Using Adaptive Novelty Search in Differential Evolution

Iztok Fister[1,3(✉)], Andres Iglesias[2,3], Akemi Galvez[2,3], Javier Del Ser[4,5,6], Eneko Osaba[4], and Iztok Fister Jr.[1]

[1] Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, Maribor, Slovenia
{iztok.fister,iztok.fister1}@um.si
[2] Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan
[3] University of Cantabria, Avenida de los Castros, s/n, 39005 Santander, Spain
{iglesias,galveza}@unican.es
[4] TECNALIA, Derio, Spain
{javier.delser,eneko.osaba}@tecnalia.com
[5] University of the Basque Country (UPV/EHU), Bilbao, Spain
[6] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain

Abstract. Novelty search ensures evaluation of solutions in stochastic population-based nature-inspired algorithms according to additional measure, where each solution is evaluated by a distance to its neighborhood beside the fitness function. Thus, the population diversity is preserved that is a prerequisite for the open-ended evolution in evolutionary robotics. Recently, the Novelty search was applied for solving the global optimization into differential evolution, where all Novelty search parameters remain unchanged during the run. The novelty area width parameter, that determines the diameter specifying the minimum change in each direction needed the solution for treating as the novelty, has a crucial influence on the optimization results. In this study, this parameter was adapted during the evolutionary process. The proposed self-adaptive differential evolution using the adaptive Novelty search were applied for solving the CEC 2014 Benchmark function suite, and the obtained results confirmed the usefulness of the adaptation.

Keywords: Adaptive Novelty search · Differential evolution · Evolutionary robotics · Open-ended evolution

1 Introduction

Two inspirations from the nature have been had the biggest influence on development of the stochastic nature-inspired population-based algorithms: (1) Darwinian evolution [1], and (2) behavior of some animals, or insects living in swarm [2]. The former has been led to emerging the Evolutionary Algorithms (EAs) [3], while the latter to Swarm Intelligence (SI) based algorithms [4]. Traditionally, these algorithms have been developed on disembodied computer systems, where there was no interaction between the system and the environment.

F. De la Prieta et al. (Eds.): PAAMS 2019 Workshops, CCIS 1047, pp. 267–275, 2019.
https://doi.org/10.1007/978-3-030-24299-2_23

Recently, both families of algorithms have been ported on the specific hardware, where act as autonomous agents devoted for solving the problems as delegated by their developers. These agents operate similar as human beings in society, where they need to cooperate, coordinate, and negotiate between each other in order to solve the problem [5]. This porting has been caused development of the so-called Evolutionary Robotics (ER), and Swarm Robotics (SR) [6].

New problems have been emerged by embedding the EAs and SI-based algorithms into hardware. Let us mentioned only two the most important ones: (1) selection pressure, and (2) fitness function evaluation. The selection pressure causes losing the population diversity that is a prerequisite for the open-ended evolution in ER and SR [13]. On the other hand, the fitness function in ER and SR cannot be evaluated in the traditional three-step evaluation chain genotype-phenotype-fitness. Due to the interaction of agent with environment, the fitness function must be evaluated according to the more suitable behavior that it has in the relation with the environment. Consequently, the fitness function is now calculated in a behavior space. As a result, the four-step evolution chain genotype-phenotype-behavior-fitness has replaced the three-step in ER and SR.

Indeed, two advantages in the Evolutionary Computation (EC) [19–22] enable ER and SR community to prevent the loosing of population diversity successfully [12]: (1) Multi-Objective Evolutionary Algorithms (MOEA) [14], and (2) Novelty Search (NS) [10]. The former technology allows to evaluate the same solution according to two or more criteria, while the latter to preserve the so-called novelty solution that will normally be eliminated by the fitness function. In more detail, the NS evaluate the solution in genotype space, where the distance from the neighborhood solutions is calculated. However, the solution is adopted as novelty one, when the distance is larger than those specified by the novelty area width parameter. In other words, the NS estimates the solutions according the two criteria (i.e., fitness function, and distance) and thus changes the traditional EA (or SI-based algorithm) into MOEA.

Interestingly, the operations of the NS are guided by two parameters: (1) the neighborhood size, and (2) the novelty area width needed for calculating the behavior distance metric. In NS for global optimization, these parameters remained unchanged during the evolutionary run until now [7,8]. According to Eiben [3], a fitness landscape of the particular problem is changed during the evolutionary search process. Therefore, changing the parameter setting during the run has potential of better adjusting the algorithm to the problem. Indeed, there are three types of parameter control in EC [3]: (1) deterministic, (2) adaptive, and (3) self-adaptive. Deterministic parameter control means that the parameters are changed according to some deterministic rule during the evolutionary search. In adaptive parameter control, a some feedback from the search process influences on the frequency and the magnitude of change of the problem variables, while the parameters are stored into representation of individuals together with problem variables and undergo the operations of variation operators during the self-adaptive parameter control. In our paper, the adaptive control of the novelty area width is applied, because the proper value of this parameter is

not known in advance, in the one hand, and because it is the most crucial for performing of the NS on the other.

The adaptive NS (ANS) was implemented within the self-adaptive Differential Evolution (jDE) of Brest et al. [16] and applied for solving the CEC 2014 Benchmark function suite. The purpose of our experimental work was to show that the ANS into jDE (AnjDE) can improve the results of the nDE and njDE variants developed by Fister et al. [7,8]. Additionally, the achieved results can be comparable also with the state-of-the-art algorithms, like L-Shade [17] (the winner of the CEC-2014 Competition on Real-Parameter Single Objective) and MVMO [18].

The structure of the remainder of the paper is as follows. Section 2 refers to highlight the background information. A description of the adaptive AnjDE are presented in Sect. 3. The results of experiments are illustrated in Sect. 4. Summarizing of the performed work is the subject of the last section.

2 Background Information

This section presents a background information needed for understanding the subject that follows. In summary, this section captures the following topics: (1) differential evolution, (2) self-adaptive evolutionary evolution, and (3) Novelty search. In the remainder of the paper, the mentioned topics are illustrated in details.

2.1 Differential Evolution

DE belongs to the class of stochastic nature-inspired population-based algorithms that is appropriate for solving continuous as well as discrete optimization problems. DE was introduced by Storn and Price in 1995 [15] and since then many DE variants have been proposed. The original DE algorithm is represented by real-valued vectors that undergo operations of variation operators, such as mutation, crossover, and selection.

In the basic mutation, two solutions are selected randomly and their scaled difference is added to the third solution, as follows:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r0}^{(t)} + F \cdot (\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}), \quad \text{for } i = 1 \dots Np, \tag{1}$$

where $F \in [0.1, 1.0]$ denotes the scaling factor that scales the rate of modification, while Np represents the population size and $r0$, $r1$, $r2$ are randomly selected values in the interval $1 \dots Np$. Note that the proposed interval of values for parameter F was enforced in the DE community.

DE employs a binomial (denoted as 'bin') or exponential (denoted as 'exp') crossover. The trial vector is built from parameter values copied from either the mutant vector generated by Eq. (1) or parent at the same index position laid i-th vector. Mathematically, this crossover can be expressed as follows:

$$w_{i,j}^{(t)} = u_{i,j}^{(t)} x_{i,j}^{(t)} \tag{2}$$

where $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution. The condition $j = j_{rand}$ ensures that the trial vector differs from the original solution $\mathbf{x}_i^{(t)}$ in at least one element. Mathematically, the selection can be expressed as follows:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise}. \end{cases} \tag{3}$$

The selection is usually called 'one-to-one', because trial and corresponding vector laid on i-th position in the population compete for surviving in the next generation, where the better according to the fitness function will survive.

Mutation can be performed in several ways in DE. Consequently, a specific notation was introduced to describe the varieties of these methods (also mutation strategies), in general. For example, 'rand/1/bin' denotes that the base vector is randomly selected, 1 vector difference is added to it, and the number of modified parameters in the trial/offspring vector follows a binomial distribution.

2.2 jDE Algorithm

In 2006, Brest et al. [16] proposed an effective DE variant (jDE), where control parameters are self-adapted during the run. In this case, two parameters namely, scale factor F and crossover rate CR are added to the representation of every individual and undergo acting the variation operators. As a result, the individual in jDE is represented as follows:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, ..., x_{i,D}^{(t)}, F_i^{(t)}, CR_i^{(t)}).$$

The jDE modifies parameters F and CR according to the following equations:

$$F_i^{(t+1)} = \begin{cases} F_l + \text{rand}_1 * (F_u - F_l) & \text{if } \text{rand}_2 < \tau_1, \\ F_i^{(t)} & \text{otherwise}, \end{cases} \tag{4}$$

$$CR_i^{(t+1)} = \begin{cases} \text{rand}_3 & \text{if } \text{rand}_4 < \tau_2, \\ CR_i^{(t)} & \text{otherwise}, \end{cases} \tag{5}$$

where: $\text{rand}_{i=1...4} \in [0, 1]$ are randomly generated values drawn from uniform distribution in interval $[0, 1]$, τ_1 and τ_2 are learning steps, F_l and F_u lower and upper bound for parameter F, respectively.

2.3 Novelty Search

NS measures the distance between each individual in a population and its k-th nearest neighbors in behavior space, in other words [10]:

$$\rho(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} dist(\mathbf{x}, \boldsymbol{\mu}_i), \tag{6}$$

where $\boldsymbol{\mu}_i$ is the i-th nearest neighbor of $\mathbf{x}$ with respect to the behavior distance metric *dist*.

Parameter k is a problem dependent, and must be determined by the developer experimentally. However, the same is also valid for selecting the distance metric. In general, the NS is weakly defined and the question, how to tailor the search so that the results are as good as possible, is left to the developer's criterium [11].

3 Adaptive Novelty Search into jDE

In our study, Adaptive NS (ANS) is applied into jDE. In line with this, two main modifications need to be performed: (1) implementation of the ANS, and (2) adjusting a jDE population scheme accordingly. Here, the same DE/jDE population scheme, as described in Fister et al. [7,8], was used. Therefore, the paper is focused on the description of the ANS. The pseudo-code of the ANS is illustrated in Algorithm 1.

Note that the function $\rho(\mathbf{x})$ in Algorithm 1 calculates the average value of the behavior distance metric in neighborhood. The behavior distance metric *dist* in ANS is defined as follows:

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_{sh}^{(t)}}, & d(\mathbf{x}_i, \mathbf{x}_j) > \sigma_{sh}^{(t)}, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes a Euclidean distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and $\sigma_{sh}^{(t)}$ determines the minimum distance needed for recognizing the novelty solution. Let us mention that the $\sigma_{sh}^{(t)}$ is not remained unchanged during the evolutionary search as proposed in previous studies, but this is modified according to the following equations:

$$\sigma_{sh}^{(t)} = K \cdot I_{\mathbf{c}}^{(t)}. \tag{8}$$

where K is a user-defined constant and $I_{\mathbf{c}}^{(t)}$ represents an inertia moment from mass center expressed as:

$$I_{\mathbf{c}}^{(t)} = \sum_{i=1}^{Np} \sum_{j=1}^{D} (x_{i.j} - c_j)^2. \tag{9}$$

In Eq. (9), vector $\mathbf{c} = \{c_j\}$ represents the mass center expressed as:

$$c_j^{(t)} = \frac{1}{Np} \sum_{i=1}^{Np} x_{i,j}, \quad \text{for } j = 1, \ldots, D. \tag{10}$$

The constant K in Eq. (8) allows users to increase of decrease the value of novelty area width. The adaptation process is launched simultaneously with the ANS in the sense of the learning rate parameter $\tau_3 \in [0, 1]$. The ANS process is controlled with additional parameters replacement size $R \in [1, Np]$ that limits the number of novelty solutions.

Algorithm 1. Adaptive Novelty Search within the jDE algorithm

procedure NOVELTY SEARCH
  $\mathcal{A} = \{\exists \boldsymbol{\mu}_i : f(\mathbf{w}_i) \leq f(\mathbf{x}_j) \wedge i \neq j\}$; // set of survivor solutions
  $\mathcal{B} = \{\exists \mathbf{x}_i : f(\mathbf{w}_i) > f(\mathbf{x}_j) \wedge i \neq j\}$; // set of eliminated solutions
  if $|\mathcal{A}| < k$ **then** // number of survivor solutions less than the neighborhood?
    $\mathcal{A} = \mathcal{A} \cup \mathcal{B}$; // increases the neighborhood set to the whole population
  end if
  $\forall \mathbf{x}_i \in \mathcal{B} : \exists \mathcal{N}(\mathbf{x}_i) : \boldsymbol{\mu}_j \in \mathcal{A} \wedge \mathbf{x}_i \neq \boldsymbol{\mu}_j \wedge |\mathcal{N}(\mathbf{x}_i)| \leq k$; // select k-nearest neighbors
  $\forall \mathbf{x}_i \in \mathcal{B} : \rho(\mathbf{x}_i)$; // calculate their novelty values adaptively
  $\mathcal{C} = \{\forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{B} : \max |\rho(\mathbf{x}_i) - \rho(\mathbf{x}_j| \wedge i \neq j \wedge |\mathcal{C}| \leq R\}$;
end procedure

4 Experiments and Results

The purpose of our experimental work was to show that (1) the ANS improves the results of the original DE/jDE and the proposed nDE/njDE, as well as to indicate that (2) they are also comparable with those obtained by the MVMO and L-Shade, the winner of the CEC 2014 Competition on global optimization. In line with this, the CEC 2014 Benchmark function suite was used as a test bed. This suite consists of 30 shifted and rotated functions that present the big challenge for the majority of the optimization algorithms. Due to the paper length, this study was focused on the functions of dimension $D = 10$.

Table 1. Parameter setting of algorithms in tests.

Alg.	F	CR	$\tau_{1,2}$	σ_{sh}	$\lvert\mathcal{N}\rvert$	τ_3	R
DE	0.5	0.9	n/a	n/a	n/a	n/a	n/a
jDE	0.5	0.9	0.1	n/a	n/a	n/a	n/a
nDE	0.5	0.9	n/a	50	10	0.1	50
njDE	0.5	0.9	0.1	5	10	0.1	5

The parameter settings of the original DE/jDE and proposed nDE/njDE are presented in Table 1. All algorithms in tests were run with population size of $Np = 100$, while 25 independent runs were conducted per each algorithm. The results were measured according to five statistical measures: minimum, maximum, mean, median, and standard deviation. These values obtained by optimization of 30 particular functions were accumulated into statistical classifiers and entered into Friedman non-parametric tests [9].

To prove our hypotheses, two tests were conducted. The aim of the first was to show that the AnjDE improve the results of the traditional DE/jDE and proposed nDE/njDE. In line with this, the constant K was varied in the interval $[0.009, \ldots, 0.01]$ in steps of 0.001, in the interval $(0.01, \ldots, 0.1]$ in steps of 0.01,

and in the interval $(0.1, \ldots, 0.6]$ in steps of 0.1. In line with this, 16 instances of the AnjDE algorithm were obtained for each of the six values of the neighborhood size $|\mathcal{N}| \in \{5, 10, 15, 20, 30, 50\}$ and the replacement size $R \in \{1, 2, 5, 10\}$. Thus, the $16 \times 6 \times 4 \times 25 = 9,600$ independent runs were conducted, from which the best AnjDE instance (i.e., AnjDE with $K = 0.3$, $|\mathcal{N}| = 30$, $\tau = 0.1$, and $R = 2$) according to the rank enters in Nemenyi post-hoc statistical test. The corresponding results according to average differences of ranks are depicted in Fig. 1.

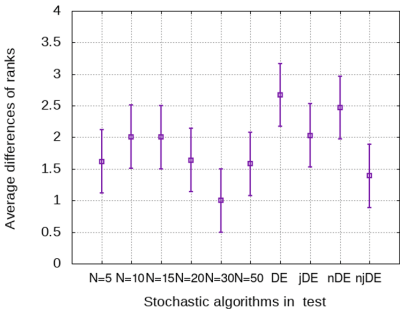

Fig. 1. Different AnjDE-variants

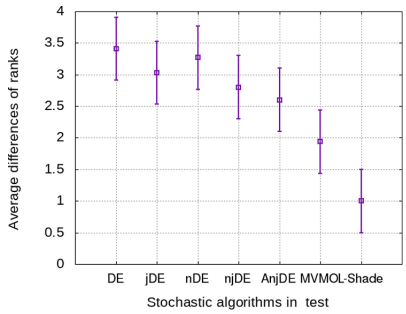

Fig. 2. State-of-the-art comparison

The second test was dedicated to show that the results of the AnjDE are comparable to the results obtained by the DE/jDE, and nDE/njDE as well as the winners of the CEC 2014 competition, i.e., L-Shade and MVMO. The results of mentioned algorithms are depicted according to average differences of ranks in Fig. 2, from which it can be seen that the AnjDE outperformed the results of the other DE algorithms in test substantially, while the results of the L-Shade and MVMO remains the hard problem for solving in the future.

5 Conclusion

Stochastic nature-inspired population-based algorithms have been achieved their maturity phase recently. In the past, they were developed on disembodied computer systems. With huge development of hardware, they have become a new application domain in ER and SR, where these algorithms were ported on specific hardware put into environment and play a role of autonomous agents. Throughout this process, two major problems have been emerged: (1) maintaining the population diversity as a prerequisite of open-ended evolution, and (2) evaluating the fitness function based on the behavior of the solution in environment.

In this paper, we were focused on solving of the first problem, where the NS was proposed as a tool for maintaining the population diversity. Actually, the paper is continuation of already published papers of Fister et al. [7,8], wherein

the value of NS parameters remains unchanged during the search process. Indeed, the novelty area width, that is the crucial for the results of the optimization, was adapted and thus the proposed AnjDE algorithm is able to adjust to the problem during the search.

The proposed AnjDE was applied to CEC 2014 Benchmark function suite. The results of optimization showed that this improved the results of the traditional DE/nDE, and nDE/njDE substantially, when they are comparable with those obtained by the MVMO and L-Shade.

As the future work, finishing experiments on the higher dimensional functions remains in first place. Additionally, the impact of self-adaptation of NS parameters could become a big challenge in the future. Finally, it would also be interesting to incorporate the NS in L-Shade or MVMO.

Acknowledgments. Iztok Fister and Iztok Fister Jr. acknowledge the financial support from the Slovenian Research Agency (Research Core Fundings No. P2-0041 and P2-0057). Javier Del Ser and Eneko Osaba would like to thank the Basque Government for its funding support through the EMAITEK program.

References

1. Darwin, C.: On the Origin of Species. Harvard University Press, London (1852)
2. Beni, G., Wang, J.: Swarm intelligence in cellular robotic systems. In: Proceedings of NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, 26–30 June 1989
3. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2003). https://doi.org/10.1007/978-3-662-05094-1
4. Blum, C., Merkle, D.: Swarm Intelligence: Introduction and Applications, 1st edn. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-74089-6
5. Wooldridge, M.: An Introduction to Multiagent Systems, 2nd edn. Wiley, Hoboken (2009)
6. Eiben, A.E., Smith, J.E.: From evolutionary computation to the evolution of things. Nature **521**(7553), 476–482 (2015)
7. Fister, I., Iglesias, A., Galvez, A., Del Ser, J., Osaba, E., Fister Jr., I.: Using novelty search in differential evolution. In: Bajo, J., et al. (eds.) PAAMS 2018. CCIS, vol. 887, pp. 534–542. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94779-2_46
8. Fister, I., et al.: Novelty search for global optimization. Appl. Math. Comput. **347**, 865–881 (2019)
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006). JMLR.org
10. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. In: Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI), pp. 329–336. MIT Press, Cambridge (2008)
11. Doncieux, S., Mouret, J.B.: Behavioral diversity measures for Evolutionary Robotics. In: IEEE Congress on Evolutionary Computation, Barcelona, pp. 1–8 (2010)
12. Doncieux, S., Mouret, J.B.: Beyond black-box optimization: a review of selective pressures for evolutionary robotics. Evol. Intell. **7**(2), 71–93 (2014)

13. Lynch, M.: The evolution of genetic networks by non-adaptive processes. Nat. Rev. Genet. **8**, 803–813 (2007)
14. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, New York (2001)
15. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
16. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)
17. Tanabe, R., Fukunaga, A.S.: Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, pp. 1658–1665 (2014)
18. Erlich, I., Rueda, J.L., Wildenhues, S., Shewarega, F.: Evaluating the mean-variance mapping optimization on the IEEE-CEC 2014 test suite. In: 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, pp. 1625–1632 (2014)
19. Valdez, F., Melin, P., Castillo, O.: An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms. Appl. Soft Comput. **11**(2), 2625–2632 (2011)
20. Precup, R.-E., David, R.-C., Petriu, E.M., Preitl, S., Paul, A.S.: Gravitational search algorithm-based tuning of fuzzy control systems with a reduced parametric sensitivity. In: Gaspar-Cunha, A., Takahashi, R., Schaefer, G., Costa, L. (eds.) Soft Computing in Industrial Applications, vol. 96, pp. 141–150. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20505-7_12
21. Saadat, J., Moallem, P., Koofigar, H.: Training echo state neural network using harmony search algorithm. Int. J. Artif. Intell. **15**(1), 163–179 (2017)
22. Vrkalovic, S., Lunca, E.-C., Borlea, I.-D.: Model-free sliding mode and fuzzy controllers for reverse osmosis desalination plants. Int. J. Artif. Intell. **16**(2), 208–222 (2018)