B-Cell Algorithm as A Parallel Approach to Optimization of Moving Peaks Benchmark Tasks

Krzysztof Trojanowski
Institute of Computer Science, Polish Academy of Sciences
Ordona 21, 01-237 Warsaw, Poland
trojanow@ipipan.waw.pl

Abstract

Efficiency of the B-Cell algorithm applied to one of the well-known test-case generators, the Moving Peaks Benchmark (MPB) is a subject of study presented in this paper. We especially focused on the family of fitness landscapes generated by scenario 2 of the MPB. All of them represent the type of randomly changing environment. Some properties of the algorithm as well as the properties of the environments created by the generator are discussed. A side effect of modification of one of the control parameters and its influence on the offline error measure is presented.

1. Introduction

Heuristic optimization algorithms inspired by immune metaphor were heavily studied for approaches to stationary optimization tasks. However there are still areas of non-stationary optimization where some research should be done. For non-stationary optimization there were published a few benchmarks which became commonly accepted for comparisons between the algorithms. One of them is Moving Peaks Benchmark (MPB) [2, 3]. The benchmark represents the case where a set of peaks randomly changes their height, width and coordinates in the search space. For this type of changes in the fitness landscape the most appropriate heuristic approach is based on careful management of a set of sub-populations for simultaneous tracking of the most promising peaks from one side and continuous search over the entire search space - from the other. Examples of such algorithms can be self-organizing scouts [4] or more recent particle swarm approach presented in [1]. Among immune optimization algorithms there are a few which seem to be naturally equipped with a parallel management of the population of solutions and one of them is the B-Cell algorithm (BCA) [5].

The B-Cell algorithm was already applied to non-stationary optimization tasks and proved its efficiency for both cyclically and non-cyclically changing environments [6, 8]. The research showed however that the key feature of the algorithm which is responsible for the efficiency is the representation of solutions. It is concerned with a very specific mutation operator called contiguous mutation. Originally in [5] authors proposed a representation of solutions where a 64-bit double precision floating point coding (according to IEEE 754 standard) was applied and as it was shown in [6, 8], this way of binary coding is the most appropriate for the proposed contiguous mutation operator.

In this paper we wanted to evaluate efficiency of the algorithm for a set of fitness landscapes with different numbers of moving peaks. We wanted to observe the influence of the complexity of the problem i.e. the number of moving peaks in the landscape on the algorithms efficiency and verify the dependency between the population size and the number of moving peaks. Thus we decided to do the tests with the MPB only and selected scenario 2 of the MPB as our test-bed.

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. B-Cell Algorithm (BCA)

Our version of the B-Cell algorithm originates from [5]. The pseudo-code of the main loop of BCA is given in Figure 1. The symbol \( x_i \in P \) represents \( i \)-th B-cell of the population \( P \), \( x_{c_i, k} \) – \( k \)-th copy (clone) of the \( i \)-th B-cell subjected further mutation, \( f(x_i) \) – B-cell affinity to the antigen, i.e. fitness of the B-cell \( x_i \), and \( x_{c_i} \) is the best mutated clone of the \( i \)-th B-cell, i.e. \( f(x_{c_i}) > f(x_i), k \in \{1, \ldots, c\} \) where \( c \) is the number of clones produced by the B-cell.

The algorithm starts with a population of randomly gen-
erated solutions from the search domain and performs the process of iterated improvement of the solutions by the execution of the main loop depicted in Figure 1. The loop consists of two main blocks: (i) affinity evaluation step 1, and (ii) clonal selection and expansion - step 2 and all of its substeps.

1. For each B-Cell $x_i \in P$ compute its fitness i.e. the value of the objective function $f(x_i)$.
2. For each B-Cell $x_i$ do
   2.1 Make $c$ clones $x_{c_i,k}$, $k \in \{1, \ldots , c\}$ of $x_i$ and place in a clonal pool $C_i$.
   2.2 Randomize one clone $x_{c_i,(j)}$ in $C_i$ where $(j)$ stands for a number from $\{1, \ldots , c\}$.
   2.3 Apply contiguous mutation to all the remaining clones in $C_i \setminus x_{c_i,(j)}$.
   2.4 For each clone in $C_i$ compute its fitness.
   2.5 The mutated clone $x_{c_i}^*$ with the highest fitness replaces the original B-Cell if $f(x_{c_i}^*) > f(x_i)$.

Figure 1. Pseudo-code of the main loop of BCA

The algorithm has three control parameters: $|P|$ – population size, $c$ – number of clones in the clonal pool, and $n_0$ – number of bits for each of the coordinates because a binary representation of solution is applied here.

A 64-bit double precision floating point coding (according to IEEE 754 standard) was selected for our experiments. Contiguous mutation was implemented as described in [5] (see also [7] for description of the operator and discussion about the properties of the operator applied to 64-bit double precision floating point coded representation of solution) and applied to both versions.

3. Testing Environment

For our tests we selected the MPB generator as the most known of the generators of randomly changing testing environments. 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 $[0; 100]$. For such a domain there exist a set of moving peaks which vary their height randomly within the interval $[30; 70]$, width within $[1; 12]$ and position by a distance of 1 every 20 iterations of the search process. With scenario 2 of the MPB we created a set of testing environments where the number of moving peaks varies from 2 to 50 with step 2. Two fitness landscapes of the MPB scenario 2 for 2-dimensional search space are presented in Figure 2: with 2 and with 50 moving peaks.

4. Results of Experiments

4.1. Plan of experiments and applied measures

As mentioned earlier, the only varied parameter of the MPB generator was the number of moving peaks. All the remaining parameters were constant and set on the same values as given in the web page [2]. To evaluate the results we used an offline error measure which is also described in [2].

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. Therefore, just this way was applied in the first group of our tests. In the second group we wanted to verify
the role of the way of evaluation of the offline error for particularly this testing environment. Therefore both ways of calculation were applied in the second group of the tests.

For 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 Figure.

4.2. The results

The results showing the influence of the number of moving peaks on the offline error are presented in Figure 3.

It can be seen that the efficiency of the algorithm depends on the number of clones which means that it also depends on the number of antibodies in the population. These two parameters are related to each other to satisfy the constraint of the constant number of evaluations between changes as well as the constant number of iterations. Clearly the smaller the population of antibodies, the larger number of clones can be generated and processed. Figure 3 shows that the least values of offline error were obtained for smaller number of clones i.e. for larger populations of antibodies.

When we take a look at the dependency between the number of moving peaks and the error we can see that the least value of the error was obtained for largest numbers of moving peaks, albeit there are also very good results for the case with just two moving peaks.

For better understanding of the observations presented in Figure 3 and especially the dependency between the number of moving peaks and the value of offline error an additional set of tests was performed. We observed current value of offline error which was evaluated continuously for subsequent changes in the fitness landscape in a single experiment. Eventually, we selected four sets of the experiment parameters’ settings for further research: two of them for the best results of the error and two – for the worst ones. We selected the following two representatives for a group of the experiment parameters’ settings with the best value of offline error: the first, with 20 antibodies in the population and just two moving peaks in the fitness landscape (for brevity identified in the Figures as 20;2) and the second with 32 antibodies in the population and 48 moving peaks in the fitness landscape (32;48). A group of the experiment parameters’ settings where the error was the biggest consisted of the case with 4 antibodies in the population and 20 moving peaks in the fitness landscape (4;20) and the case with 6 antibodies in the population and 8 moving peaks in the fitness landscape (6;8).

Additionally, we measured the value of the offline error
For these four experiments in two ways each. The first way was based on evaluation of the error from the beginning of the experiment while in the second way the error was started to be evaluated after 10-th change in the fitness landscape. The graphs with the error evaluated in the first way are presented in Figure 4 and with the error evaluated in the second way – in Figure 5.

When we compare the Figures 4 and 5 it can be seen that the first phase of the search process significantly influences the final value of the offline error. For the same experiment the values of the error evaluated with and without taking into account first 50,000 evaluations of the fitness function (i.e. first 10 changes of the fitness landscape) differ by the value from 0.5 to 1. When we take a look at the value of offline error evaluated for randomly generated population of antibodies and for the population after two hundred iterations, the difference is much higher. The precise values of initial and final values of offline error for all the cases presented in Figures 4 and 5 can be found in Table 1. In case of evaluation started from the beginning of the experiment the first value in the Table’s cell represents the error returned after the evaluation of the initial population. In case of evaluation started from the 10-th change, the first value represents the error returned after reevaluation of the population just after the change. In both cases the second
value in Table’s cell is the value of the error returned at the end of the experiment.

Table 1. First and final values of offline error calculated in two ways: when evaluation started from the beginning of the search process and from the 10-th change (presented in Figures 4 and 5).

<table>
<thead>
<tr>
<th>ev. started from.</th>
<th>.10-th change</th>
<th>..the beginning</th>
</tr>
</thead>
<tbody>
<tr>
<td>20;2</td>
<td>3.57⇒3.55</td>
<td>165.88⇒4.48</td>
</tr>
<tr>
<td>32;48</td>
<td>8.68⇒4.67</td>
<td>49.79⇒5.27</td>
</tr>
<tr>
<td>4;20</td>
<td>15.56⇒12.89</td>
<td>180.28⇒13.34</td>
</tr>
<tr>
<td>6;8</td>
<td>8.82⇒8.23</td>
<td>164.4⇒8.67</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper the results of two sets of experiments are presented. We focused on two aspects: what are the final values the offline error obtained for different numbers of moving peaks in the fitness landscape and what do the changes of the offline error during the search process in a single experiment look like.

The first conclusion drawn from the first group of experiments is concerned with the number of antibodies in the population of BCA. The population should be as big as possible to make not more than 10 clones for each of the antibodies from one side and satisfy all the assumed constraints on the number of fitness function evaluations between subsequent changes from the other. This conclusion becomes clearer when we return to the last conclusion in [7] where authors wrote about the B-Cell Algorithm: "BCA can be viewed as a family of parallel hill-climbers. Namely each B-cell $x_i$ produces $c$ offspring, and only these offspring compete with a given B-cell for surviving. This is just the hill-climbing strategy: only the best neighbor of the current best solution can replace it (if it is better than that current solution). Here we observe the analogical behavior." Clearly for multimodal nonstationary landscape we need at least as many subpopulations as the number of moving peaks. Here we want to stress that there is nothing wrong with having more subpopulations than the number of moving peaks as far as we do not have to respect the constraint on the number of evaluations between subsequent changes. The constraint implies the fact that the more populations we have, the less fitness evaluations between subsequent changes is granted to a single subpopulation. So the search impact of a single subpopulation is less. Therefore unlimited increasing of the number of subpopulations can be treated as a wastefulness.

The next conclusion is simple but non-intuitive thus demands more complex explanation. The conclusion says: the best results were obtained for more complex environments i.e. with the bigger number of moving peaks. Fortunately, Table 1 helps to understand this observation. When the number of moving peaks grows, the average loss of quality of the recently found solutions just after the change in the fitness landscape is lesser than for environments with small number of moving peaks. Why? When we take a closer look at Figure 2, it can be seen that the fitness function for the case with just two moving peaks returns values from wider range than for 50 moving peaks. In the first case the range starts from -140 while in the second case – from -30. Please note that the Figure 2 represents the landscape for 2-dimensional search space while scenario 2 of the MPB requests to work on 5-dimensional search space. For higher number of dimensions this range stretches to the left (to the left only because the right limit of fitness values is defined in the MPB parameters and in case of scenario 2 it is within [30; 70]). So when we think that the difference between the test-cases is just in the number of moving peaks the truth is that they differ also in yet another way, i.e. by the range of possible fitness values obtained by the solutions which belong to the domain. The additional hidden difference described above changes the conditions of the experiments in an unexpected way. From one side the environments with higher number of moving peaks could have opinion of being more difficult but from the other – the returned offline error value says the opposite i.e. the value could be lesser. They can be lesser because in case of change in the fitness landscape chances for a significant loss of the solution’s value are much lesser than in case of the environment with just a few moving peaks. Therefore we could venture an opinion that for the MPB generator the results obtained for the test-cases differing just by the value of a parameter which controls number of moving peaks are not comparable to each other with the offline error measure. Simply we cannot be sure if the lesser offline error really means better following the running peaks.

The last conclusion is also known in the literature and tells that it is necessary to omit calculation of the value of offline error in the initial phase of the search process, otherwise the obtained final value would be falsified by very high values of the error obtained at the beginning. We showed that in case of scenario 2 of the MPB this falsification can be quite significant, i.e. approximately 10 to even 20 per cent in comparison to the results obtained when the error is not evaluated for the period of the first ten changes in the fitness landscape.
6. Acknowledgements

The author would like to thank Jarosław Stańczak (Systems Research Institute, Polish Academy of Sciences) for fruitful discussion which was an inspiration for this paper.

References


