments, and only for population size of 20 individuals. As mentioned above, however, the results scale accordingly as we change the population size. It is particularly important to note, that in all sets of experiments, the worst performance of an SEAxx was always better than the average performance of the equivalent EAxx, and very close to the global optimum, as one can clearly see in the figures mentioned above.



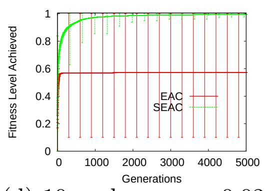
(a) 10 peaks, σ_{peak}=0.02



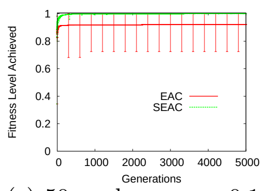
(b) 50 peaks, σ_{peak}=0.1



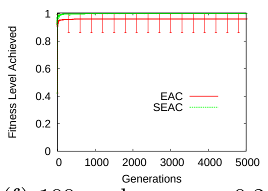
(c) 100 peaks, σ_{peak}=0.2



(d) 10 peaks, σ_{peak}=0.02



(e) 50 peaks, σ_{peak}=0.1



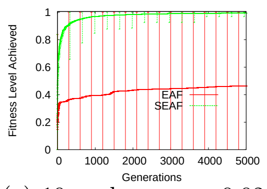
(f) 100 peaks, σ_{peak}=0.2

Fig. 4. Fitness level achieved per generation by EA vs SEA(first row), and by EAC vs SEAC(second row), for few of the experiment sets. It is clear that as the peaks and their width decrease, the improvement scouting achieves is more impressive.

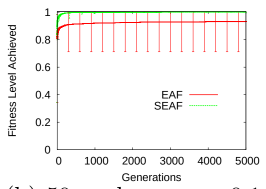
SEAC had similar performance with SEA, as it was originally hypothesized. The crossovers did not seem to have a negative effect in the way scouting affected the EA, and the improvement of the equivalent simple EA configurations was similar. (see Fig.4)

Scouting has also managed to significantly enhance the techniques that used FUSS. The effect of scouting overpowered the effect of FUSS (see Fig.5), the performance of which was not particularly impressive in these problems and EA configurations. For example, EAF and EAFC had the lowest average and worst fitness level reached for the test case of 10 peaks and $\sigma_{peak} = 0.02$. While it did enhance the performance of the traditional EA, in certain test cases, the improvement was not significant. However, it is important to note that FUSS was originally designed for steady-state EAs and for larger population sizes.

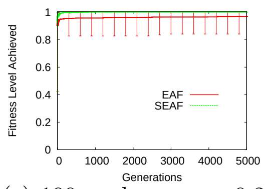
The most impressive achievement of scouting was exhibited in the test case with 10 peaks and $\sigma_{peak} = 0.02$. The landscape can be imagined as an almost completely flat surface of objective values of zero, with 10 narrow cones that rise up to 10 different optima, the global of which is at the objective value of 1. One can clearly see the plateaus of the zero fitness level in Fig. 6, which displays the fitness function for this test case—generated using (3). All EAxx configurations performed rather poorly with the EA and EAC getting stuck in a local optimum, and the EAF and EAFC performing even worse, fitness-wise, but showing signs of slow improvement. (See Figs.4(a), 4(d), 5(a) and 5(d).) All four techniques,



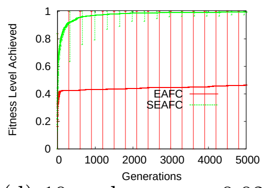
(a) 10 peaks, σ_{peak}=0.02



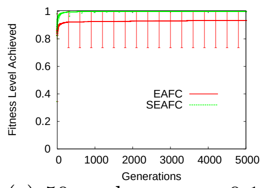
(b) 50 peaks, σ_{peak}=0.1



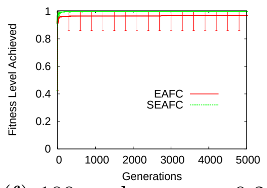
(c) 100 peaks, σ_{peak}=0.2



(d) 10 peaks, σ_{peak}=0.02



(e) 50 peaks, σ_{peak}=0.1



(f) 100 peaks, σ_{peak}=0.2

Fig. 5. Same as Fig. 4 but for EAF vs SEAF(first row), and EAFC vs SEAFC(second row).

once enhanced with scouting, got very close to the global optimum, even from the early stages of the evolution process.

We speculate that the reason the simple EA techniques performed better as the number and width of peaks increased, is because the EAxx small-σ mutation can more easily avoid premature convergence, when the landscape contains more and wider peaks, in which case individuals can “push” their offspring to other peaks with weaker mutation.

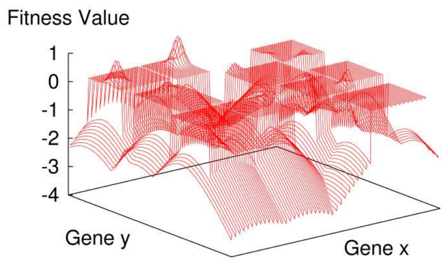


Fig. 6. The fitness landscape for the two-dimensional test case with 10 peaks and σ_{peak} = 0.02. This test case (see Fig. 3(a) for the objective function) is particularly deceptive for a traditional EA, because of its large plateaus of 0.

4.2 A three-dimensional example

After examining the above results for the two-dimensional test cases generated by the TCG-2, we decided to try the SEAxx techniques on a three-dimensional example. As outlined in Sect.3.1, we used the TCG-2 parameters for the test case of 10 peaks and $\sigma_{peak} = 0.02$, and increased the dimensionality of the problem to three.

The results for this three-dimensional deceptive problem are similar to the ones for the two-dimensional cases. The only difference is that the scouting-aided techniques are a little slower in reaching higher levels of fitness. Figure 7 shows the different fitness levels achieved per generation by the EAC and the SEAC. One can clearly see the improvement scouting has on the EAC. The SEAC catches up with the EAC at ca. 500 generations and continues reaching better fitness levels at a steady pace, whereas the EAC on average converges prematurely.

The behaviors of the EA and the SEA are very similar to the EAC and the SEAC. However, the EAFx techniques performed very poorly, whereas the SEAFx ones performed similarly to the SEA techniques that used Roulette Wheel selection.

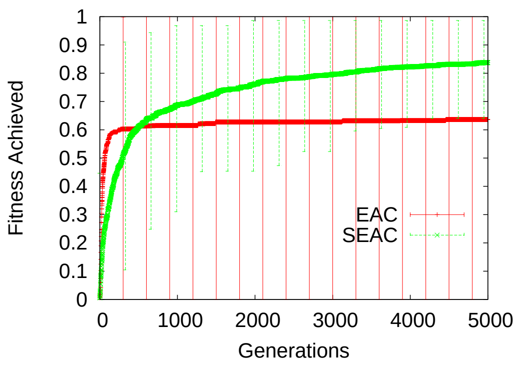


Fig. 7. Fitness level achieved per generation by the EAC and the SEAC with a population size of 20 individuals, for the 3-dimensional case of 10 peaks and $\sigma_{peak} = 0.02$.

5 Conclusion-Future Work

The results obtained by all experiments prove that scouting can make a significant positive difference to the performance of an EA in the domain of NLP problems with rugged landscapes and multiple peaks. We have suggested a new mutation strength modulator and shown that scouting improves a variety of different EA configurations for twelve two-dimensional, multi-modal and deceptive problems, and one three-dimensional example. These configurations, among others, include crossover, and the Fitness-Uniform Selection Scheme(FUSS), a scheme specifically designed to improve performance in such problems. We have also examined the kind of problems SEA is particularly good at, and found that it exhibits particularly impressive performance when dealing with landscapes that contain large plateaus.

Future work will extensively examine the performance of the SEAxx techniques on multi-dimensional problems. Another research direction could be the effect of different crossover and mutation rates on SEAxx techniques. Scouting-driven adaptation of these rates is also an interesting path. Finally, another avenue of exploration could include steady-state EAs and higher population sizes, namely configurations that would favor FUSS. Additional testing could also include some of the problems FUSS was originally tested on.[5]

References

1. Pfaffmann, J.O.; Bousmalis, K.; Colombano, S., "A scouting–inspired evolutionary algorithm," Evolutionary Computation, 2004. CEC2004. Congress on, vol.2, no.pp. 1706–1712, 19–23 June 2004
2. Pfaffmann, J.O., Zauner, K.P., "Scouting context–sensitive components." In Keymeulen, D., Stoica, A., Lohn, J., Zebulum, R.S., eds.: The Third NASA/DoD Workshop on Evolvable Hardware–EH-2001, Long Beach, 12–14 July 2001, IEEE Computer Society, Los Alamitos (2001) pp.14–20
3. Matsumaru, N., Colombano, S., Zauner, K.P., "Scouting enzyme behavior." In Fogel, D.B., El–Sharkawi,M.A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M., eds.: 2002 World Congress on Computational Intelligence, May 12–17, Honolulu, Hawaii, IEEE, Piscataway, NJ (2002) CEC pp.19–24
4. Matsumaru, N., Centler, F., Zauner, K.P., and Dittrich, P., "Self–Adaptive–Scouting Autonomous Experimentation for Systems Biology." EvoWorkshops 2004, LNCS 3005, pp. 52–62, 2004.
5. Hutter, M.; Legg, S., "Fitness uniform optimization," Evolutionary Computation, IEEE Transactions on, vol.10, pp.568–589, Oct. 2006
6. 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, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M.Guervos, and H.–P.Schwefel, Eds. Heidelberg, Germany: Springer–Verlag, 2000, pp. 539–548.
7. 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
8. Beyer H-G, The Theory of Evolution Strategies. Springer, Berlin, 2001
9. 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.
10. M. Lüscher, "A portable high-quality random number generator for lattice field theory calculations," Computer Physics Communications, vol. 79, pp. 1000-110, 1994.