ISSN: 1803-3814

REGARDING THE BEHAVIOR OF BISON RUNNERS WITHIN THE BISON ALGORITHM

Anezka Kazikova, Michal Pluhacek, Roman Senkerik

Faculty of Applied Informatics
Tomas Bata University in Zlin
T.G.Masaryka 5555, 760 01 Zlin, Czech Republic
{kazikova, pluhacek, senkerik}@utb.cz

Abstract: *This paper proposes a modification of the Bison Algorithm's running technique, which allows the running group to exploit the areas of discovered promising solutions. It also provides a closer examination of the successful running behavior and its impact on the overall optimization process. The new algorithm is then compared to other optimization algorithms on the IEEE CEC 2017 benchmark solving continuous minimization problems.*

Keywords: *bison algorithm, bison seeker algorithm, optimization, swarm algorithms*.

1 Introduction

A recent trend in the artificial intelligence is to take inspiration out of natural sources like the genome structure [1], evolution [2], or even animal behavior patterns [3] and use it for optimization. Particularly popular are the swarm algorithms - based on the collective intelligence phenomenon. Thanks to the inspiration from various occasions like the mating of fireflies [4], hunting techniques of wolf packs [5], or even cuckoos laying eggs [6], many successful optimization techniques were created. However, some of the swarm algorithms lean towards the premature convergence, since many algorithms rather emphasize exploitation at the expense of exploration in the later optimization process.

To address this problem, the Bison Algorithm was proposed by Kazikova et al. in 2017, promoting the exploration phase in the same amount throughout the whole optimization process. The algorithm is based on a basic observation of the exploration and exploitation behavior of bison herds. The former follows the swarming manner of an endangered herd, while the latter finds inspiration in the bison running habit. Both of these practices happen in two separate groups simultaneously throughout the optimization course [7]. While the swarming group is mainly responsible for the exploitation and thus has a high impact on the overall convergence, the exploring running group secures the diversity of the population and keeps on searching the area to prevent the first group from getting stuck in local optima.

However, up to now research suggested, that the running group has rather a limited impact on the overall optimization process [7,8,9]. This discovery provoked a question of the actual influence on the population and whether the runners have, in fact, the potential to affect and improve the final solution.

In this paper, we examine the behavior of the successful running individuals and their impact on the bison population. We also propose a new running approach, that could inflict a greater influence of the successful running solutions, as it allows the running group to investigate the area of the promising solutions.

The paper is structured as follows: section 2 introduces the Bison Algorithm and proposes a new run and seek behavior. Section 3 outlines the experiments for simulating the success of the running group and describes the methods used to determine the effectivity of the proposed modification. Section 4 presents the achieved results of the success simulation and the comparison of the modified Bison Algorithm with other well-established swarm algorithms like the Particle Swarm Optimization [10] and the Cuckoo Search [6] on the IEEE CEC 2017 benchmark [11]. Finally, in section 5 we discuss the outcome of the experiments and future research is considered in section 6.

2 Bison Algorithm

2.1 Inspiration

The inspiration behind the Bison Algorithm comes from two behavior patterns of bison herds. When bison are in danger, they protect the weak by letting them swarm into the circle outlined by the strongest individuals. Bison herds are also a good example of an exploration model, as they are tireless runners [12].

2.1 Definition of the Bison Algorithm

The algorithm divides the population into two groups: the swarming and the running group. The swarming group exploits the search space, by computing the center of the strongest individuals and moving all of the swarming members closer to the center. Meanwhile, the running group runs through the search space in a slightly altering direction, exploring the undiscovered areas. When a runner finds a better solution than a swarmer, the runner gets promoted to the swarming group, abandoning the weaker solution, while the running group stays unchanged (Algorithm 1).

Algorithm 1: Pseudo code of the basic Bison Algorithm

```
Initialization:
  objective function: f(x)=(x1, ..., xd)
  generate the swarming group randomly
  generate the running group around xbest
  generate the run direction vector
for every iteration do
  bison in the swarming group swarm
  bison in the running group run
  copy successful runners to the swarming group
  sort the swarming group by f(x) value
end for
```

Swarming behavior. The swarming movement is described in Algorithm 2. It starts by computing the center of the strongest individuals in the swarming group (Eqs. 1,2). Their number is specified by the *elite group size* parameter. Then each swarming member computes a new position candidate in the direction to the center and moves there if it improves its objective function value (Eqs. 3,4).

$$weight = (10, 20, \ldots, 10 \cdot s) \tag{1}$$

$$ranked\ center = \sum_{i=1}^{s} \frac{weight_i \cdot x_i}{\sum_{j=1}^{s} weight_i} \tag{2}$$

$$direction = center - x_{old} \tag{3}$$

$$x_{new} = x_{old} + direction \cdot rand(0, overstep)_D \tag{4}$$

Where:
- s is the *elite group size* parameter,
- x_i, x_{old}, and x_{new} represent the i[th] and current solutions, and the new solution candidate,
- *rand(from, to)* is a random number in the range of the two given arguments,
- *overstep* is a parameter defining the maximum length of the swarming movement,
- D represents the dimensionality of the problem.

Algorithm 2: Pseudo code of the swarming behaviour

```
compute the center of elite bison group (Eqs. 1,2)
for every bison in the swarming group do
  compute new position candidate xnew (Eqs. 3,4)
  if f(xnew) < f(xold)
    then move to xnew
```

Running behavior. The running movement comes from the run direction vector, randomly generated during the initialization (Eq. 5) and subtly altered in each iteration (Eq. 6). Unlike the swarmers, the runners do not mind the solution quality when moving (Eq. 7). In the basic Bison Algorithm [8], the runners who find a better solution than a swarmer, copy their solutions into the swarming group. While the worse swarming solutions are abandoned, the original successful runners stay in the running group, and they move in the next iteration as expected.

$$run\ direction = rand\left(\frac{ub-lb}{45}, \frac{ub-lb}{15}\right)_D \tag{5}$$

$$run\ direction = run\ direction \cdot rand(0.9, 1.1)_D \tag{6}$$

$$x_{new} = x_{old} + run\ direction \tag{7}$$

Where:
- *rand(from, to)* is a random number in the range of the two given arguments,
- *ub* and *lb* are boundaries of the search space,
- D represents the dimensionality of the problem,
- x_{old}, and x_{new} represent the old and the new solutions.

Parameters and out of bound behavior. The algorithm operates on a hypersphere: when bison cross the borders of the search space, they appear on the other side of the dimension. The parameters of the algorithm are described in Table 1.

Table 1: Parameters of the Bison Algorithm

Parameter	Description	Recommended value
Population		50
Elite group size	No. of best solutions for center computation	20
Swarm group size	No. of bison performing the swarming movement	40
Overstep	The maximum length of the swarming movement 0 – no movement 1 – max to the center	3.5

2.2 Bison Seeker Algorithm

To address the question of the actual influence of the running group on the whole optimization process, we propose a modification of the Bison Algorithm called the Bison Seeker Algorithm. The modification has the same parameters and swarming behavior as the basic Bison Algorithm, and even the principles of the running movement based on the run direction vector coincide. However, it advances the behavior of the successful runners and adds them the possibility to exploit the promising solutions on their own as described in Algorithm 3.

Algorithm 3: Pseudo code of the Bison Seeker Algorithm

```
Initialization:
  objective function: f(x)=(x1, ..., xd)
  generate the swarming group randomly
  generate the run. group around xbest
  generate the run direction vector
  seek until=0
for every iteration do
  bison in the swarming group swarm
  for every bison in the running group do
    if iteration > seek until
      then run
    else if iteration == seek until
      then return runners to the original formation (Eq. 8)
    else
      seek
  end
  check boundaries
  if f(xrunner) < f(xswarmer) // running group is successful
    then copy xrunner to the swarming group
    seek until = iteration + 2
  sort the swarming group by f(x) values
end for
```

Run and seek behavior. When the running group finds a better solution than at least one member of the swarming group, the solution is copied to the swarming group, and runners become seekers for the following iteration. The seekers exploit the area around the promising solution with a behavior similar to the swarming movement with these alterations:
1) the center of the running group is computed only from the successful runners,
2) the overstep parameter of the seeking behavior is temporarily set to *overstep=2.0*. Since the running behavior happens without elitism, a higher overstep threshold would disband the seeking group instead of bringing them closer towards the solution.

After 1 iteration without any further progress, the seekers are set into their original formation and moved to the corresponding location (Eq. 8). Figure 1 shows the movement of the Bison Seeker Algorithm on a 2-dimensional Rastrigin's function.

$$x_{new} = x_{formation} + iteration \cdot run\ direction \tag{8}$$

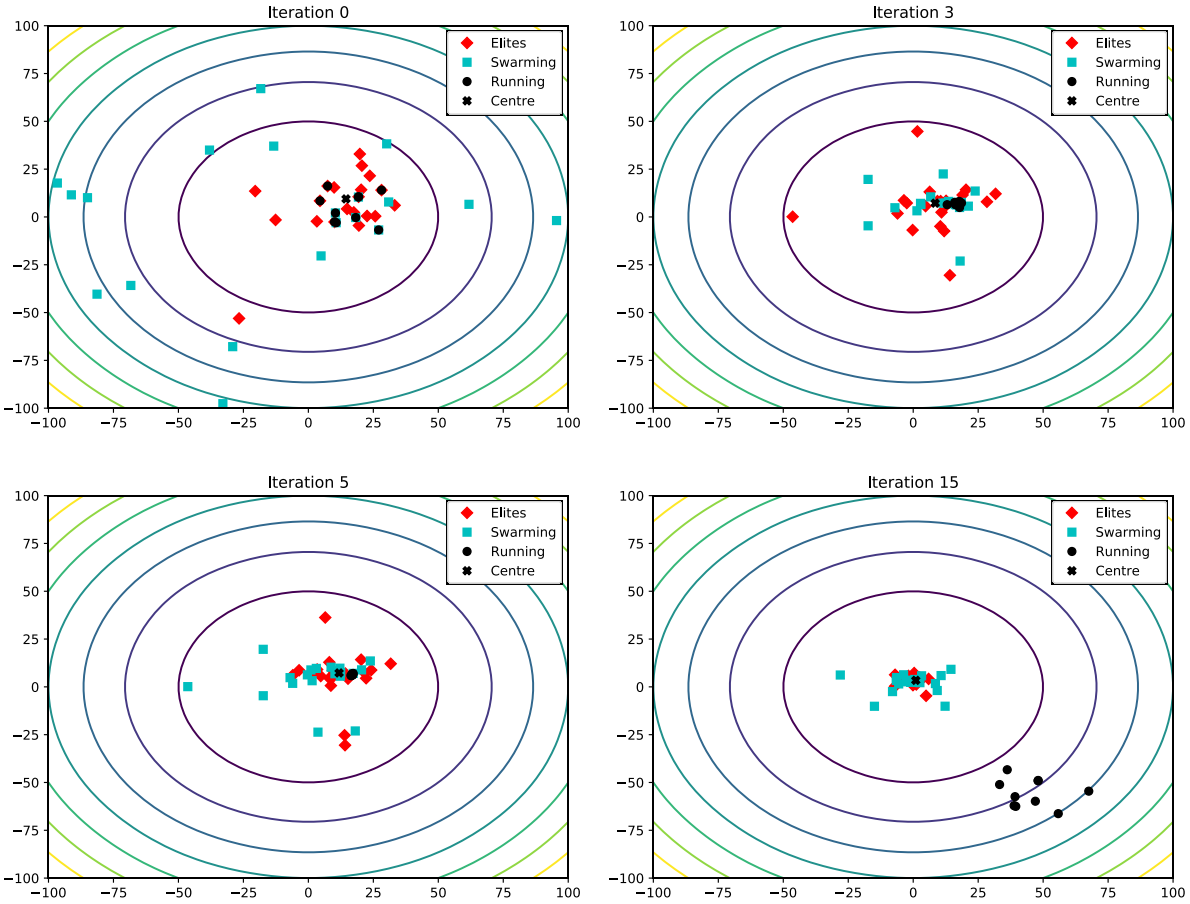

Figure 1: Movement of the Bison Seeker Algorithm on 2D Rastrigin's function

3 Methods

To investigate the behavior of successful runners and seekers, we generated the running group around the known optimum location at 1000th iteration similarly as at the initialization (Eq. 9). All the following experiments were carried out on 51 independent runs, each consisting of *n=10 000·dimension* evaluations of the objective function. The test bed of functions is described below (Eqs. 10-13).

$$runner = x_{opt} + rand\left(-\frac{ub-lb}{15}, \frac{ub-lb}{15}\right)_D \tag{9}$$

Where:

- x_{opt} is the known optimum location,
- *rand* is a random number,
- *ub* and *lb* are the boundaries of the search space,
- *D* represents the dimensionality of the problem.

The 2nd De Jong´s function (Rosenbrock´s valley).

$$f(x) = \sum_{i=1}^{dim-1} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \tag{10}$$

Function minimum: Position for E_n: $(x_1, x_2 \ldots x_n) = (1, 1, \ldots, 1)$. Value for E_n: $y = 0$.

Rastrigin's function.

$$f(x) = 10\ dim + \sum_{i=1}^{dim} x_i^2 - 10cos(2\pi x_i) \tag{11}$$

Function minimum: Position for E_n: $(x_1, x_2 \ldots x_n) = (0, 0, \ldots, 0)$. Value for E_n: $y = 0$.

Schwefel´s function.

$$f(x) = 418.9829 - \sum_{i=1}^{dim} -x_i \sin(\sqrt{|x|}) \tag{12}$$

Function minimum: Position for E_n: $(x_1, x_2 \ldots x_n) = (420.9687, 420.9687, \ldots, 420.9687)$. Value for E_n: $y = 0$.

Easom function.

$$f(x) = -\prod_{i=1}^{dim} cos(x_i) \cdot \sum_{i=1}^{dim} -(x_i - \pi)^2 \quad (13)$$

Function minimum: Position for E_n: (x_1, x_2…x_n) = (π, π,…, π). Value for E_n: y = -dim.

Table 2 presents the statistics of the obtained solutions during the success simulation experiments in the form: *mean solution ± standard deviation (optimum find rate)*. The optimum find rate indicates the percentage of all the runs, where the algorithm found the exact global optimum with error value < E-30. The optimum find rate of 0% points to the fact that the error value was higher than E-30 in all the runs, which is expectable with high dimensional problems. Table 3 shows all the cases of success simulation experiments, where one algorithm significantly outperformed the other one according to the Wilcoxon rank-sum test ($\alpha = 0.05$).

Figure 2 shows the mean convergence of the basic Bison Algorithm and the Bison Seeker modification comparing the standard runs with the simulated success experiment. Figure 3 presents the population distribution of a special case, which happened during the success simulation. The figure shows only 2 dimensions out of the 10-dimensional Rastrigin's problem.

The Bison Seeker Algorithm was then compared to other swarm optimization techniques like the Particle Swarm Optimization and the Cuckoo Search on the set of 30 functions of benchmark IEEE CEC 2017 [11]. Both these algorithms are well-established optimization techniques approved on many real-world problems [13,14]. The optimization algorithms are referenced as BIA (the basic Bison Algorithm), BSA (the Bison Seeker Algorithm), PSO (the Particle Swarm Optimization) and CS (the Cuckoo Search). Both the PSO and CS algorithms were implemented from the EvoloPy library [15] with their default parameters: PSO population = 50, $v_{max} = 6$, $w_{max} = 0.9$, $w_{min} = 0.2$, $c_1 = 2$, $c_2 = 2$; CS population = 50, Pa = 0.25. The BIA and BSA parameters were set in accordance with values recommended in Table 1.

Table 4 shows the algorithms, which significantly outperformed all of the other ones given the Wilcoxon rank-sum for 10, 30 and 50 dimensions ($\alpha = 0.05$) and Table 5 compares the average performance and the standard deviation of the algorithms.

Finally, Figure 4 shows the mean solution of the BSA, PSO, and CS on the IEEE CEC 2017 benchmark in 30 dimensions.

4 Results

Table 2: Performance of BSA and BIA on functions with simulated success (avg ± std (optimum find rate))

10 dimensions	**Bison Seeker Algorithm**	**Bison Algorithm**
Schwefel	342.8 ± 544.85 (55%)	1007.26 ± 627.48 (20%)
Rastrigin	3.31 ± 4.12 (2%)	3.42 ± 3.11 (8%)
Easom	-7.62 ± 2.83 (80%)	-6.47 ± 3.3 (59%)
Rosenbrock	0.91 ± 0.84 (0%)	1.06 ± 1.14 (0%)
30 dimensions		
Schwefel	578.43 ± 969.15 (18%)	3910.62 ± 1212.89 (0%)
Rastrigin	20.19 ± 7.08 (0%)	22.44 ± 17.65 (0%)
Easom	-9.37 ± 11.1 (22%)	-6.79 ± 9.29 (14%)
Rosenbrock	13.68 ± 4.45 (0%)	12.3 ± 3.31 (0%)
50 dimensions		
Schwefel	1152.8 ± 1094.59 (2%)	7562.9 ± 1501.15 (0%)
Rastrigin	47.79 ± 2580714 (0%)	50.32 ± 27.83 (0%)
Easom	-3.73 ± 6.48 (2%)	-5.35 ± 8.55 (2%)
Rosenbrock	34.2 ± 11.35 (0%)	36.97 ± 16.27 (0%)

Table 3: Winning algorithms on functions with simulated success (Wilcoxon rank-sum test α=0.05)

Dimension	**Schwefel**	**Easom**	**Rosenbrock**	**Rastrigin**
10 D	BSA	BSA	-	-
30 D	BSA	-	BIA	-
50 D	BSA	-	-	-

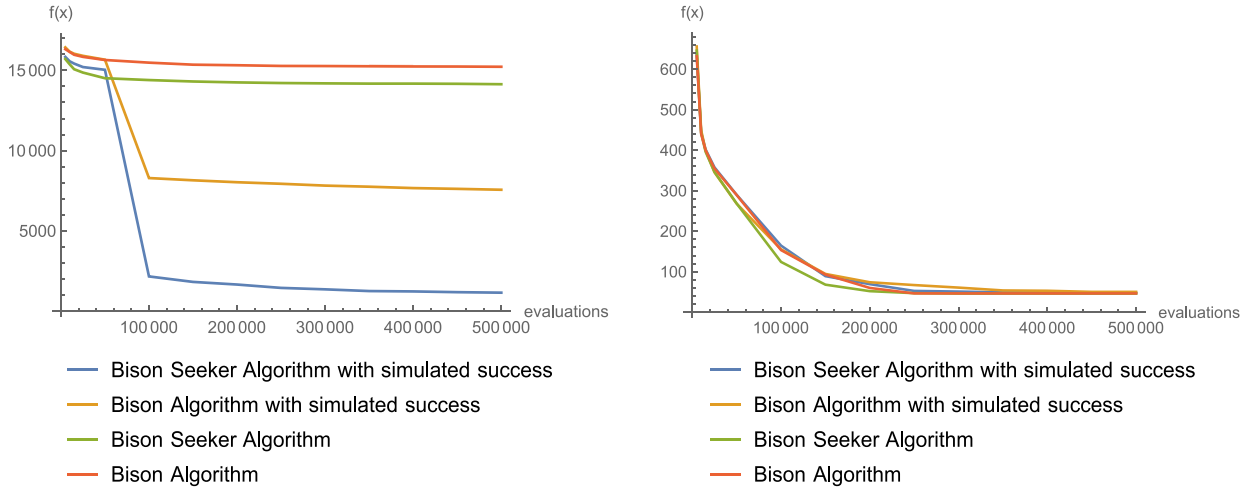

Figure 2: Mean convergence curves of BIA and BSA on 50D Schwefel's function (on the left) and 50D Rastrigin's function (on the right)

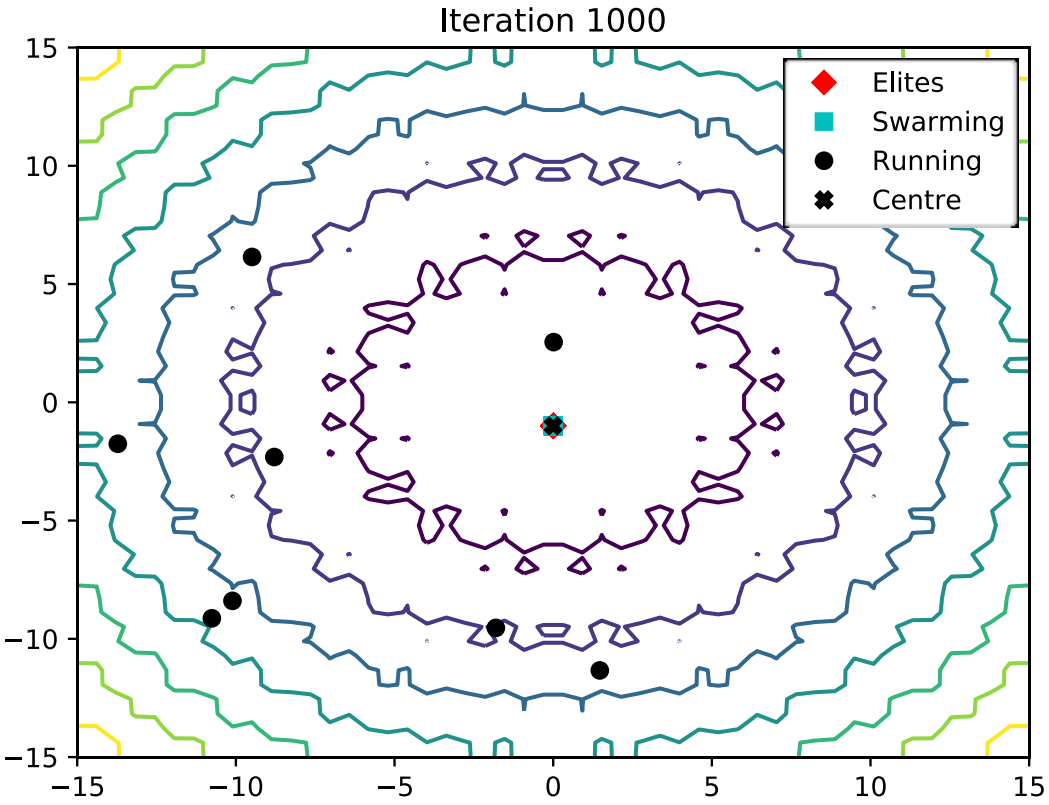

Figure 3: Distribution of 2 dimensions on 10D Rastrigin's success simulation

Table 4: Winner algorithms on CEC 2017

Dimension	None	BSA	PSO	CS
10 D	10	8	2	10
30 D	7	14	3	6
50 D	3	14	7	6

Table 5: Performance of BSA, PSO and CS on benchmark IEEE CEC 2017 in 30D

	BSA		PSO		CS	
	avg	*std*	*avg*	*std*	*avg*	*std*
f_1	2.15E+03	2.66E+03	3.91E+03	5.11E+03	8.90E+02	7.49E+02
f_2	1.06E+12	5.00E+12	5.02E+23	3.58E+24	2.85E+11	6.21E+11
f_3	7.45E+01	8.59E+01	3.42E-04	6.75E-04	3.39E+04	5.76E+03
f_4	1.36E+01	2.39E+01	9.31E+01	2.89E+01	6.82E+01	1.89E+01
f_5	6.54E+01	5.98E+01	1.52E+02	2.82E+01	1.37E+02	2.00E+01
f_6	9.34E-04	3.16E-03	3.06E+01	9.78E+00	4.15E+01	9.07E+00
f_7	1.79E+02	3.73E+01	1.02E+02	2.21E+01	1.63E+02	2.11E+01
f_8	6.34E+01	5.92E+01	1.06E+02	1.90E+01	1.36E+02	1.76E+01
f_9	4.53E+00	5.61E+00	2.21E+03	9.68E+02	3.72E+03	1.14E+03
f_{10}	6.92E+03	5.70E+02	3.51E+03	6.38E+02	3.74E+03	2.57E+02
f_{11}	2.79E+01	2.53E+01	1.00E+02	3.55E+01	8.84E+01	1.82E+01
f_{12}	2.89E+04	1.87E+04	6.57E+05	2.05E+06	1.50E+05	5.39E+04
f_{13}	1.19E+04	1.18E+04	1.03E+05	6.20E+05	5.91E+03	4.01E+03
f_{14}	4.40E+03	3.52E+03	5.08E+03	5.80E+03	1.18E+02	1.84E+01
f_{15}	3.66E+03	4.49E+03	1.06E+04	1.28E+04	9.92E+02	6.81E+02
f_{16}	8.79E+02	4.71E+02	8.49E+02	2.54E+02	9.25E+02	1.35E+02
f_{17}	1.56E+02	1.34E+02	5.21E+02	1.90E+02	3.01E+02	8.57E+01
f_{18}	1.72E+05	1.61E+05	1.53E+05	1.55E+05	5.57E+04	1.50E+04
f_{19}	4.84E+03	6.15E+03	4.86E+03	6.42E+03	3.82E+02	2.91E+02
f_{20}	2.38E+02	1.87E+02	1.83E+02	1.30E+02	4.13E+02	8.57E+01
f_{21}	2.66E+02	5.71E+01	3.37E+02	2.85E+01	3.25E+02	4.04E+01
f_{22}	1.00E+02	5.90E-01	1.98E+03	1.96E+03	8.29E+02	1.52E+03
f_{23}	3.79E+02	1.24E+01	6.58E+02	1.04E+02	4.86E+02	2.73E+01
f_{24}	4.50E+02	2.36E+01	6.98E+02	6.72E+01	5.44E+02	4.98E+01
f_{25}	3.92E+02	1.26E+01	3.90E+02	4.03E+00	3.85E+02	1.27E+00
f_{26}	1.06E+03	5.48E+02	2.07E+03	1.60E+03	1.05E+03	4.45E+02
f_{27}	5.31E+02	1.20E+01	5.77E+02	5.06E+01	5.29E+02	7.43E+00
f_{28}	3.22E+02	4.31E+01	4.24E+02	4.38E+01	3.87E+02	3.43E+01
f_{29}	5.65E+02	1.30E+02	9.34E+02	2.20E+02	9.28E+02	7.87E+01
f_{30}	4.15E+03	1.46E+03	5.19E+03	3.11E+03	1.10E+04	3.43E+03

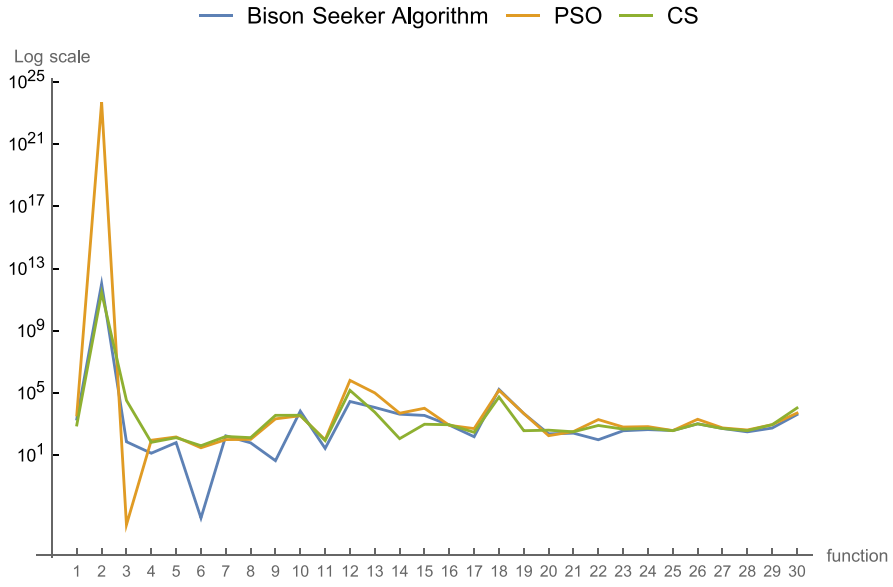

Figure 4: Mean solution quality on 30D CEC 2017

5 Discussion

The success simulation experiments revealed the supremacy of the Bison Seeker Algorithm over the basic Bison Algorithm on the Schwefel's function and 10-dimensional Easom function. When solving the 10-dimensional Schwefel's function, the Bison Seeker Algorithm was ultimately able to find the exact optimum at 55% of the tested cases.

Interestingly, in some functions, the success simulation did not seem to have the desired effect on the convergence of the algorithm (Figure 2). Closer investigation of the population distribution showed that in these cases the swarming

group was already stuck in a local optimum quite close to the global one. Therefore the much wider running group had no chance of affecting the final result even with the new seeking behavior, as illustrated in Figure 3. This phenomenon might transcend the Rosenbrock's and Rastrigin's function and point to a general issue, that the original wide formation of the running group may be unable to change the course of the optimization when the population is stuck in a local optimum very close to the global one.

However, this clearly does not concern the Schwefel's function, which operates on a considerably wider search space than the other tested functions (from -512 to 512 instead of the general –100, 100). It should also be pointed out, that without the success simulation, none of the Bison Algorithms outperformed the other.

The comparison of BSA, PSO and CS showed promising results of the Bison Seeker Algorithm on a wider sample of functions. The BSA significantly outperformed the other algorithms on 14 functions out of 30 in both 30 and 50 dimensions according to Wilcoxon rank-sum test (α=0.05). Especially well it performed on F6 and F9, where it carried out very stable results even in high dimensions. The Cuckoo Search excelled when solving the 10-dimensional problems.

6 Conclusion

The proposed variation of the Bison Algorithm proved to be effective when solving the Schwefel's function and was very often able to find the exact optimum. Also, the overall performance of the Bison Seeker Algorithm in comparison with other optimization techniques like the Particle Swarm Optimization and the Cuckoo Search proved the superiority of the Bison Seeker Algorithm when solving the 30 and 50-dimensional IEEE CEC 2017 benchmark problems.

The investigation of the success simulation brought up an interesting discovery concerning the range of the exploring group. While the wide distribution of the running formation seemed to have a beneficial effect when solving the larger Schwefel's function, it might have moderated the impact of the running group on the overall success of the Rastrigin's function. Restricting the seeking behavior on a narrower space around the promising solution, or an adaptive range of the running formation might help to resolve this problem. Addressing this phenomenon should direct our future research.

Acknowledgement: This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO), and Action IC1406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet). The work was further supported by resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

References

[1] Goldberg, DE, Holland, JH.: Genetic algorithms and machine learning. Mach Learning.3(2): pp. 95-9. (1988)
[2] Back, T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press (1996)
[3] Chakraborty, A, Kar, AK.: Swarm intelligence: A review of algorithms. In: Nature-Inspired Computing and Optimization, pp. 475-94. Springer (2017)
[4] Yang, X.: Firefly algorithm, Levy flights and global optimization. In: Research and development in intelligent systems XXVI, pp. 209-18. Springer (2010)
[5] Mirjalili, S, Mirjalili, SM, Lewis, A.: Grey wolf optimizer. Adv Eng Software. 69: pp. 46-6. (2014)
[6] Yang, X.-S., Deb, S. : Cuckoo search via Levy flights. In: Proc. Of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), December 2009, India, pp. 210-214. IEEE Publications, USA (2009)
[7] Kazikova, A., Pluhacek, M., Viktorin, A., Senkerik, R.: Proposal of a new swarm optimization method inspired in bison behavior. In: R. Matousek (ed.) Recent Advances in Soft Computing (MENDEL 2017), Advances in Intelligent Systems and Computing, Springer, in press.
[8] Kazikova, A., Pluhacek, M., Viktorin, A., Senkerik, R.: New Running Technique for the Bison Algorithm. In: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. Zurada (eds) Artificial Intelligence and Soft Computing, ICAISC 2018, Lecture Notes in Computer Science, vol 10841. Springer, Cham (2018)
[9] Kazikova, A., Pluhacek, M., Senkerik, R.: Performance of the Bison Algorithm on Benchmark IEEE CEC 2017. In: R. Silhavy (ed.) Artificial Intelligence and Algorithms in Intelligent Systems, CSOC2018 2018, Advances in Intelligent Systems and Computing, vol 764. Springer, Cham (2018)
[10] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, 4. (1995)
[11] Awad, N. H., Ali., M. Z., Liang, J. J., Qu, B. Y., Suganthan, P. N.: Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization, Technical Report. Nanyang Technological University, Singapore (2016)
[12] Berman, R.: American bison. Nature Watch, Lerner Publications, Minneapolis (2008)
[13] Pluhacek, M., Senkerik, R., Viktorin, A., Kadavy, T., Zelinka, I.: A review of real-world applications of particle swarm optimization algorithm. In: Lecture Notes in Electrical Engineering, pp. 115-122. ISSN 1876-1100. Springer Verlag, Ho Chi Minh City (2018)
[14] Mohamad, A., Zain, A. M., Bazin, N. E. N., Udin, A.: Cuckoo search algorithm for optimization problems-a literature review. In: Applied Mechanics and Materials, Vol. 421, pp. 502-506. Trans Tech Publications (2013)
[15] Faris, H., Aljarah, I., Mirjalili, S., Castillo, P., Merelo, J.: EvoloPy: an open-source nature-inspired optimization framework in Python. In: Proceedings of the 8th International Joint Conference on Computational Intelligence (IJCCI 2016), volume 1: ECTA, pp. 171-177. (2016)