Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

ANNALES UMCS INFORMATICA DOI: 10.17951/AI.2016.16.2.32

FLC control for tuning exploration phase in bio-inspired metaheuristic

Kazimierz Kiełkowicz
Department of Computer Science,
Faculty of Electrical and Computer Engineering,
Cracow University of Technology, Cracow, Poland

Damian Grela
Department of Computer Science,
Faculty of Electrical and Computer Engineering,
Cracow University of Technology, Cracow, Poland

Growing popularity of the Bat Algorithm has encouraged researchers to focus their work on its further improvements. Most work has been done within the area of hybridization of Bat Algorithm with other metaheuristics or local search methods. Unfortunately, most of these modifications not only improves the quality of obtained solutions, but also increases the number of control parameters that are needed to be set in order to obtain solutions of expected quality. This makes such solutions quite impractical. What more, there is no clear indication what these parameters do in term of a search process.

In this paper authors are trying to incorporate Mamdani type Fuzzy Logic Controller (FLC) to tackle some of these mentioned shortcomings by using the FLC to control the exploration phase of a bio-inspired metaheuristic. FLC also allows us to incorporate expert knowledge about the problem at hand and define expected behaviors of system – here process of searching in multidimensional search space by modeling the process of bats hunting for their prey.

Bat algorithm, swarm intelligence, metaheuristics, optimization, fuzzy logic, Mamdami-Type inference system

I. INTRODUCTION

In general metaheuristics algorithms can be divided into few groups, e.g. algorithms based on evolutionary approach that models evolutionary process or algorithms exploring phenomena of a Swarm Intelligence [1]. Others approach for evolutionary metaheuristic, such as algorithms for modeling response of a human immune system (e.g. Artificial Immune System algorithms) might be considered as separate category due to their multiplicity of proposed solutions.

Metaheuristics methods which are focused on exploring models of a natural evolution are (mostly but not limited to) as follows: Genetic Algorithms (GA) [2], Genetic Programming (GP) and Differential Evolution (DE) [3]. Algorithms based on Swarm Intelligence are broadly presented by Particle Swarm Optimization (PSO) [4], Ant Colony Optimization (ACO) [5] or some of its modifications.

Recently introduced method, based on population of solutions which explore phenomena of Swarm Intelligence was presented by Yang [6] in 2010 and it is called Bat Algorithm (BA). In [6] by modeling the behavior of bats hunting for prey and by exploring phenomena of their echolocation capabilities, author managed to incorporate methods for balancing the exploration phase as well as exploitation phase of a modern Swarm Based Algorithms.

Bat Algorithms had already been applied to solve numerous hard optimization problems such as multi-criteria optimization [7] or optimization of topology of microelectronic circuits [8].

Growing popularity of the Bat Algorithm has encouraged researchers to focus their work on its further improvements. Most work has been done within the area of hybridization of Bat Algorithm with other metaheuristics or local search methods [9]. Some other solutions were involved within the area of adding self-adaptability capabilities to algorithm [10]. Some works has also been in area of adaptation of standard Bat Algorithm for binary problems [11].

Unfortunately, most of these modifications not only improves the quality of obtained solutions, but also increases the number of control parameters that are needed to be set to obtain solutions of expected quality. This makes such solutions quite impractical.

This paper introduces fuzzy logic control system build on Mamdami-Type inference method to control the exploration and exploitation phase of an evolutionary system based on modified Bat Algorithm [12]. Application of fuzzy logic to control the exploration and exploitation phase frees the user from explicit specifying control parameters and only require to define expected behavior of an algorithm in human readable knowledge base form of *if-then* sentence.

Paper is organized as follows, in Section 2 basic scheme of the Bat Algorithm and some custom modifications are introduced and briefly discussed, Section 3 discuss the use of a fuzzy inference system to dynamically change algorithm parameters, Section 4 presents simulation experiments. Section 5 summarize presented results and discuss some concluding remarks.

II. BAT ALGORITHM AND ITS MODIFICATION

Bat Algorithm is recently proposed bio-inspired metaheuristics method for solving hard real valued optimization tasks. It tries to mimic behavior of bats hunting for their prey. Algorithm was introduced by Yang in 2010 [6]. Bat Algorithm is based on population of bats, which by flying thru solution search space explore it in order to find interesting areas. Each single bat represents one solution in n-dimensional search space. Solutions are evaluated in terms of their fit value by provided fit function.

For example, we can consider n-dimensional, real valued solution space in which optimization takes place. Each solution,

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

represented as a bat, is evaluated with provided fit function. There are also two real valued n-dimensional vectors associated with each bat in population. First vector is real valued vector representing position of a bat in solution search space. Second vector is real valued vector representing velocity in each of n-dimensional directions. Usually position vector and velocity vector are initialized randomly at the beginning of the algorithm. Main loop of the algorithm consists of iterative improvement in founded solution. At each iteration step fit value is calculated for every member of population of bats by provided fit function, and new velocity vector is calculated based on relative distance from best and current solution in population. Next, position of every bat is updated accordingly to its velocity vector. At the end of each iteration best solution is founded and used as new reference point. Exploring search space continues until some termination conditions are satisfied. Usually these conditions are the maximum number of iterations or improvements in the best solution. As a result, after satisfied stop conditions, the best solution is returned. Pseudo code for Bat Algorithm is listed in Fig.1.

1:	Randomly initialize position x_i and velocity v_i of i-th bat in population
2:	Initialize pulsation frequency $Q_i \in [Q_{min}, Q_{max}]$, pulsation r_i and loudness A_i of i-th bat in population
3:	**while not** termination conditions are satisfied:
4:	**for_each** bat in population:
5:	$v_i(t) = v_i(t-1) + Q_i(x_i(t-1) - x^*)$ $x_i(t) = x_i(t-1) + v_i(t)$
6:	**if randn(0,1)** $> r_i^t$: Generate new solution around current bests solutions
7:	Generate new solution by flying randomly
8:	**if randn(0,1)** $< A_i^t$ **and** $f(x_i) < f(x^*)$: Accept new solution and update pulsation and loudness factors r_i^t and A_i^t as: $A_i^{t+1} \leftarrow \alpha A_i^t;\ r_i^{t+1} \leftarrow r_i^t(1 - \exp(-\gamma t))$
9:	Evaluate bats population using fit function f
10:	Find best bat in population and mark him as x^*

Fig. 1. Bat Algorithm.

where:
$v_i(t)$ - real valued velocity vector of i-th bat,
$x_i(t)$ - real valued position vector of i-th bat,
Q_i - pulsation frequency of i-th bat,
$\alpha, \gamma, Q_{min}, Q_{max}$ - constant.

Equations used for bat position and velocity update, used in algorithm 1 step 5, were introduced in [6].

A. Modification to Bat Algorithm

An important aspect of a population-based metaheuristic is balance between exploration and exploitation phase of a search process. Exploration (sometimes called diversification) is responsible for global search capability. While, in contrast, exploitation (sometimes called intensification) is responsible for local search ability of algorithm. As it was pointed out in [13] Bat algorithm is powerful at exploitation but has some insufficiency at exploration phase. In our opinion Bat Algorithm also suffer from lack of memory of best solution found during the time of optimization what in effect sometimes cause bats to escape from promising area of solutions search space. Bat Algorithm also tends to direct bats outside of the solution search space box. Yang in [6] proposed to use upper bound limits on position vector to overcome these limitations. Bat Algorithm also too often tends to accept solution of worse fit value.

Few modifications to Bat Algorithm has been proposed in literature. In [14] Inertia Weight Factor Modification relative to current iteration and max iteration and Adaptive Frequency Modification based on relative bat distance to best solution has been introduced. In [15] dynamic and adaptively adjustment of a bat speed and flight direction has been examined. Self-adaptive capability has also been examined in [10].

Bat Algorithm has also been hybridized with Harmony Search Algorithm [13] or with Differential Evaluation Algorithm [9]. In [16] Bat Algorithm with self-adaptation of control parameters has been hybridized with different DE strategies as local search heuristics. However there are no systematic solutions to previously mentioned problem hence proposed modifications.

Modifications to Bat Algorithm introduced by Kiełkowicz and Grela in [12] are twofold: scheme of acceptance of a new solution, and velocity equation is modified to overcome some mentioned limitation. Introduced modifications are summarized in pseudo code listing in Fig.2. Memory of best solution found during the process of optimization by the algorithm is also introduced.

1:	Randomly initialize position x_i and velocity v_i of i-th bat in population
2:	Initialize pulsation frequency $Q_i \in [Q_{min}, Q_{max}]$, pulsation r_i and loudness A_i of i-th bat in population
3:	**while not** termination conditions are satisfied: Q=fuzzyInferenceSystem(diversity, error, iteration)
4:	**for_each** bat in population:
5:	$v_i(t) = \alpha_i v_i(t-1) + Q_i\big(x^* - x_i(t-1)\big) +$ $+ Q_i(x^*_{ever} - x_i(t-1))$ $x_i(t) = x_i(t-1) + v_i(t)$
6:	**if randn(0,1)** $> r_i^t$: $x_i' \leftarrow$ generate new solution around current bat x_i
7:	**if** $f(x_i') < f(x_i)$ **or randn(0,1)** $< A_i^t$: $x_i \leftarrow x_i'$ Update values of pulsation and loudness, respectively r_i^t and A_i^t as: $A_i^{t+1} \leftarrow \alpha A_i^t;\ r_i^{t+1} \leftarrow r_i^t(1 - \exp(-\gamma t))$
8:	Evaluate bats population using fit function f
9:	Find best bat in population and mark him as x^*
10:	**if** $f(x^*) < f(x^*_{ever})$:
11:	$x^*_{ever} \leftarrow x^*$

Fig. 2. Modification of Bat Algorithm.

Modifications introduced in [12] also change bat position and velocity update equations. In comparison with equations presented in [6], use of an archive component to help direct bats towards area where good solutions were used to be known; and concept of cognition coefficients instead of using upper bounds limits is used in [12]. Finally, equations (1) and (2) shows introduced modification:

$$v_i(t) = \alpha_i v_i(t-1) + Q_i\big(x^* - x_i(t-1)\big) + \\ + Q_i(x^*_{ever} - x_i(t-1)) \quad (1)$$

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

$$\alpha + \beta = \chi. \quad (1)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (2)$$

where:
α_i - cognition coefficient of i-th bat,
$x^* - x_i(t-1)$ - social component,
$x^*_{ever} - x_i(t-1)$ - archive component,
Q_i - pulsation frequency of i-th bat.

In comparison to equations proposed by Yang in [6] modified velocity equation (1) is using cognition coefficients to limit the influence of past direction (taken at time t-1) at the decision taken at current t iteration. There is also archive component that helps bats build social knowledge of the previously, globally found best solution.

Proposed modification to the scheme of acceptance of new solutions are tend to limit the probability of acceptance of worse solution. Comparing original Bat Algorithm with modification in [12] the worse solution is accepted with probability A_i where in modified algorithm worse solution is accepted only with probability $(1 - r_i)A_i$. There is obvious relation that, satisfying that $r_i > 0$ and $A_i > 0$, the following relation is true $(1 - r_i)A_i < A_i$. Moreover, modifications introduced in [12] also includes form of memory x^*_{ever} of a best solution ever found.

It is important that introduced modifications doesn't change computation complexity of the algorithm in the context of big O notation since these modifications are linear in nature and are not based on additional computation or evaluation of a fitness function.

III. Parameter adaptation with fuzzy logic

The dynamic of Modified Bat Algorithm is defined by position and velocity update equations (1) and (2). Pulsation frequency Q_i was chosen to be adjusted using fuzzy logic Mamdami-Type inference type system since this parameter has an influence on the movement of bats in the flock. Dynamical changes of parameter can improve overall performance of algorithm. However, it is not always possible to derive clear mathematical formula describing how parameters should be adopted during optimization process. However it is easier to describe expected behavior of an algorithm in form of an *if-then* sentence describing situation and expected behavior, e.g.: *"If iteration is small, then explore is intensive"* or *"if diversity is small and iteration is big, then explore is less"*.

The goal of these paper is to explore possibility of using fuzzy logic Mamdami-Type inference system to control exploration/exploitation phase of a Bat Algorithm. To build Mamdami-Type inference system it is required to: define input values (and their fuzzification methods), define linguistic variable and knowledge base in form of an *if-then* sentence and define output values (and their defuzzification method). In these paper as it was introduced in [17] we also use diversity of the flock, the error of the flock and the iterations themselves as input parameters. As our output parameter, we choose Q_{max}. We expect our input and output parameters to be in $[0, 1]$.

The diversity (dispersion) of the flock is defined by following equation (3):

$$diversity(t) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\sum_{j=1}^{D}\left(x_{ij}(t) - x_j^*(t)\right)^2} \quad (3)$$

It can be considered as an average Euclidean distance between each bat and bat representing best solution at the i-th iteration. Diversity measure degree of dispersion in the flock. When bats are close to each other the diversity is small. Diversity to be considered as input to fuzzy inference system needs to be normalized before, since input must be in $[0, 1]$. Equation (4) was used to normalize diversity:

$$normalizedDiversity(t) = \begin{cases} if\ minDiversity = maxDiversity, 0 \\ if\ minDiversity \neq maxDiversity, \frac{diversity(t) - minDiversity}{maxDiversity - minDiversity} \end{cases} \quad (4)$$

$$\alpha + \beta = \chi. \quad (1)$$

The error of the flock measures the difference between the flock and the best bat, by averaging the difference between the fitness of each bat and the fitness of the best bat. It is defined by following equation (5):

$$error(t) = \frac{1}{n}\sum_{i=1}^{n}(fitness(x_i) - fitness(x^*)) \quad (5)$$

$$\alpha + \beta = \chi. \quad (1)$$

Error in the flock to be considered as input to fuzzy inference system needs to be normalized, since we expect it to be in $[0, 1]$. Equation (6) was used to normalize error in the flock:

$$normalizedError(t) = \begin{cases} if\ minFitness = maxFitness, 1 \\ if\ minFitness \neq maxFitness, \frac{error(t) - minFitness}{maxFitness - minFitness} \end{cases} \quad (6)$$

$$\alpha + \beta = \chi. \quad (1)$$

From now on *normalizedDiversity(t)* and *normalizedError(t)* will be referred simply as *diversity(t)* and *error(t)* respectively and Q_{max} as Q.

For iteration to be considered as input to fuzzy logic inference system it needs to be normalized, we used formula (7):

$$Iteration = \frac{currentIteration}{maximumNumberOfIteration} \quad (7)$$

$$\alpha + \beta = \chi. \quad (1)$$

Knowledge base for Mamdami-Type inference is in form of a set of an *if-then* sentence, where *if* part is a premise and *then* part is conclusion. Each sentence is constructed using linguistic variables and (possibly) "*and/or*" connectors and hedges. In these paper, we consider three linguistic variables diversity, error and iteration as input to inference system and one output linguistic variable Q. Each variable can take linguistic values from set {small, big}. Input and output linguistic values are

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

fuzzy sets defined on interval $[0,1]$. Hence, we expect crisp input values and output to be in interval $[0,1]$.

IV. Simulation Experiments

To examine how exploration and exploitation phase can be controlled with fuzzy logic controller in bio-inspired metaheuristic few simulation experiments were performed.

Parameters of the algorithm were dynamically controlled by fuzzy Mamdami-Type inference system. First, we examine how exploration and exploitation can be controlled with knowledge-base and different input linguistic variable. It was done by examining how diversity in the flock change over time. More diversity means algorithm is in exploration phase, where less diversity in the flock can be determined as exploitation phase. To reduce influence of local search (line 7, algorithm 2) during experiments no local search $(r_i^t = 1)$ was conducted. Simulations were performed on well-known test functions, with computer running on Intel Core i5 class processor, with 8GB of RAM. Algorithm has been implemented in Java, using FuzzyLite [18] library. Section IV.A briefly introduce used test functions, section IV.B reports obtained results.

A. Test Functions

Experiments were performed on three well known and wildly accepted test function for continues real-valued optimization problems. Used test functions are: Sphere, Rastrigin and Rosenbrock [16]. In every equation, D will stand for dimension of the function and $\vec{x}$ is real valued vector in search space, $\vec{x} \in \mathcal{R}^D$.

First function was standard test function called Sphere. It is convex, unimodal simple test function for metaheuristics (8), with global solution at the point $\vec{x} = (0, 0, \dots, 0)$.

$$f_{Sphere}(\vec{x}) = \sum_{i=1}^{D} x_i^2 \tag{8}$$

$$\alpha + \beta = \chi. \tag{1}$$

(1)

Second function was Rastrigin's function. It is based on Sphere function (8) by adding sinusoidal modulation what results as Rastrigin function (9). It is multimodal non-linear function with global minimum at point $\vec{x} = (0,0, \dots, 0)$.

$$f_{Rastrigin}(\vec{x}) = 10D + \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i)) \tag{9}$$

$$\alpha + \beta = \chi. \tag{1}$$

(1)

Last was Rosenbrock's function (10) which has its global solution at point $\vec{x} = (0,0, \dots, 0)$. Rosenbrock's solution is located in wide parabolic shaped valley. This makes it very complicated point to reach by evolutionary methods. Rosenbrock function is unimodal for D=2,3 while it is multimodal for more dimensions [19]

$$f_{Rosenbrock}(\vec{x}) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \tag{10}$$

$$\alpha + \beta = \chi. \tag{1}$$

(1)

B. Experiments

To examine how exploration and exploitation phase can be affected by knowledge-base of the inference system and input variable we execute our algorithm on different test function with different knowledge-base, each time starting from the same initial population for one test function. Each time diversity and average population fit at iteration were reported.

Input linguistic variable iteration, diversity and error were defined accordingly with terms {big, small} as depicted in Fig.3-5:

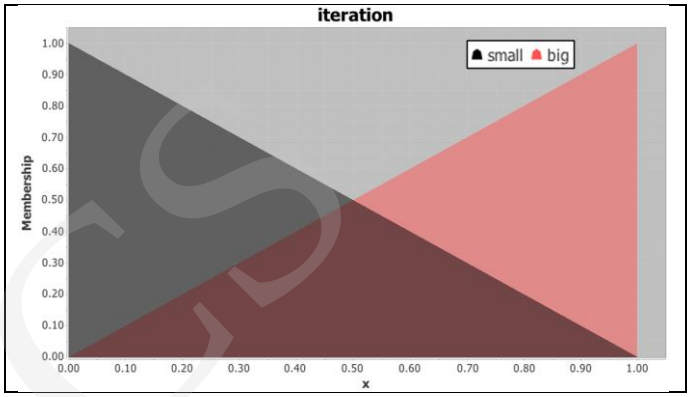

Fig. 3. Iteration input variable

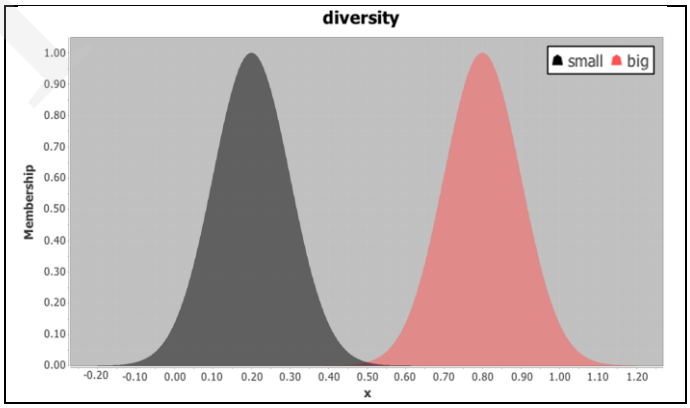

Fig. 4. Diversity input variable

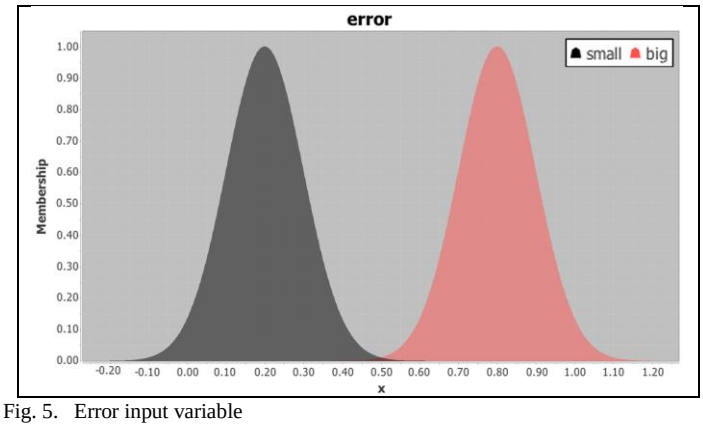

Fig. 5. Error input variable

Input linguistic variable Q was defined accordingly with terms {big, small} as depicted in Fig. 6:

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

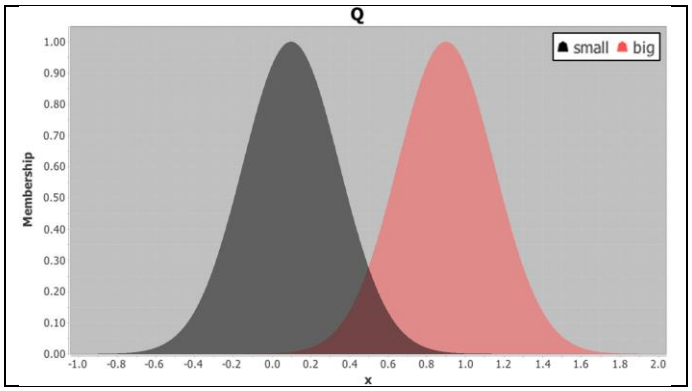

Fig. 6. Q output variable

Four Knowledge-base were examined:

KB1:= „if iteration is small then Q is big”,
„if iteration is big then Q is small”

KB2:= „if iteration is small then Q is small”,
„if iteration is big then Q is big”

KB3:= „if iteration is small or diversity is small then Q is big”, „if iteration is big or diversity is big then Q is small”

KB4:= „if diversity is small or error is small then Q is big”,
„if diversity is big or error is big then Q is small”

Corresponding control surface are depicted on Fig. 7.

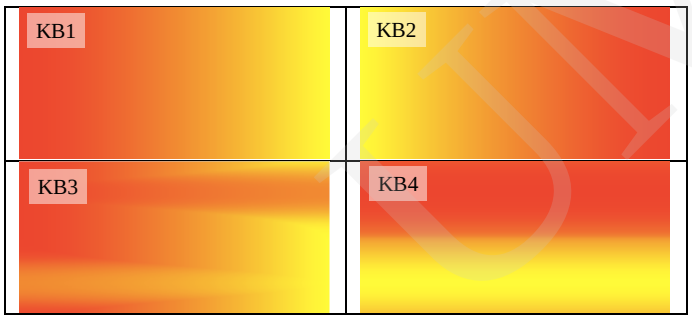

Fig. 7. KB{1,2,3,4} control surface

During the experiments, standard *max* function was chosen as “or” operator and *Centroid* method was chosen as defuzzification.

Results for Rastrigin function are presented in Fig.8-11.

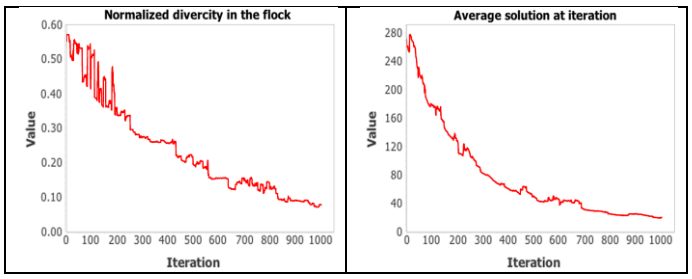

Fig. 8. Normalized diversity in the flock and average solution for KB1

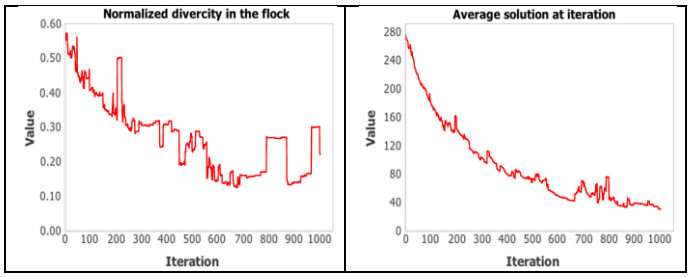

Fig. 9. Normalized diversity in the flock and average solution for KB2

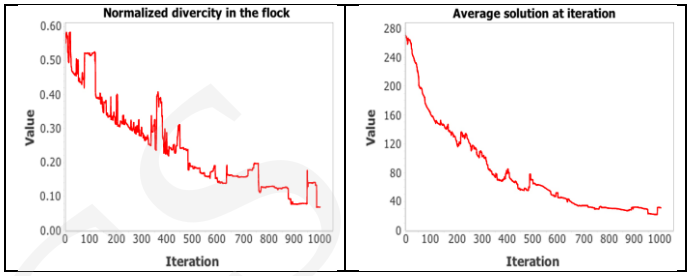

Fig. 10. Normalized diversity in the flock and average solution for KB3

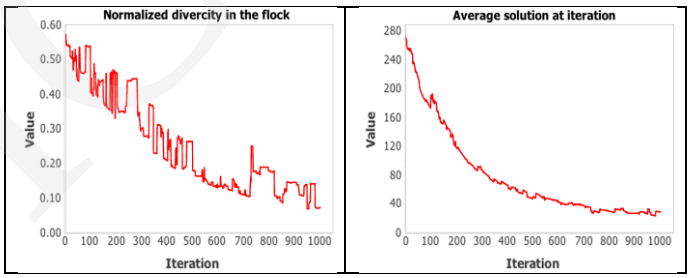

Fig. 11. Normalized diversity in the flock and average solution for KB4

Results for Rosenbrock function are presented in Fig.12-15.

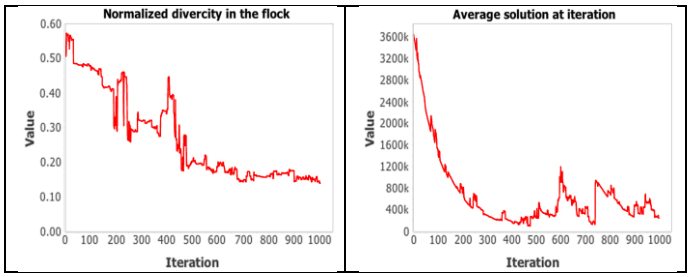

Fig. 12. Normalized diversity in the flock and average solution for KB1 for Rosenbrock function.

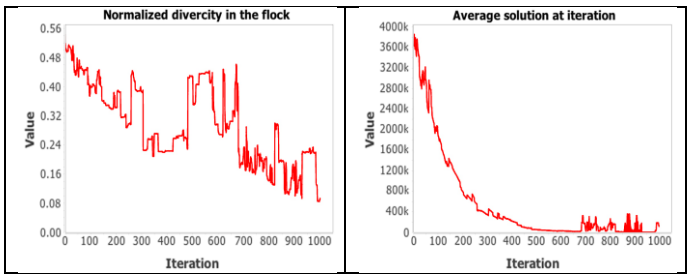

Fig. 13. Normalized diversity in the flock and average solution for KB2 for Rosenbrock function.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

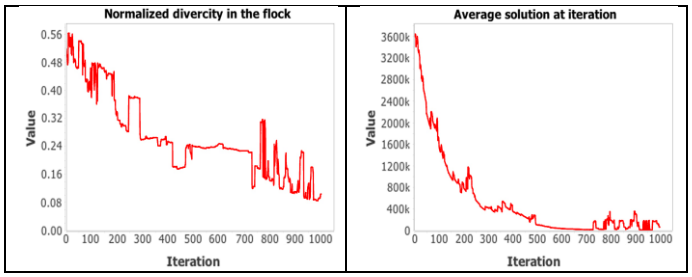

Fig. 14. Normalized diversity in the flock and average solution for KB3 for Rosenbrock function.

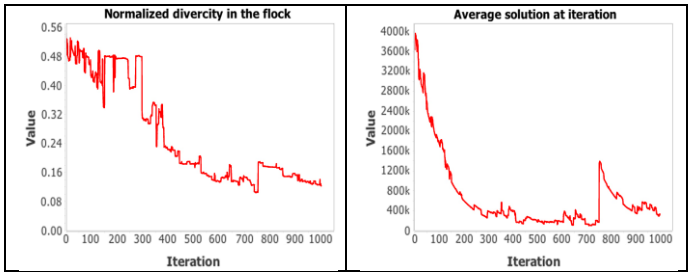

Fig. 15. Normalized diversity in the flock and average solution for KB4 for Rosenbrock function.

Results for Sphere function are presented in Fig.16-19.

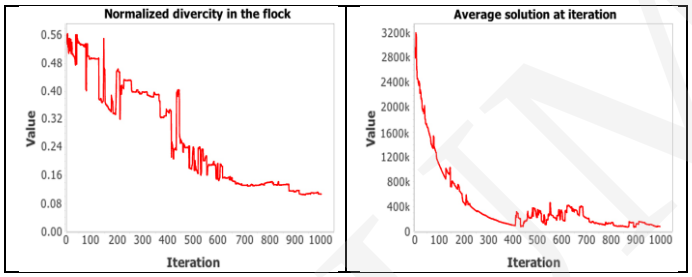

Fig. 16. Normalized diversity in the flock and average solution for KB1 for Sphere function.

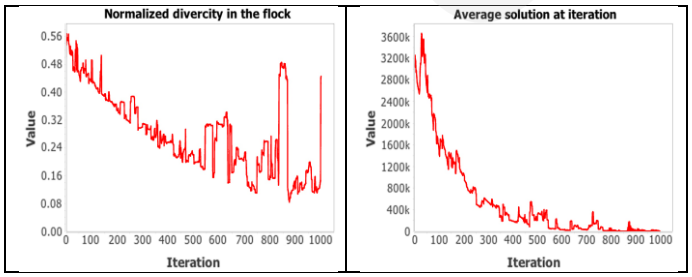

Fig. 17. Normalized diversity in the flock and average solution for KB2 for Sphere function.

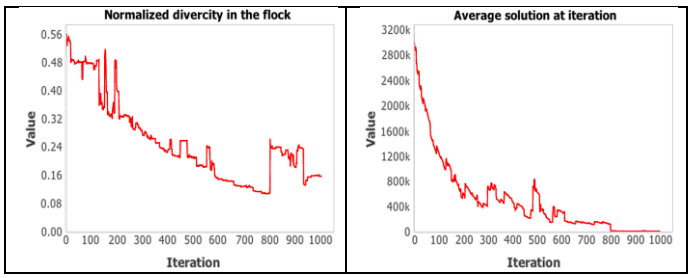

Fig. 18. Normalized diversity in the flock and average solution for KB3 for Sphere function.

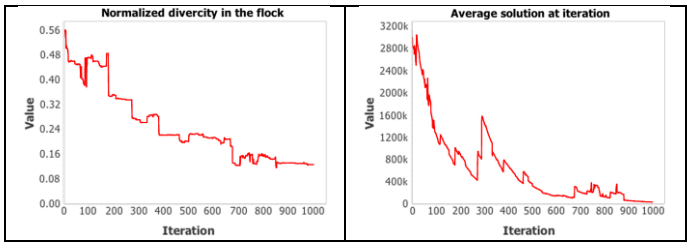

Fig. 19. Normalized diversity in the flock and average solution for KB4 for Sphere function.

Analyzing figures 8, 12, 16 that shows influence of knowledge-base KB1 on tested function let us see that flock, as iteration times go, tend to fly closer together. That behavior is coherent with our intuitive analysis and understanding of KB.

KB1:= „if iteration is small then Q is big”,
„if iteration is big then Q is small”

Analyzing figures 9, 13, 17 that shows effect of KB2 on the flock, opposite tendencies can be seen. Here, as iteration times go, diversity gets bigger. This kind of behavior is also coherent with our intuition on how KB might affect the flock.

KB2:= „if iteration is small then Q is small”,
„if iteration is big then Q is big”

Figures 10, 14, 18 and 11, 15, 19 shows how KB3 and KB4 respectively affect the diversity as iteration time goes.

KB3:= „if iteration is small or diversity is small then Q is big”, „if iteration is big or diversity is big then Q is small”

KB4:= „if diversity is small or error is small then Q is big”, „if diversity is big or error is big then Q is small”

KB3 and KB4 tends to increase diversity in the flock as algorithm getting closer to stop criteria (max. iterations). To verify how considered knowledge-base KB1, KB2, KB3 and KB4 affect algorithm capability to find solution and how they affect mean solution (and standard deviation of solutions) algorithm was rerun 50 times for each knowledge-base. Each time starting points (bats locations and their velocities) were initialized randomly (with unit distribution) within search space. Simulations were performed for Sphere, Rastrigin and Rosenbrock function in D=25-dimensional real valued cube with limitation [-10, 10] on each side. For each test 100 bats were used, no local search ($r_i^t = 1$) were performed during experiments. Termination condition was set to 1000 iteration. Obtained results are reported in Table 1 and 2.

TABLE I. MEAN AND STANDARD DEVIATION FOR SPHERE, RASTRIGIN AND ROSENBROCK FUNCTION USING KB1, KB2

Function	**KB1**		**KB2**	
	mean	***std***	***mean***	***std***
Sphere	0.00293	0.00333	0.00522	0.00484
Rastrigin	21.810	4.9913	36.76379	6.1945

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/01/2025 03:41:15

Rosenbrock	28.096	5.5234	35.916	6.0601

TABLE II. MEAN AND STANDARD DEVIATION FOR SPHERE, RASTRIGIN AND ROSENBROCK FUNCTION USING KB3, KB4

Function	**KB3**		**KB4**	
	mean	***std***	***mean***	***std***
Sphere	0.01657	0.01986	0.01744	0.02736
Rastrigin	37.525	8.3211	31.417	6.0132
Rosenbrock	35.704	6.5570	30.861	5.9652

Analyzing Table 1 and 2 it can be seen that KB1 generate solutions with less standard deviation within, where solutions found using KB2 vary from another. The biggest diversity in reported solutions were generated by KB3.

V. CONCLUSIONS

This paper examines usage of a Knowledge-base Mamdami-Type inference system to dynamically modify configuration parameters of a bio-inspired metaheuristic. Four Knowledge-base build using standard *if-then* sentence were considered and their effect on exploration and exploitation phase of the modified Bat Algorithm has been tested. This paper considers three linguistic variables as input: *diversity*, *error* and *iteration* in the flock to fuzzy inference system and one linguistic output *Q*. Each linguistic variable can take {small, big} as linguistic variable. Input and output linguistic variable are defined as fuzzy sets, with membership functions depicted as it is shown in Figures 3, 4, 5 and 6. Four different knowledge-base are considered and corresponding control surface is shown in Figure 7. Three well known and wildly accepted test function for continues real-valued optimization problems has been used to test how algorithm behave under different Knowledge-base. Used test functions are Sphere, Rastrigin and Rosenbrock [15]. Simulation experiments shows that it is possible to incorporate expert knowledge about the problem at hand and define expected behaviors of system in form of an *if-then* sentence. For example, if operator want (or need) algorithm to have exploration phase at the beginning and exploitation at the end, can use KB1. Figure 8, 12 and 16 depict how diversity of the flock (found solutions) changes during iteration under KB1. For each different test function (Rastrigin, Rosenbrock, Sphere) similar behavior can be observed. Initially (iteration is small) there is big changes in diversity of the solutions pool – which reflect exploration phase of an algorithm. As iteration times goes (iteration is big) diversity getting smaller – algorithm starts exploitation phase. Analyzing Table I and II with statistical properties of an algorithm (obtained after 50 rerun of an algorithm starting with different randomly initialed population), it can be seen that when search process is controlled with KB1 there is a less deviation in found solution than using KB2. It can be explained that KB1, as times goes, emphasize exploitation phase of a search process, where KB2, as times goes, emphasize exploration phase – hence bigger deviation within found solutions. Similar analysis can be conducted for KB3 and KB4 where algorithm dynamically change emphasizes of exploration or exploitation phase -back and for- during search process as needed accordingly to provided knowledge-base.

REFERENCES

[1] R. C. Eberhart, Y, Shi, “Empirical Study of Particle Swarm Optimization”, 1999.

[2] D. E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning”, Kluwer Academic Publishers, 1989.

[3] R. Storn, K. Price, “Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces”, Technical Report, 1995.

[4] J. Kennedy, R. C. Eberhart, “Particle swarm optimization”, In Proc. of IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[5] M. Dorigo, V. Maziezzo, A. Colorni, “The ant system: optimization by a colony of cooperating ants”, IEEE Trans. on Systems, Man and Cybernetics B, vol. 26, no. 1, 1996, pp. 29–41.

[6] X. S. Yang, “A New Metaheuristic Bat-Inspired Algorithm”, Nature Inspired Cooperative Strategies for Optimization, 2010, pp. 65-74.

[7] X. S. Yang, “Bat Algorithm for Multi-Objective Optimization”, International Journal of Bio-Inspired Computation, vol. 3, issue 5, 2011, pp. 267-274

[8] S. Fong, X. S. Yang, M. Karamanglu, “Bat Algorithm for Topology Optimization in Microelectronic Application”, International Conference on Future Generation Communication Technology (FGCT), IEEE, 2012, pp. 150-155.

[9] I. Fister Jr, D. Fister, X. S. Yang, “A Hybrid Bat Algorithm”, Elektrotehniski Vestnik, 2013, pp. 1-7.

[10] A. Baziar, A. A. Kavoosi-Fard, J. Zare, “A Novel Self Adoptive Modification Approach Based on Bat Algorithm for Optimal Management of Renewable MG”, Journal of Intelligent Learning System and Application, vol. 5, issue 1, 2013, pp. 11-18

[11] S. Mirjalili, S. M. Mirjalili, Xin-She Yang, “Binary Bat Algorithm”, Neural Computing and Applications, vol. 25, issue 3, 2014, pp. 663-681.

[12] K. Kiełkowicz, D. Grela, “Modified Bat Algorithm for Nonlinear Optimization”, International Journal of Computer Science and Network Security (IJCSNS), 2016, pp. 46-50.

[13] G. Wang and L. Guo, “A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization”, Journal of Applied Mathematics, vol. 2013, 2013, pp. 1-21.

[14] S. Yilmaz and E. U. Kucuksille, “Improved Bat Algorithm (IBA) on Continuous Optimization Problems”, Lecture Notes on Software Engineering, Vol. 1, No. 3, 2013, pp. 279-283.

[15] X. Wang, W. Wang, Y. Wang, “An Adaptive Bat Algorithm”, Lecture Notes in Computer Science vol. 7996, 2013, pp. 216-223.

[16] I. Fister Jr., S. Fong, J. Brest, and I. Fister, “A Novel Hybrid Self-Adaptive Bat Algorithm”, The Scientific World Journal, 2014, pp. 1-12.

[17] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, M. Valdez, “Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic”, Expert Systems with Applications, vol. 40, issue 8, 2013, pp. 3196–3206.

[18] Juan Rada-Vilela. FuzzyLite: A Fuzzy Logic Control Library, 2017. URL http://www.fuzzylite.com.

[19] W. Gao, S. Liu, “A Modified Artificial Bee Colony Algorithm”, Computers and Operations Research, vol. 39, 2012, pp. 687–697.