Non-Stationary Optimization with Multi-Population Evolutionary Algorithm

Jarosław Stańczak[1] and Krzysztof Trojanowski[2,3]

[1] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland,
email: stanczak@ibspan.waw.pl
[2] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland,
email: trojanow@ipipan.waw.pl
[3] Institute of Computer Science, University of Podlasie, Siedlce, Poland

Abstract. Non-stationary optimization of randomly changing environments is a subject of unfading interest. In this paper we study application of multi-population evolutionary algorithm to this problem. Presented algorithm works with a set of sub-populations managed by the mechanism of exclusion coming from the multiswarm version of particle swarm approach. The results show significant improvement of the efficiency of the new algorithm in comparison with a single population approach.

1 Introduction

Publication by Goldberg and Smith [8] in 1987 started exploration of a new research area of a non-stationary optimization with heuristic techniques. This type of optimization problems appears in the real world very often. It could be a problem of efficient finding the shortest path from one point to another in the crowded streets of a big city as well as a task of the continuous control of the parameters of the engine to keep it working the most efficiently i.e. in the optimal conditions for varying requests of the user. During two decades there appeared lots of publications concerning evolutionary approach to this problem. Recently there are also publications where the heuristic approaches other than the evolutionary computation are applied. There are papers about particle swarm (starting from 2000 e.g. [5]) or immune optimization algorithms (starting from 1999 e.g. [7]). It is natural that some ideas move between heuristics and this way new approaches are created. An example of such transfer of ideas is presented in this paper. The presented algorithm employs mechanisms of multiswarm management in the multi-population version of the evolutionary algorithm.

The paper is organized as follows. In Section 2 a brief description of the optimization algorithm is presented. Section 3 presents some details of the selected testing environment while Section 4 – the results of experiments performed with the environment. Section 5 concludes the presented research.

2 The multi-population evolutionary algorithm

As a main framework of our search engine a classic evolutionary algorithm was applied. The algorithm works with real valued vectors of coordinates of solutions in an n-dimensional search space.

Evolutionary operators The algorithm is equipped with a suite of evolutionary operators. Some of them were already tested for non-stationary optimization [13] while the others are new in the suite however they are also known in the literature. The set consists of the following operators:

- mutation, version one – replaces one of the values in the array of size n by the value generated by an uniform random number generator of the range $(-50\%; 50\%)$ respectively to the selected value (rather small change of the value). A new value of an l-th coordinate of a solution $\mathbf{x}$ is calculated as follows:

$$\begin{array}{lll} \texttt{if } U(0,1) > 0,5 & \texttt{then} & \mathbf{x}'[l] = \mathbf{x}[l] + U(0, hi_l - x[l]) \cdot 0.05, \\ & \texttt{else} & \mathbf{x}[l] = \mathbf{x}[l] - U(0, x[l] - lo_l) \cdot 0.05. \end{array}$$

  where: $\mathbf{x}[l]$ — the value of the l-th coordinate of the solution, $\mathbf{x}[l]$ — the value of the l-th coordinate of the clone's predecessor, lo_l,hi_l — lower and upper limit for the l-th coordinate, i.e. $\mathbf{x}[l] \in [lo_i, hi_i]$, $U(a, b)$ — uniformly distributed random value from $[a, b]$.
- mutation, version two – classic real valued mutation, calculated as follows:

$$\mathbf{x}'[l] = \mathbf{x}[l] + 0.5 * N(0, 1) \tag{1}$$

- mutation, version three – originates from clonal selection based algorithms, precisely from [6] where it was a component of opt-Ainet optimization algorithm. The operator showed its efficiency for non-stationary optimization [15] and therefore it was added to the suite. A new value of an l-th coordinate of a solution $\mathbf{x}$ is calculated as follows:

$$\mathbf{x}'[l] = \mathbf{x}[l] + N(0, \sigma_l), \tag{2}$$

$$\sigma_l = (r_m/\beta) \cdot (dom\_width_l/2) \cdot \exp(-f'(\mathbf{x})). \tag{3}$$

  where: $\mathbf{x}[l]$ – a value of the l-th coordinate of $\mathbf{x}$, $N(0, 1)$ – a Gaussian random variable of zero mean and $\sigma = 1$, r_m – the mutation range ($0 < r_m \leq 1$), β – a weight factor which for all our experiments was set to 1, dom_width_l — a constant value which is equal to the distance between the upper and the lower boundary of the l-th dimension of the search domain, and $f'(\mathbf{x})$ is the fitness of the clone's predecessor normalized in [0,1]:

$$f'(\mathbf{x}) = \frac{f(\mathbf{x}) - f_{min}}{(f_{max} - f_{min})}, \tag{4}$$

$$f_{max} = \max_{\mathbf{x}_j, \forall j \in \{1,\ldots|P|\}} f(\mathbf{x}_j) \text{ and } f_{min} = \min_{\mathbf{x}_j, \forall j \in \{1,\ldots|P|\}} f(\mathbf{x}_j).$$

- mutation, version four – same as in version three, but works for all cells in the vector.

- mutation, version five – inversion of some randomly chosen sub-vector the whole vector representing multi-dimensional solution.

If the value of any of the coordinates after mutation $\mathbf{x}[l]$ is out of the domain then the remainder of $\mathbf{x}[l]$ and dom_width_l is evaluated: $\mathbf{x}[l] = \mathbf{x}[l] \bmod dom\_width_l$.

Operators efficiency evaluation rules In the algorithm a method of dynamic selection of operators was applied, as described in [10, 11, 12]. The method is based on a general rule that the more improvement of the fitness value of an individual is obtained, the larger is the growth of the value of the quality factor assigned to the operator that caused the improvement. The quality factors are assigned to the individuals i.e. each of the individuals possess its own set of factors. Probabilities of operators' selection are obtained using simple normalization of quality factors and each individual selects one operator in one iteration according to its own experience. The factors are modified every time the operator works on the selected individual. They are increased when the modification is successful, i.e. an offspring is better than the parent, and decreased when the modification fails. The same operator can have (and usually has) different evaluations at different individuals. For more details about the dynamic selection the reader is referred to [13].

The only difference between the selection published in [10, 11, 12] and the selection applied in the research presented below lies in the way of calculation of the quality factors for the operators in a solution. In the presented approach we applied a new method called TD(λ). The method is based on machine learning and reinforcement learning as well. An individual is an agent which role is to select and call one of the evolutionary operators. When the selected i-th operator is applied it can be said that an agent performs action a_i leading to a new state s_i which in our case is a new solution. Agent receives rewards and penalties respectively to the quality of the new state. The aim of the agent is to perform the actions which give the highest long term discounted cumulative reward V^*.

$$V^* = \max_{\Pi} V^{\Pi}, \qquad V^{\Pi} = E_{\Pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\} \tag{5}$$

where:

- Π – represents strategy of the agent,
- V^{Π} – represents discounted cumulative reward obtained using strategy Π,
- E_{Π} – represents expected value of the reward when using strategy Π,
- γ – represents discount factor,
- k – represents following time steps,
- t – represents current time

For evaluation of the optimal strategy Π the following formula is applied:

$$V(s_{t+1}) = V(s_t) + \alpha[r_{t+1} + \gamma V^*(s_{t+1}) - V(s_t)] \tag{6}$$

where:

- $V(s_t)$ – is a quality factor or discounted cumulative reward,
- $V^*(s_{t+1})$ – estimated value of the best quality factor (in our experiments we take the value gained by the best operator)
- α – is a learning factor

- γ – is a discount factor
- r_{t+1} – represents the reward for the best action which is equal to the improvement of the quality of a solution after execution of the evolutionary operator
- t – current moment in time.

In the presented experiments the values of α and γ were set to 0.1 and 0.2 respectively. The selection was based on the set of additional experiments where each possible pair of values for α and γ from 0 to 1 with step 0.1 was tested (11 steps each parameter).

Selection At the beginning of our research we tested a set of selection methods to find the most efficient one. We did tests with tournament, niching, deterministic, histogram and mixed selection. All of them were tested with a single population of individuals and the obtained results were not satisfying. A real improvement was obtained when we turned to a multi-population management of the set of solution. The selected method of population management is based on the idea of *Exclusion* [1].

In our approach the population is divided into a set of sub-populations. The number of subpopulation is constant during the process of search. However there are mechanisms which eliminate some of the sub-populations and introduce another ones. To guarantee distribution of the subpopulation over the search space the exclusion mechanism eliminates sub-populations which are located too close to each other. When the populations are too close, most probably they occupy the same optimum. In such a case one of them is eliminated and a new subpopulation is generated from scratch. Any two sub-populations are too close if for the two best individuals from the compared sub-populations the distance for at least n coordinates is closer than the defined threshold ρ. If so the sub-population with the better solution stays unchanged while the another one is exchanged by a set of randomly generated individuals. Every m iterations of the search process all the populations were tested if they were too close and if the respective conditions were satisfied the mechanism of exclusion was executed.

It is worth to note that except for exclusion yet another mechanism of sub-populations' management suitable to transfer to evolutionary algorithms called *Anti-Convergence* was proposed in [1]. Anti-Convergence protects against convergence of sub-populations which is a negative phenomenon since such sub-populations are not resistant to changes in the environment. The performed action is simple: in case of convergence of all the populations the worst one is exchanged by a set of randomly generated individuals. We did also some preliminary tests with this mechanism but in opposite to the results of particle swarm approach presented in [1] the results obtained with evolutionary algorithm decreased when this mechanism was employed. Therefore we did not use this in our research.

Except for the exclusion mechanism described above there is no interaction between sub-populations. Each of them is treated as an independent self-governing population which is not influenced by any of the neighbors. The implemented process of evolution in a single sub-population is based on the classic idea of parents and offspring. The individuals have successors and the number of successors of an individual is proportional to the fitness function value for this individual respectively to the mean fitness of the entire sub-population. The offspring solutions are gathered in a set and when the set is complete i.e. when all the parent solutions generated their offspring the replacement starts. Those of the solutions from a set of parents are replaced by the solutions from

the set of offspring for whom the fitness of the offspring is better. This deterministic method of succession looks like very primitive however for the proposed sub-population management and non-stationary optimization tasks showed to be the most efficient.

3 Test case and applied measures

For our tests we selected the MPB [3, 9] generator as the most known of the generators of randomly changing testing environments. In MPB the landscape is build of a set of unimodal functions individually controlled by the parameters allowing to create different types of changes. The parameters of the MPB were set exactly the same as specified in the publicly available web page [2] for scenario 2. The fitness landscape was defined for the 5-dimensional search space with boundaries for each of dimensions set to $\langle 0; 100 \rangle$. For such a domain there exist a set of moving peaks which vary their height randomly within the interval $\langle 30; 70 \rangle$, width within $\langle 1; 12 \rangle$ and position by a distance of 1 every 10 iterations of the search process.

In the performed experiments the offline error, *oe* measure [3, 4] of obtained results was used. The returned value represents the average deviation from the optimum of the fitness of the best individual evaluated since the last change of the fitness landscape. Formally:

$$oe = \frac{1}{N_c} \sum_{j=1}^{N_c} \frac{1}{N_e(j)} \sum_{k=1}^{N_e(j)} (f_j^* - f_{jk}^*) \tag{7}$$

where:
N_c is the total number of changes of the fitness landscape in the experiment, $N_e(j)$ is the number of evaluations of the solutions performed for the j-th state of the landscape, f_j^* is the value of optimal solution for the j-th landscape and f_{jk}^* is the best value found among the ones belonging to the set from f_{j1} till f_{jk} where f_{jk} is the value of the fitness function returned for its k-th call performed between the j-th and $(j+1)$-th change in the landscape.

During the process of search the offline error can be calculated in two ways: in one of them the error is evaluated from the beginning of the experiment while in another one – the value of offline error starts to be evaluated only after some number of changes in the fitness landscape. This second way is advised in the literature as saddled with the less measurement error caused by the initial phase of the search process (for extended discussion on the possible influence of the initial phase on the results obtained for MPB the reader is referred to [14]). Therefore, just this way was applied in our tests.

To make a possibility of comparisons with results published by other authors the number of evaluations between subsequent changes is similar and equals approx. 5000. During a single experiment the fitness landscape changed 110 times (however for the first 10 changes the error was not evaluated). Every experiment was repeated 50 times and the mean is presented in the Figures.

4 Results of experiments

Our experiments can be divided into three stages. In the first stage we wanted to check if the verification of the closeness criterion should have to be based on the full set of

coordinates. In this stage we did two groups of experiments where we observed the values of offline error obtained for different numbers of dimensions checked to verify the closeness of any two sub-populations and for different numbers of sub-populations. The criterion of closeness was satisfied if for any n of 5 dimensions the coordinates of compared solutions were in a closer distance than the selected threshold ρ where n changes from 1 to 5. We did tests for two values of ρ: for 0.05 and for 0.04. The results are presented in Figure 1.

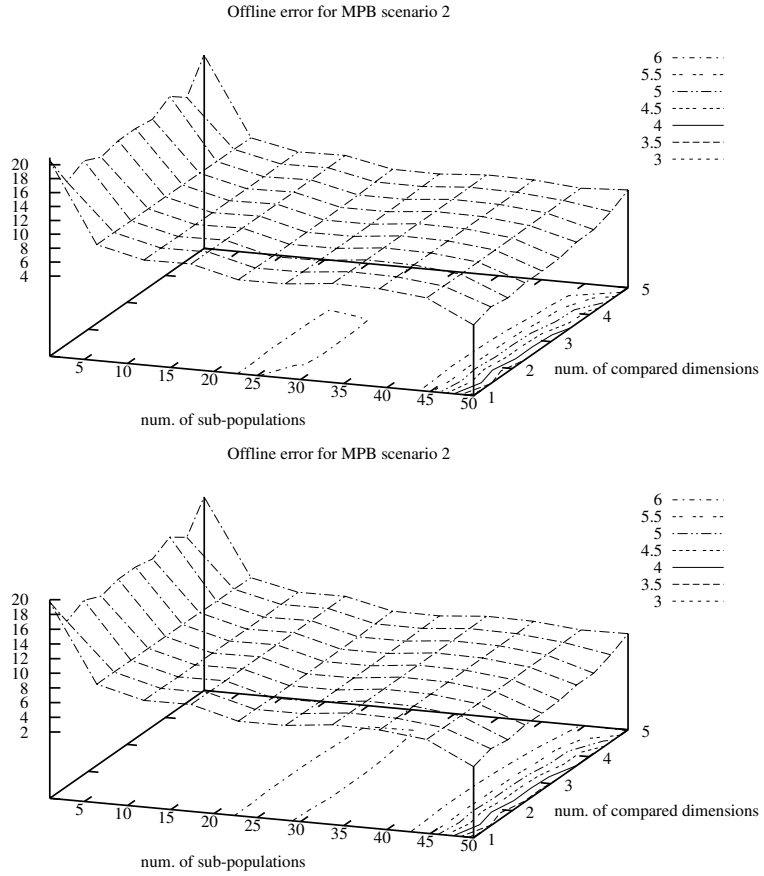

Figure 1. Offline error for different numbers of dimensions checked to verify the closeness of any two sub-populations vs. numbers of sub-populations for two values of closeness threshold: $\rho = 0.05$ (top) and $\rho = 0.04$ (bottom)

In the second stage we observed the influence of the closeness threshold on the offline error for different numbers of sub-populations. The results from the first stage allowed us to select the best number of coordinates used to verify the closeness of any two population. The number was set to 3. The results are presented in Figure 2.

In the third stage we selected the number of 25 sub-populations and for such a number of sub-populations we compared the value of offline error obtained for different values

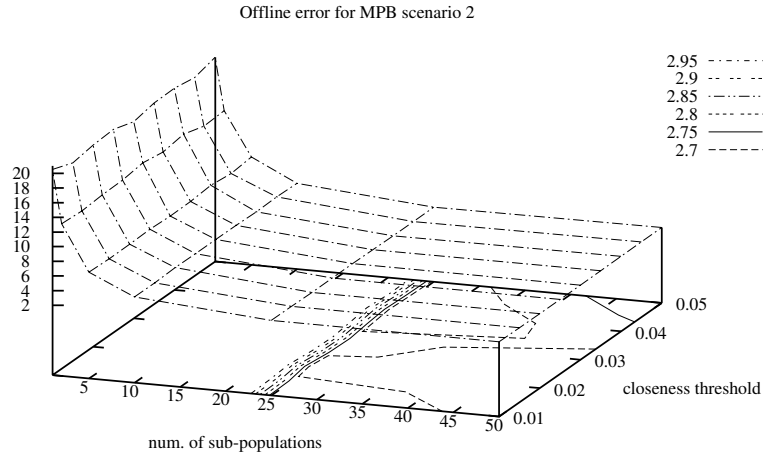

Figure 2. Offline error for different values of closeness threshold ρ vs. number of sub-populations for the rule of closeness where for just 3 of the five dimensions of the search space the distance between coordinates is checked.

of closeness threshold ρ and for different numbers of dimensions checked to verify the closeness of any two sub-populations. The results are presented in Figure 3.

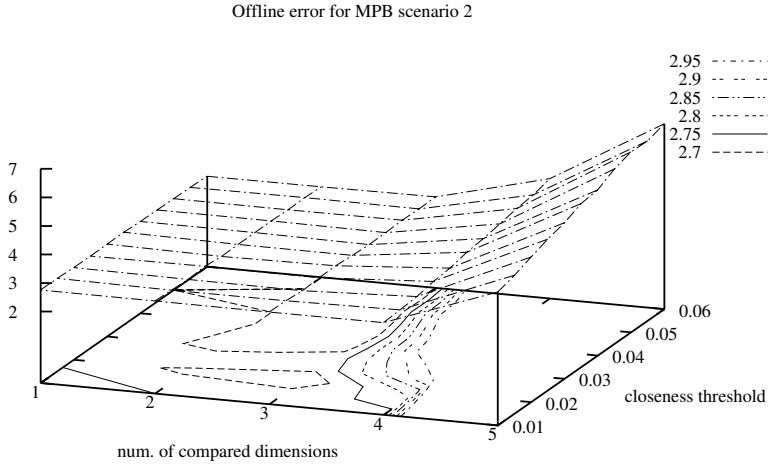

Figure 3. Offline error for different values of closeness threshold ρ vs. different numbers of dimensions checked to verify the closeness of any two sub-populations for a set of 25 sub-populations

The three stages of the experiments presented above allowed us to find the optimal set of parameters' values (of course the set is as optimal as precise was the search through the domain of the possible parameters' settings). For the selected set of values i.e. for $\rho = 0.05$, 25 sub-populations, and for 3 coordinated checked to verify the closeness of the sub-populations we did a single experiment where the current value of offline error was

monitored during the search of process. The series of 50 runs of this experiment took 941,8 sec for the Pentium III 790MHz with RedHat 9.0. The observed changes in the offline error in a single experiment are presented in Figure 4.

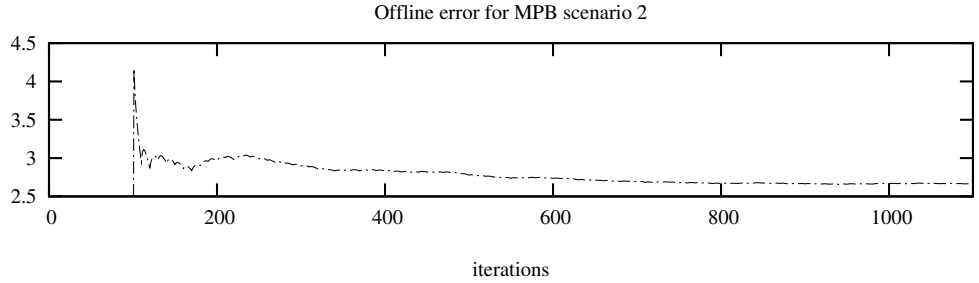

Figure 4. Offline error evaluated at subsequent changes of the fitness landscape. Evaluation was started after the 10-th change in the landscape. Key parameters settings: $\rho = 0.05$, 25 sub-populations, 3 coordinated are checked to verify the closeness of the sub-populations. Final value of the offline error: 2.66

5 Conclusions

The main goal of this work was further research in the field of evolutionary optimization of non-stationary tasks. The results show that almost all the components of the algorithm should be fitted to this very specific type of optimization. Among them the most important role plays appropriate population management. The empirical results showed that co-evolutionary approach with a simple deterministic selection in sub-populations and the mechanism of exclusion of populations being too close is the most efficient. Mechanism of exclusion introduced additional parameters necessary to tune like number of dimensions to check or closeness range.

Another component of the algorithm to rebuild was a set of evolutionary operators allowing quick and precise following the changing optimum. Performed experiments confirmed necessity of careful selection of modification operators: leaving those which introduce the most valuable modifications and avoiding those which generate less valuable modifications and thus simply waste calls of fitness function evaluation. The set of the operator needed appropriate management strategy of use of them. The proposed strategy is based on the methods applied in the reinforcement learning.

Finally there was created an algorithm equipped with strong adaptive abilities which follows the varying optimum quickly and with respectively low level of the offline error. We hope that with a further research of the population management and selection methods in the algorithm the current level of efficiency can be yet improved.

Bibliography

[1] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Trans. on Evolutionary Computation*, 10(4):459–472, 2006.

[2] J. Branke. The moving peaks benchmark. URL: http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/movpeaks/.

[3] J. Branke. Memory enhanced evolutionary algorithm for changing optimization problems. In *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1875–1882. IEEE Press, Piscataway, NJ, 1999.

[4] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

[5] A. Carlisle and G. Dozier. Adapting particle swarm optimization to dynamic environments. In *Proc. of the Int'nal Conf. on AI (IC-AI 2000)*, volume I, pages 429–434. CSREA Press, 2000.

[6] L.N. de Castro and J. Timmis. An artificial immune network for multimodal function optimization. In *Proc. of the Congress on Evolutionary Computation*, volume 1, pages 699–674. IEEE Press, Piscataway, NJ, 2002.

[7] A. Gaspar and P. Collard. From GAs to artificial immune systems: Improving adaptation in time dependent optimization. In *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1859–1866. IEEE Press, Piscataway, NJ, 1999.

[8] D. E. Goldberg and R. E. Smith. Non-stationary function optimisation using genetic algorithms with dominance and diploidy. In *Genetic Algorithms and Their Applications: Proc. of the 2nd Int'nal Conf. on Genetic Algorithms (ICGA-2)*, pages 59–68. Lawrence Erlbaum Associates, 1987.

[9] R. W. Morrison and K. A. De Jong. A test problem generator for non-stationary environments. In *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1859–1866. IEEE Press, Piscataway, NJ, 1999.

[10] J. Mulawka, J. Stańczak, and B. K. Verma. Genetic algorithms with adaptive probabilities of operators selection. In *Third Int'nal Conf. on Computational Intelligence and Multimedia Applications (ICCIMA'99)*, pages 464–468. IEEE Computer Society, 1999.

[11] J. Stańczak. Algorytm ewolucyjny z populacją 'inteligentnych' osobników. In *Proc. of the 4th National Conf. on Evolutionary Computation and Global Optimisation*, pages 207–218. Warsaw Univ. of Technology Publishing House, 2000.

[12] J. Stańczak. Biologically inspired methods for control of evolutionary algorithms. *Control and Cybernetics*, 32(2):411–433, 2003.

[13] J. Stańczak and K. Trojanowski. Properties of selection methods applied to non-stationary optimization tasks. In *Proc. of the 7th National Conf. on Evolutionary Computation and Global Optimisation*, pages 171–180. Warsaw Univ. of Technology Publishing House, 2004.

[14] K. Trojanowski. B-cell algorithm as a parallel approach to optimization of moving peaks benchmark tasks. Accepted for publication at the Int'nal Conf.: 6-th Computer Information Systems and Industrial Management Applications, CISIM 2007, Ełk, Poland, June 28 - June 30, 2007.

[15] K. Trojanowski and S. T. Wierzchoń. B-cell algorithm for non-stationary optimization. In *Advanced Computer Systems, Proc. of ACS/CISIM Conf.*, volume 1, pages 53–64. Publishing House of Szczecin Univ. of Technology, 2006.