Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 2602 – 2629

doi:10.3906/elk-1909-12

Adaptive modified artificial bee colony algorithms (AMABC) for optimization of complex systems

Rabia KORKMAZ TAN[1,*], **Şebnem BORA**[2]

[1]Department of Computer Engineering, Çorlu Faculty of Engineering, Tekirdağ Namık Kemal University, Tekirdağ, Turkey

[2]Department of Computer Engineering, Faculty of Engineering, Ege University, İzmir, Turkey

Received: 02.09.2019 • **Accepted/Published Online:** 11.04.2020 • **Final Version:** 25.09.2020

Abstract: Complex systems are large scale and involve numerous uncertainties, which means that such systems tend to be expensive to operate. Further, it is difficult to analyze systems of this kind in a real environment, and for this reason agent-based modeling and simulation techniques are used instead. Based on estimation methods, modeling and simulation techniques establish an output set against the existing input set. However, as the data set in a given complex systems becomes very large, it becomes impossible to use estimation methods to create the output set desired. Therefore, a new mechanism is needed to optimize data sets in this context. In this paper, the adaptive modified artificial bee colony algorithm is shown to be successful in optimizing the numerical test function and complex system parameter data sets. Moreover, the results show that this algorithm can be successfully adapted to a given problem. Specifically, this algorithm can be more successful in optimizing problem solving than either the artificial bee colony algorithm or the modified artificial bee colony algorithm. The adaptive modified artificial bee colony algorithm performs a search in response to feedback received from the simulation in run-time. Because of its adaptability, the adaptive modified artificial bee colony algorithm is of great importance for its ability to find solutions to multiple kinds of problems across numerous fields.

Key words: Complex system, optimization, parameter setting, artificial bee colony, modified artificial bee colony, adaptive modified artificial bee colony

1. Introduction

Given that large-scale data sets and complicated structures are associated with complex systems, these kinds of systems cannot be solved through analytical methods. Complex systems comprise all the components involved in a given interaction, each of which necessitates an advanced level of information processing and each of which tends to respond to changes in the environment. As it is difficult to analyze complex systems in the real environment, agent-based modeling and simulation techniques are used instead [1–3]. Agent-based modeling and simulation are used as a design tool for analysis and newly installed systems and also to estimate the real system criteria[4].

The purpose of modeling a complex system is to identify its behaviors in order to establish how it will interact with any given input set. Agent-based modeling and simulation techniques establish an output set against the existing input set by using estimation methods and rule sets. As the data set of a complex system

*Correspondence: rkorkmaz@nku.edu.tr

increases, it becomes impossible to create the output set desired using estimation methods. Therefore, new mechanisms are needed in order to optimize the data sets of complex systems[5].

Simulation and modeling are not sufficient to use as an optimization technique by themselves. Therefore, optimization algorithms are frequently used in simulation and modeling[6]. Because of this, it has been proven that data sets defined as almost optimum can be obtained from big data spaces when metaheuristic algorithms (MHAs) are used to optimize solutions to simulation and modeling of complex systems[7, 8].

The artificial bee colony (ABC) algorithm is a swarm-based heuristic optimization algorithm that has been used in optimization studies. This algorithm was written by drawing on the food-seeking behavior of bees. The ABC algorithm has less critical parameters compared to many other optimization technique; therefore,the ABC algorithm is very easy to implement [9]. It is also flexible due to ease of hybridization with other optimization algorithms.

The ABC algorithm has been found to be successful in finding global optimization solutions since it was first introduced by Karaboga [10]. Thus, it has been widely used to solve both continuous and discrete optimization problems [11]. However, it has some drawbacks, such as suffering from early convergence and stagnation [10]. In order to overcome those drawbacks and make the ABC algorithm the most preferred algorithm, scholars are trying to improve the algorithm. Several versions of the algorithm have been developed for different problem solutions. Karaboga and Basturk compared the performance of the ABC algorithm with that of the genetic algorithm (GA), particle swarm optimization (PSO) algorithm, and particle swarm evolutionary algorithm (PS-EA) for multidimensional numeric problems. The results showed that the ABC algorithm outperforms the GA, PSO, and PS-EA. In their paper, the findings and analysis detail that the ABC algorithm can avoid a local minimum and it can be used for multivariable, multimodal function optimization. However, the algorithm still has slow convergence speed compared to population-based stochastic algorithms such as differential evolution (DE) and PSO. Karaboga applied the ABC algorithm to design infinite impulse response (IIR) filters. The performance of the ABC algorithm was compared to that of the nonlinear least-squares data fitting algorithm (LSQ-nonlin) [12] and PSO algorithm for designing IIR filters. When the relative performance of the new and existing methods is evaluated, it seems that the ABC algorithm can be an alternative approach to design low- and high-order digital IIR filters [13].

Zhu and Kwong proposed the Gbest-guided ABC (GABC) algorithm by defining a new position update strategy based on the global best (gbest) solution and new control parameter ψij, which is a uniform random number in [0, C], where C is a nonnegative constant. Thus, the gbest term can drive the new solution towards the global best solution. GABC was verified to perform better than ABC, in terms of efficiency and reliability [14]. A new version of GABC, namely modified Gbest-guided ABC (MGABC), was proposed with a few modifications to the original GABC. The MGABC algorithm iteratively tunes the step sizes of solutions during the global optima search process [9]. The performance of MGABC was measured by testing it on several complex optimization problems. The results were compared with those of ABC, best-so-far ABC, GABC, and modified GABC [15].

Karaboga and Gorkemli introduced a quick ABC algorithm (qABC) where Euclidean distance is exploited in order to choose the fittest food source within a restricted area. The qABC and standard ABC were examined for several well-known benchmark problems and the performance results were compared. The results showed that the convergence of the qABC is much better than that of the standard ABC [16, 17]. The ABC with memory (ABCM) algorithm exploits a memory mechanism in order to memorize artificial bees' previous successful

experiences of foraging behavior. This memory mechanism provides a guide for the further foraging of the artificial bees and it results in a more efficient search performance than the traditional ABC without memory ability [18]. The success of ABCM was analyzed on a set of benchmark problems and was compared with that of ABC, qABC, GA, PSO, and DE.

The qABC algorithm is one of the successful variants of the ABC algorithm. Aslan, Badem, and Karaboga introduced a new qABC algorithm, namely the improved quick ABC (iqABC), in order to balance between local and global search ability. In the iqABC algorithm, a new control parameter called bestLimit is included in the set of control parameters of the original ABC algorithm to adjust the workflow of the best limited employed and onlooker bee phases into the original employed and onlooker bee phases [19].

In the present study, the ABC algorithm is a swarm-based heuristic optimization algorithm that has been used in optimization studies because given its limited parameters it can be adapted to meet multiple goals [8]. This algorithm was written by drawing on the food-seeking behavior of bees. Several versions of the algorithm have been developed for different problem solutions [20]. The original artificial bee colony (ORJ-ABC), the modified artificial bee colony (MABC) algorithms, the GA, and the adaptive genetic algorithm (AGA) are used for parameter-setting problem-solving in the present study. In addition, the adaptive modified artificial bee colony (AMABC) algorithm has been developed in the present study. The results show that the AMABC algorithm can be successfully adapted to different problems in optimization operations. Further, the ORJ-ABC algorithm did not achieve success in optimization processes. The study results show that compared with the ORJ-ABC algorithm the MABC algorithm, which was developed to speed up the solution, is more successful in both complex system parameter optimization and numerical test functions optimization. It has been observed that the values of critical parameters of the MABC algorithm have a significant effect on problem solving. Therefore, the AMABC algorithm was developed by using the online adaptable critical parameter setting (OACPS) algorithm, which sets the critical parameter values in response to feedback received from the problem. According to the results reported herein, this algorithm can be adapted to the optimization problem and can give better results as well. GA and AGA, used in various optimization problems while proving successful, and the AMABC algorithm, produced in the present study, are compared.

When the studies reported in the literature are examined, it is seen that many algorithms are developed for the adjustment of MHAs' critical parameters, and the parameter setting process is performed by two different methods: online and offline. Studies using CALIBRA [21], SPO [22], REVAC [23], and design of experiments (DoE) [24] algorithms are offline methods that have found parameter values that take only numerical values. Moreover, studies using F-Race, iterated F-Race, and GA are offline methods that are aimed to set parameters that take values other than numerical values. In all these studies, good parameter values are found before the algorithm is executed and are assigned to the parameters of the algorithm as constant values and these values are not changed during the execution. Online parameter setting methods have been developed for reasons such as the inability to adapt to constantly changing situations during the operation of complex systems, the need to readjust the parameter values in each problem solution, and the high cost [4].

Different techniques have been used for online critical parameter setting [20]. One of these techniques being the deterministic parameter setting method updates the parameter values according to the determined rules and time periods. The second method is to set the parameter values of MHAs by using MHA. In this case, the use of a two-stage MHA increases the process complexity and causes performance loss. In the present study, it is important to find the right parameters in the agent-based modeling and simulation (ABMS) online

with the feedback received during the operation of the system.

In this direction, the online adaptive parameter setting (OAPS) method has been developed and the parameter values of the MABC algorithm used are adapted to the problem from the feedback coming from the OAPS algorithm, and the optimization process is performed. Thus, the developed AMABC algorithm is significant in terms of adapting to the different problems while optimizing and having a better performance overall [4].

The descriptions of the algorithms used, as set out in the method section, are useful for understanding the performance of the algorithms demonstrated in this section. The algorithms are used in experiments performed for this research and the performance of each compared in the prey predator model using numerical test functions. The general results are considered in the discussion section and future research directions are suggested in the conclusion.

2. Method

2.1. Artificial bee colony (ABC)

The food-seeking and -processing behavior of three types of bees—employed, onlooker, and scout—were adjusted to the ABC algorithm [8]. Each type of bee—employed, onlooker, and scout—in the ABC algorithm has a clearly defined function. The duty of the employed bees is to calculate the amount available of a given food source and specify that number to each of the employed bees. The number of onlooker bees is equal to the number of employed bees; they choose good food sources. Scout bees are responsible for finding food sources. If the existing food source gathered by an employed bee becomes scarce, this same bee becomes a scout bee by seeking food instead. The food source represents the possible solution. The amount of food is directly proportional to the quality of the solution set. The amount of the food source is defined as the fitness value in the ABC algorithm.

The working stages of the ABC algorithm are as follows:

1. Random food sources are established at regular intervals by using equation (1):

$$X_{ik} = minX_y + rand(0,1) \times (maxX_n - minX_n), \tag{1}$$

where i represents the index number of the food source selected, k represents the randomized index number, X_{ik} represents the solution set, $minX_n$ represents the minimum value of the nth element in the set, and $maxX_n$ represents the maximum value.

2. The amount of the food source, namely the fitness value of each of the food sources, is obtained via equation (2) or (3) of the fitness function:

$$if f_i \geq 0 \rightarrow f_i = 1/(1+f_i) \tag{2}$$

$$if f_i < 0 \rightarrow (1+ \mid f_i \mid), \tag{3}$$

where fi is the fitness value of the solution set at line i of the optimization problem used in the model.

3. Equation (4) is applied to each of the food sources established and new candidate food sources are created. The fitness values of the new food sources are calculated and then compared with the fitness values of the

old food sources. If the fitness value of any new food source is higher than that of any old food source, the new food source replaces the old one:

$$V_{ik} = X_{ik} + \varphi_{ik} \times (X_{ik} - X_{jk}) \tag{4}$$

V_{ik} represents the parameter value of new food source that emerges at the end of the formula. X_{ik} shows the parameter value at the k line of the food source selected and X_{jk} shows the parameter value at the k line of the randomized neighbor food source. j represents the index number of the randomized neighbor food source. φ_{ik} represents the distance of change between the parameters of the two food sources selected. This value is computed by the formula $(random(0,1) - 0.5) \times 2$ and varies between –1 and 1.

4. It may be possible to create better food sources by applying equation (5) to the food sources selected. Existing possibilities for selection (roulette wheel) are used in this process. The fitness values are computed again and compared with the old ones. If the fitness value improves, the new food source replaces the old one.

$$O_i = f_i / \sum_j f_j \tag{5}$$

Oi represents the possibility that a given solution set will be selected for the onlooker bee, f_i represents the fitness value of the solution set selected, and $\sum_j f_j$ represents the total of the fitness values of all the solution sets.

5. If the existing food source cannot be enhanced as much as the number of the cycle count determined, that source is deleted and a new random solution set is established using equation (1) [25].

2.2. Modified artificial bee colony (MABC)

The MABC algorithm, which is used in the present study, was developed in order to solve problems in complex systems more successfully and more quickly than is possible with ABC. The MABC algorithm has two parameters that differ from those of the original ABC algorithm, one of which pertains to the ratio and the other to the scaling factor (SF) parameter [26]. In regard to the ratio, i.e. the modification rate (MR), whereas the original ABC algorithm updates a single food source while producing that food source, the MABC algorithm determines the number of food sources, which could be multiple, that need to be changed. The modification rate is determined by comparing the randomly produced Rij value and the MR value in equation (6) or (7).

$$if Rik < MR then V_{ik} = X_{ik} + \varphi_{ik} \times (X_{ik} - X_{jk}) \tag{6}$$

$$Else V_{ik} = X_{ik} \tag{7}$$

The change in terms of the SF parameter pertains to the value range produced. Whereas the number randomly produced in the solution search equation in the ABC algorithm is in the range of [–1,1], in the MABC algorithm the value is in the range of [–SF, SF]. In the case of SF parameters with high values, the possibility of finding the optimum value decreases while the algorithm comes to the solution faster. Thus, this range [–1,1] is used in the present study.

2.3. Adaptive modified artificial bee colony (AMABC)

The MABC algorithm and different MHA can solve the problem of complex system parameter setting modeled in agent-based models and simulations. The study results show that the MABC algorithm's critical parameter variable values determine the solution. This is determined to be the case because in the present study the best solutions are generated by different critical parameter variable values for different problems. Manually setting the critical parameter values for each problem would be tedious and time consuming. Therefore, the online adaptable critical parameter setting (OACPS) algorithm was used to obtain critical parameters specific to the problem. In the present study, the OACPS algorithm is used given that it is suitable for each kind of data set focused on finding the appropriate parameter value online with the feedback received from the algorithm. Critical parameter values are determined in relation to the feedback from the algorithm, and the value range of the parameters can be adjusted if necessary. The OACPS algorithm can extend and narrow the search space. The algorithm adapts to the problem, thereby speeding up the process whereby a solution is reached. The MABC, the GA, and other metaheuristic algorithms need to be adaptable to the problem of showing the same success in different problems as well. In the present research, the AMABC algorithm and the AGA were developed to find a solution for parameter-setting problems in agent-based models and simulations by using the OACPS algorithm (The OAPS algorithm has been developed to adjust the critical parameter values of MHAs). Figure 1 shows the flow diagram of the algorithm developed.

It is seen when this flow diagram is examined that the critical parameter value is adapted to the problem by using the OAPS method. Further information about the method used in the next chapter will be given.

2.3.1. Online adaptable critical parameter setting (OACPS) algorithm

The literature shows that MHAs with fixed parameter values are not sufficient to address problems in complex systems because of the dynamic structure of the latter [8, 25–27]. However, through the OACPS algorithm, MHAs can use the proper parameter values established based on feedback received from the problem and are, therefore, more successful in optimization operations than classical methods. When the proper critical parameter values are found during the working of the algorithm, the performance of the latter improves.

For operation of the OACPS algorithm, first a trial array (tArray) is created for the critical parameter values, which are selected at random. The OACPS algorithm is executed with the values in the trial array. The worst parameter sets found by statistical tests are deleted, a step that prevents bad solution clusters from being run on the model. The remaining solution clusters are sent to the algorithm run online, and the fitness values obtained in this way are compared. If the global fitness value is improved, the critical parameter values used in that cycle are taken into the success array (sArray). This process continues until the elements in the array end off. Next, the values in the tArray are initialized again, and those elements in the number determined by the success percentage (sPercent) from the sArray are randomly assigned to the tArray. The rest of the values are filled with values that are randomly produced. The sPercent value increases during the run-time, which means that more sArray values can be used as the algorithm progresses. However, in order to avoid narrowing the search space, not all the values in the sArray are transferred to the tArray. If there are no elements in the sArray, the value ranges are extended and the same procedures repeated until there are elements in the sArray. The chief goal is to improve the critical parameter values in the MABC algorithm online with the needs of the problem. A flowchart of the OACPS algorithm is shown in Figure 2.

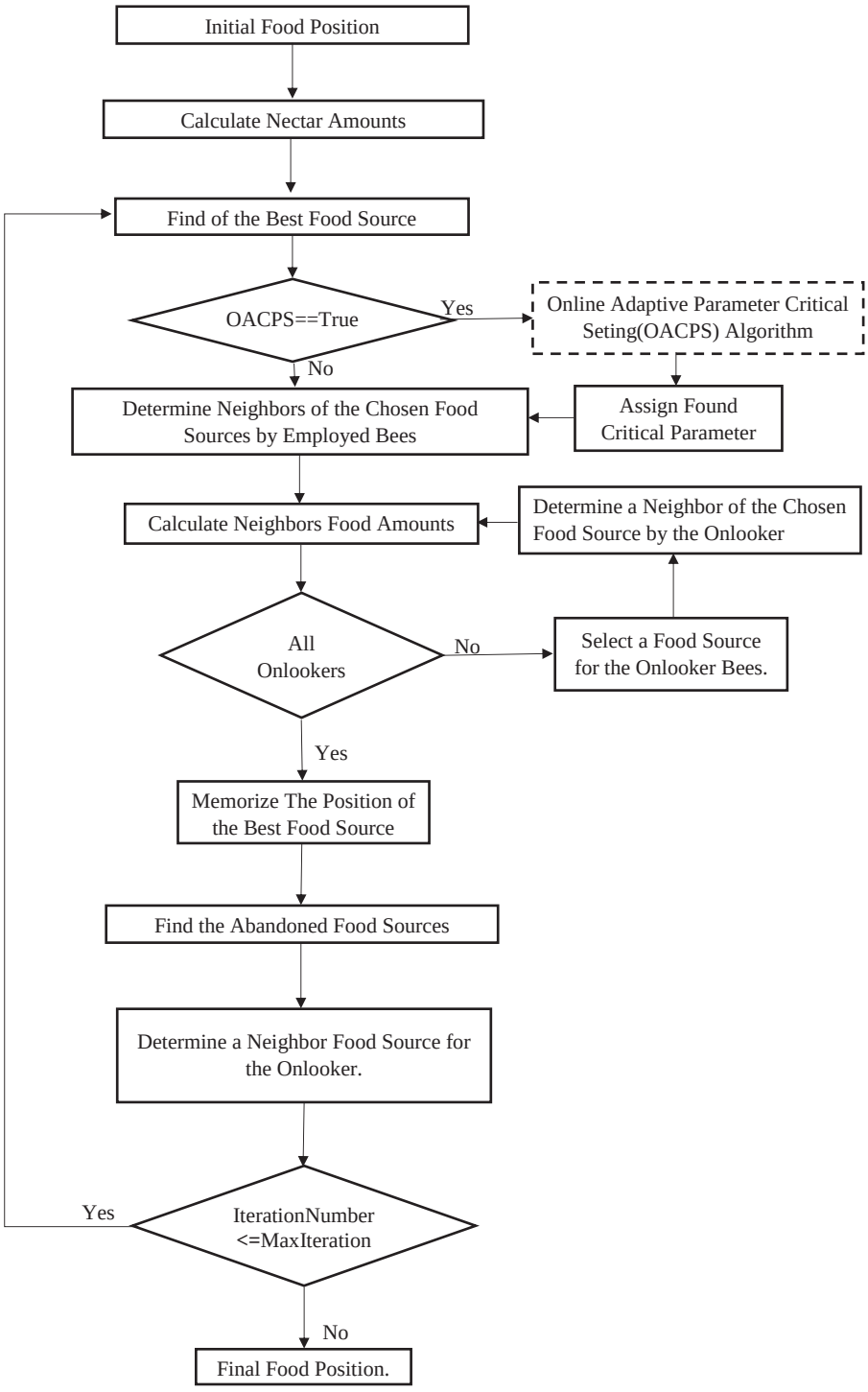

Figure 1. Flowchart of the adaptive modified artificial bee colony algorithm.

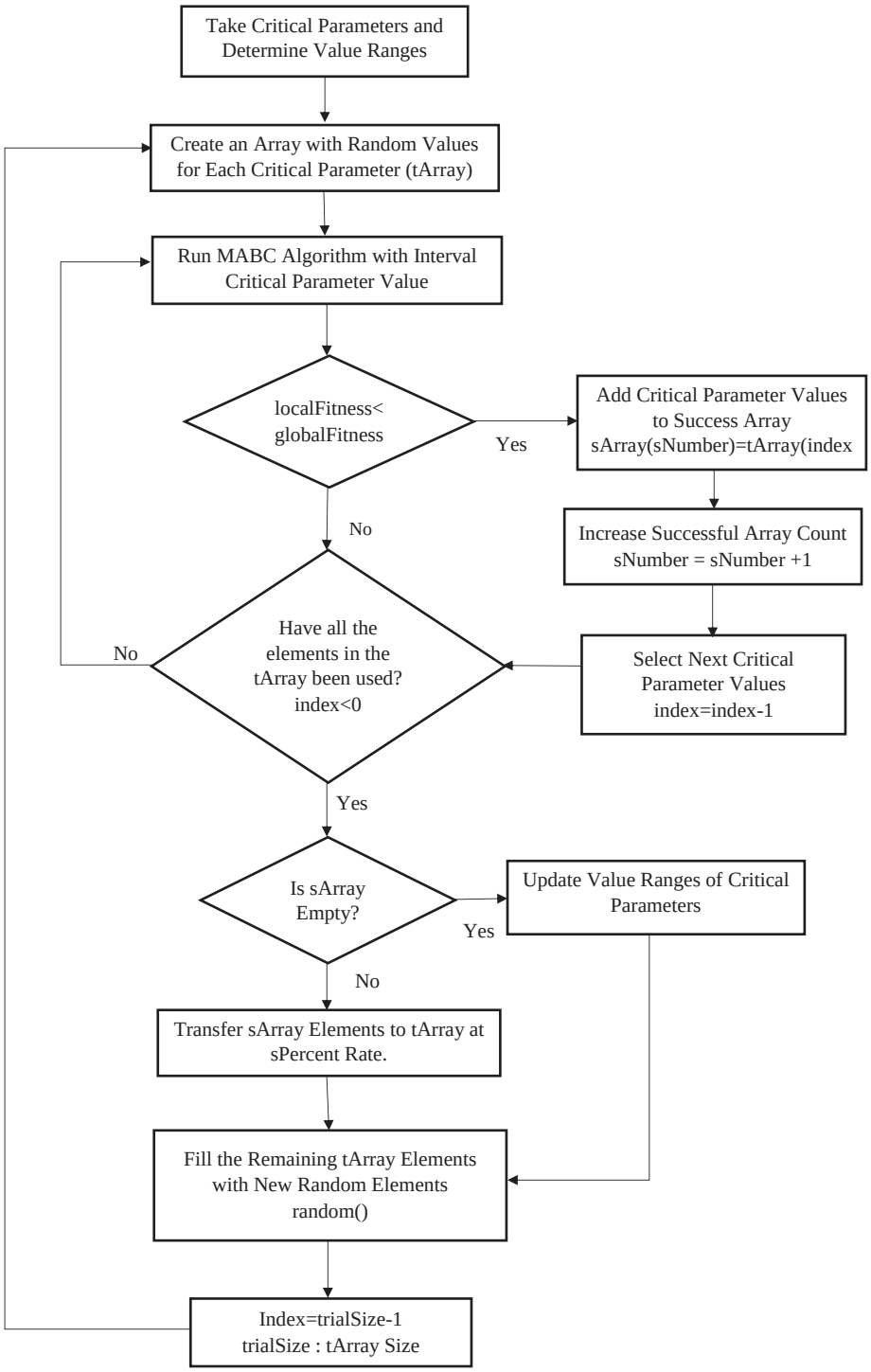

Figure 2. Flowchart of the online adaptive critical parameter setting algorithm.

2.4. Genetic algorithm (GA) and adaptive genetic algorithm (AGA)

The GA aims heuristically to find the best solution or approximate optimum solution using the genetic code structure of living beings. It seeks for the global solution in the complex, multidimensional search space,

according to the principle of survival of the best. While GA parameters refer to the genes in biology, the collective set of the parameters forms the chromosomes as well. Each individual of GAs, that is, each possible solution, is represented as a chromosome. This candidate set of solutions is also called a population. The fitness of the population is maximized or minimized within certain rules. Each new generation is acquired by combining survivors within the sequences created by random information exchange. There are two basic genetic processors, known as crossover and mutation, in GAs. Two individuals are selected from the population for crossover process. The points to be crossed are determined in these individuals, and the elements of the individuals are mutually displaced from that point. Thus, two new individuals are obtained. The genes of the individual are changed by the mutation processor. This change generally covers 1% to 5% of the population. The mutation causes variability in the population and prevents the problem result from being intervened by local solutions. The parameters used in the GA are given in Table 1.

Table 1. Parameter values used in the genetic algorithm.

Genetic algorithm parameters	Parameter value
Selection method	Tournament
CrossovermMethod	One-point crossover
Parent percentage	70
Percent mutation	0.5

The critical parameter values of the GA given in Table 1 are the parameters that should be determined in the GA. These parameter variables can take different values in each problem. The critical parameter values of the GA algorithm should be adjusted manually for each problem that it is used to solve. This process is time consuming and it is impossible to produce the most appropriate values manually. The AGA has been developed to solve this problem. Thus the critical parameter values of the GA can be adapted to the problem [28].

3. Experimental studies

3.1. Model used (prey predator)

The purpose of the prey predator model is to create and maintain the conditions needed to ensure the continuity of the prey predator ecosystem. The wolf–sheep ecosystem is examined in this simulation model to test the approach developed. The parameters that affect the continuity of this ecosystem are determined and adjusted in line with the goals of the ecosystem. There are wolf, sheep, and grass agents in the prey predator model. The wolf agent and sheep agent move randomly in a simulation environment. These agents need energy to move in the environment; they are created with a specific amount of energy. They disappear when their energy runs out. To replenish its energy, the wolf preys on the sheep. Likewise, to replenish its energy, the sheep eats the grass. Figure 3 shows a prey predator model class diagram.

There are four classes in the prey predator model in agent-based modeling and simulation: wolf, sheep, grass, and SimpleAgent, the last of which has the characteristics common to all of the other three classes. “Die” refers to death behavior, “move” to moving behavior, and “heading” refers to the coordinate change by determining the direction of the agent. The move function, which is abstract, is different for each of the agents. The wolf class represents the wolf agent. The step() function is the function in which the reproduction and preying behaviors of the wolf and sheep take place. The life cycle of the grass is represented by constantly reducing the lifetime randomly determined for each agent in the grass class. The consume function represents

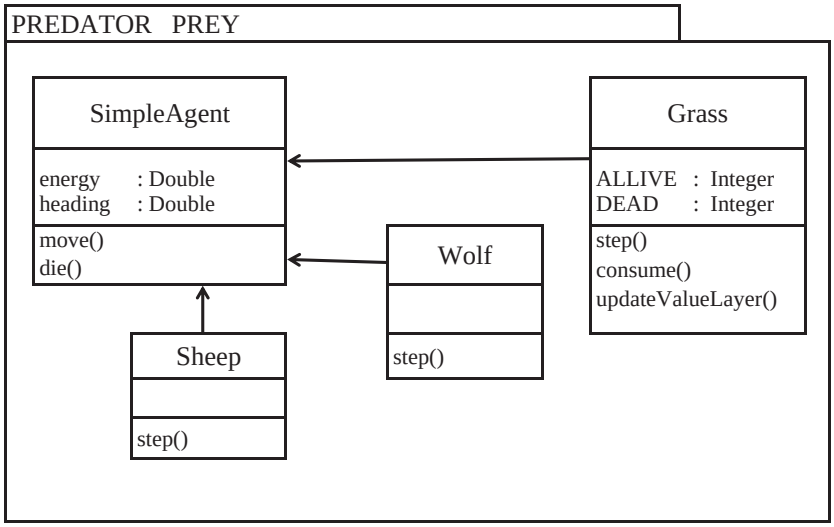

Figure 3. Prey predator model class diagram.

the process whereby the grass is eaten by the sheep, and the process of regrowth takes place in the step function. The updateValueLayer provides an updated account of the grass in order to determine its current status.

The purpose of prey predator model is to ensure the continuity of the ecosystem. Another purpose is to create a model that represents real life in which wolf, sheep, and grass agents do not become depleted. It is necessary, therefore, to set the parameter values of the model. Table 2 shows the parameters of the prey predator model to be adjusted in order to ensure the continuity of the population with the best parameter set.

Table 2. Parameters of the prey predator model that is modeled in the ABMS environment.

Model parameters	Parameter explanation	Value ranges
initialnumberofwolves	Initial number of wolves agent in the population	40-250
initialnumberofsheep	Initial number of sheep agent in the population	40-450
initialnumberofgrass	Initial number of grass agent	Whole area
wolfgainfromfood	Energy gained by wolf agent from preying	10-40
Sheepgainfromfood	Energy gained by sheep agent from grazing	1-12
wolfreproductionrate	Reproduction rate of wolf agent	3-15
sheepreproductionrate	Reproduction rate of sheep agent	3-15
Grassregrowtime	Regrow time of grass agent	10-60

The initial solution set is randomly established based on the value ranges that these values can receive. The solution set comprises the following: [initialnumberofwolves, initialnumberofsheep, wolfgainfromfood, sheepgainfromfood, wolfreproductionrate, sheepreproductionrate, grassregrowtime]. An initial population composed of 10 subjects (10 solution sets) is constituted. Then the best solution set is sought via a search in the search space via the algorithms used.

3.2. Parameter setting in agent-based modeling and simulation

The complex systems modeled in agent-based modeling and simulation were tested in a prey predator model that is an agent-based model and simulation. This test was applied to observe the performance of the ORJ-ABC, MABC, and AMABC algorithms used to optimize the data set. These algorithms were used to adjust the prey predator parameter set, which is an ecological system problem. Details regarding the PC, the tool used to develop the approach, and the programming are set out in Table 3.

Table 3. PC details used for the experiments.

System	Windows 10 professional 64 bit
RAM	16 GB
CPU	Intel(R) Core(TM) i7 2.10 GHz
Algorithm	ABC [8], MABC [26], AMABC
Prog. language	Repast Simphony 2.5-Eclipse-Java [29]

The results of the parameter setting operation for the prey predator model obtained by 20 iterations are shown in Table 4. The number of steps of each of the iterations required for the simulation is 2000 ticks. Each of the algorithms was performed on the model 10 times, and the average of the results is shown in Table 4. Explanations of the column headings in this table are as follows:

Optimization algorithms: The ORJ-ABC, MABC, and AMABC algorithms used to optimize the problem dataset. **Initial best fitness:** The average best fitness value of the solution sets randomly established in the first iterations in the simulation. **Best fitness:** The average of the best solution sets from the simulations. **Improvement difference:** The difference between the initial best value and the best fitness value. **Average local:** The average of the local values from each of the iterations in the simulations. **Average improvement number:** The average improvement in the fitness value of the prey predator model over 20 iterations in each run.

Table 4. Algorithms and optimization results of the prey predator model in agent-based modeling and simulation.

Optimization algorithms	Tick count	Initial best fitness value	Best fitness value	Improvement difference	Average improvement number
GA	12.675,000	0,900000000	0,229411760	0,670588240	2,0
AGA	156.468,222	0,32946697	0,07659524	0,25287173	4,8
ORJ-ABC	76.380.0	0,264330395	0,406486594	-0,109092491	4,2
MABC MR=10	107.464000	0,584705882	0,264978932	0,31972695	3,4
MABC MR=50	171.250,250	0,689411765	0,166560033	0,522851732	4,4
MABC MR=100	178.258,8	0,515516340	0,100767780	0,414748560	4,8
AMABC	211.718,625	0,801176471	0,085745833	0,715430637	5,8

Table 4 shows the fitness value of each algorithm, all of which are between 0 and 1, obtained at the end of the prey predator model. The present study shows that a best solution set can be found with a minimum fitness value of zero. Thus, the fitness values obtained from the algorithms tested on the model indicate which algorithm provides the best solution. It is known that in terms of the initial best fitness value the ORJ-ABC algorithm has a smaller fitness value than those used by the other algorithms and the random solution produced first

positively affects the solutions that follow. However, the best fitness value found by the ORJ-ABC algorithm at the end of the optimization process produced a value bigger than the initial value. Therefore, with this algorithm, it was not possible to perform an optimization operation in the prey predator model.

The GA, used in various optimization problems while proving successful, and the AMABC algorithm, produced in the present study, are compared. According to the results shown in Table 3, the GA is more successful than the MABC-MR=10 and ORJ-ABC algorithms. When the MR parameter of the MABC algorithm is inputted as 50 and 100 consecutively, it is observed that the parameter setting success proportionally increases in the prey predator model. When the compatibility values are examined, it is seen that the MABC-MR=50 and MABC-MR=100 algorithms are more successful than the GA. However, when the improvement difference of the algorithms is examined, it is seen that that GA produces better results than both of the MABC-MR=50 and MABC-MR=100. When the AMABC algorithm is compared with the GA, no major difference was found for improvement differences or initial best fit values. However, when the best fit values and the number of enhancements are examined, it is seen that AMABC yields approximately 2.5 times better results than the GA. The GA seems to be much faster than other algorithms. There are two reasons why the GA is faster:

1. Since each iteration is made up of as many new individuals as the percentage of parental selection, it creates fewer individuals in each iteration, unlike other algorithms, depending on the percentage of selection. Thus, the suitability value of fewer individuals than other algorithms is calculated.

2. Because the children created in the GA have inherited the genes of their parents, meaning the solution sets are crossed, similar sets of solutions are formed more than in other algorithms. Hence, the fitness values of the same solution cluster are recorded. Clusters of solutions with known fitness values are not simulated again, which shortens the time.

In the present study, it is seen that the AGA developed to solve complex system optimization problems [29] is more successful than the GA in solving optimization problems. The AGA completes the optimization process in a longer time than the GA does since the AGA has better fitness values and therefore the model runs longer compared to the model using the GA. The best fitness value and average improvement number produced are other results that show successful optimization. When we compare the AGA with the MABC algorithm, the best fitness value of the AGA is better than that of the MABC algorithm and this shows that the AGA is more successful than the MABC algorithm. However, when the AGA is compared with the AMABC algorithm, it is seen that the best fitness values are very close. When the initial best fitness values are compared, the AGA starts with a lower fitness value compared to the AMABC algorithm and therefore it will be easier to achieve a better fitness value. Furthermore, when the improvement difference and average improvement numbers are analyzed, it is seen that the AMABC algorithm is successful in solving this problem.

When the MABC algorithm is used, several best fitness values are obtained, with each depending on the respective value of the MR parameter. Thus, there is a need for the MR value to be determined in relation to the specific structure of each given problem. The AMABC algorithm was developed so that it would no longer be necessary to manually adjust the MR parameter (value between 0 and 100) to different problems. Instead, via the online parameter setting method, the AMABC algorithm optimizes the problems by finding the proper parameter value. Even though the initial best fitness value of the AMABC algorithm is worse than those found by the other algorithms, the best fitness value found (0.085745833) by the AMABC is better than the best values found by the other algorithms (Table 4). Again, the AMABC algorithm finds the best algorithm, i.e. the best improvement difference number for the prey predator problem. Moreover, the improvement number of

the AMABC algorithm is better than those of the other algorithms for the same iteration (Table 4). There is a direct proportion between finding the best fitness value and the improvement numbers. Further, although the improvement number of the ORJ-ABC algorithm was found to be high, the fitness value indicates that a close to optimum solution could not be found.

The reason why the number of ticks is MABC-MR=10 « MABC-MR=50 « MABC-MR=100 « AMABC, that the simulation is terminated in order to prevent the extinction of the species in the algorithms in order to shorten the solution generation time in the hunt-hunter model used, until the determined tick number (2000). Clustered solution sets are considered to be bad. Because of this, ORJ-ABC takes much less time than MABC MR=10, MABC MR=50, MABC MR=100, and AMABC because it produces worse solutions. Likewise, the number of undesirable solutions of the MABC MR=100 algorithm being higher than those of AMABC caused it to take less time. The average improvement number and the best fitness values also indicate this order.

The results achieved with the GA, ORJ-ABC, MABC-10, MABC-50, MABC-100, and AMABC algorithms used in the agent-based model and simulations are shown in Figures 4–8, respectively. The number of first created solution sets of each of the algorithms is 10. The first solution sets were randomly established around the value ranges determined. The length of time specified to run each solution set for the simulation was 20,000 ticks, and each solution set was run for 20 iterations. It should be noted here that there is the option of changing these values on the interface.

All the algorithms used were run in the prey predator model as an agent-based model and simulation.

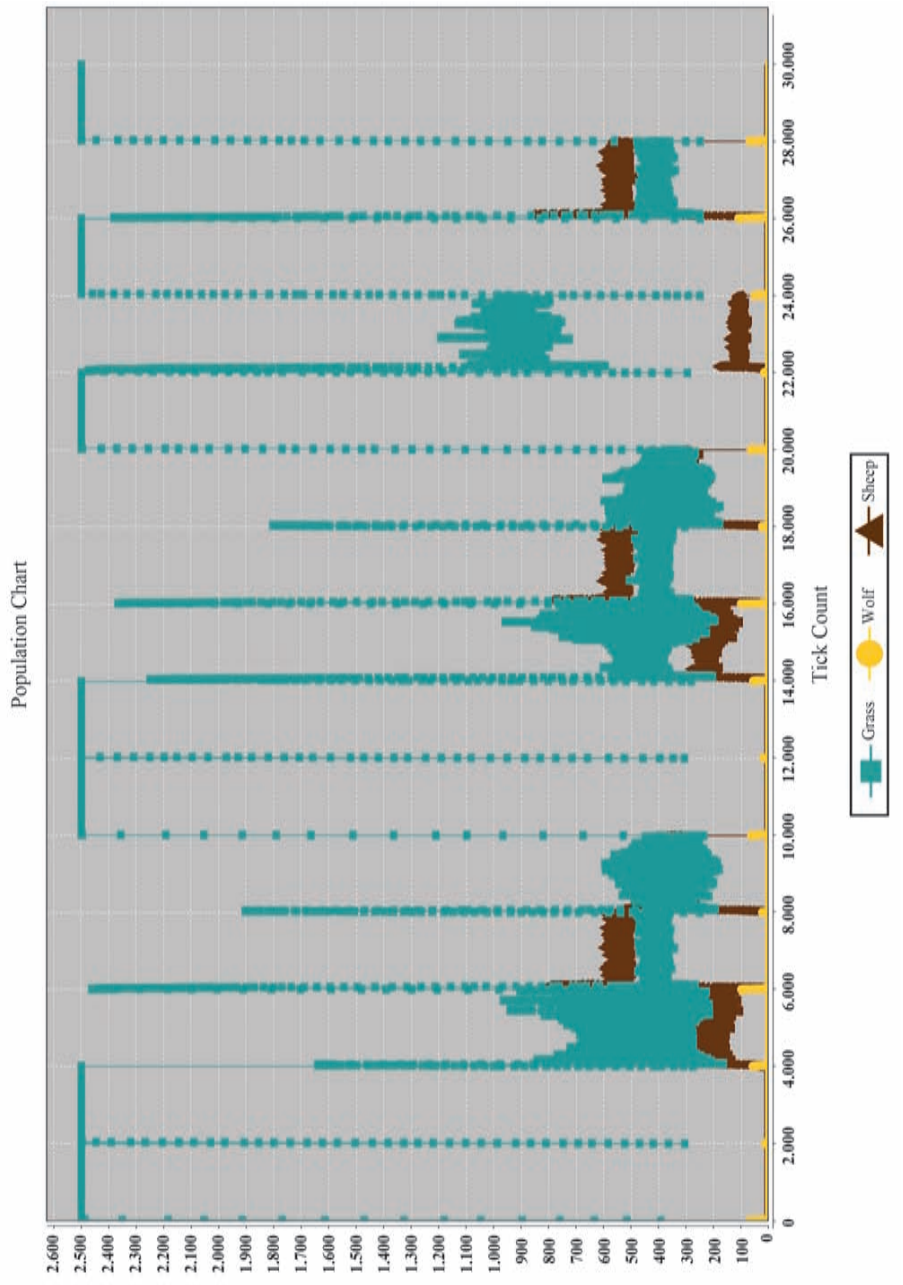

Figure 4. Parameter setting via the ORJ-ABC algorithm in the prey predator model.

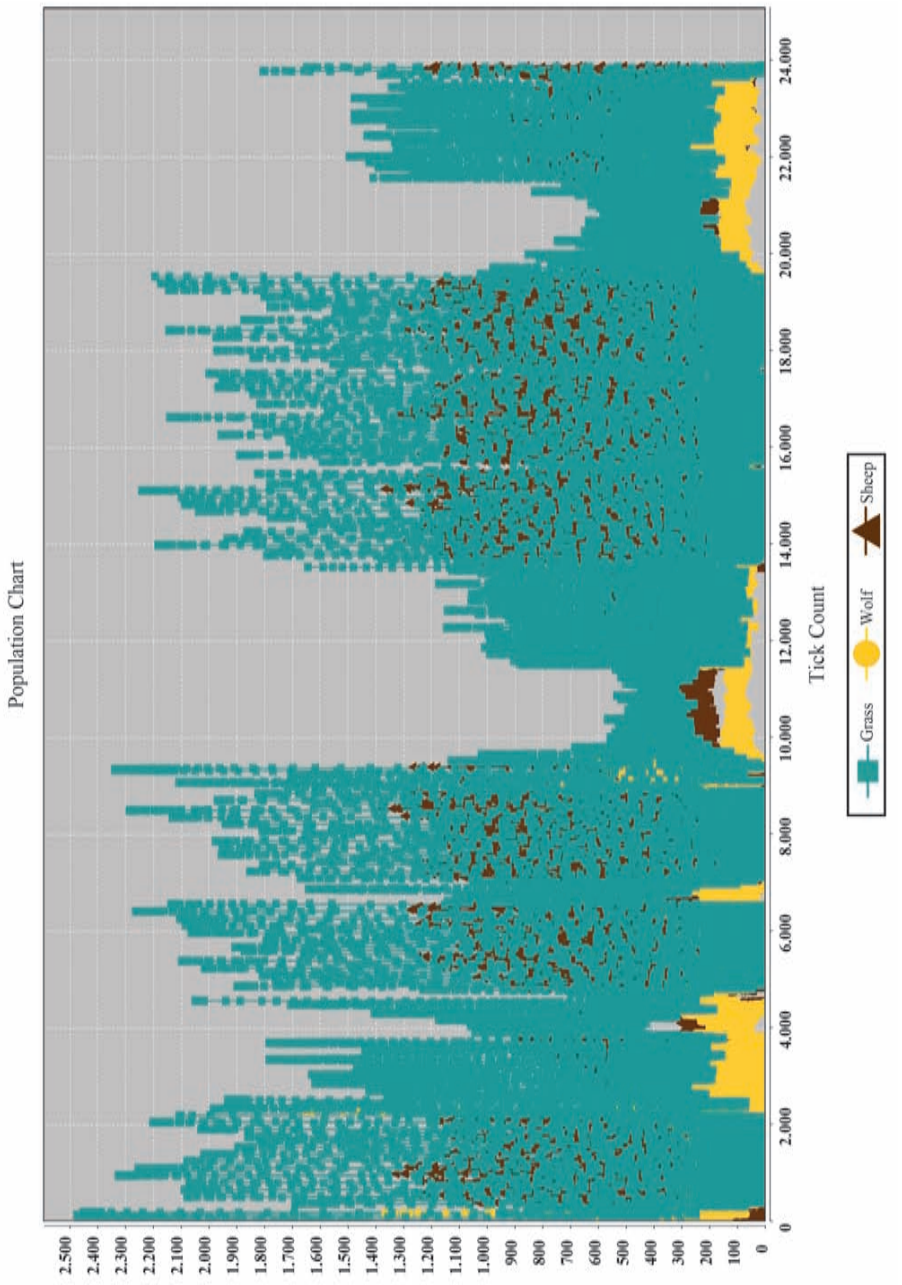

Figure 5. Parameter setting via the MABC-10 algorithm in the prey predator model.

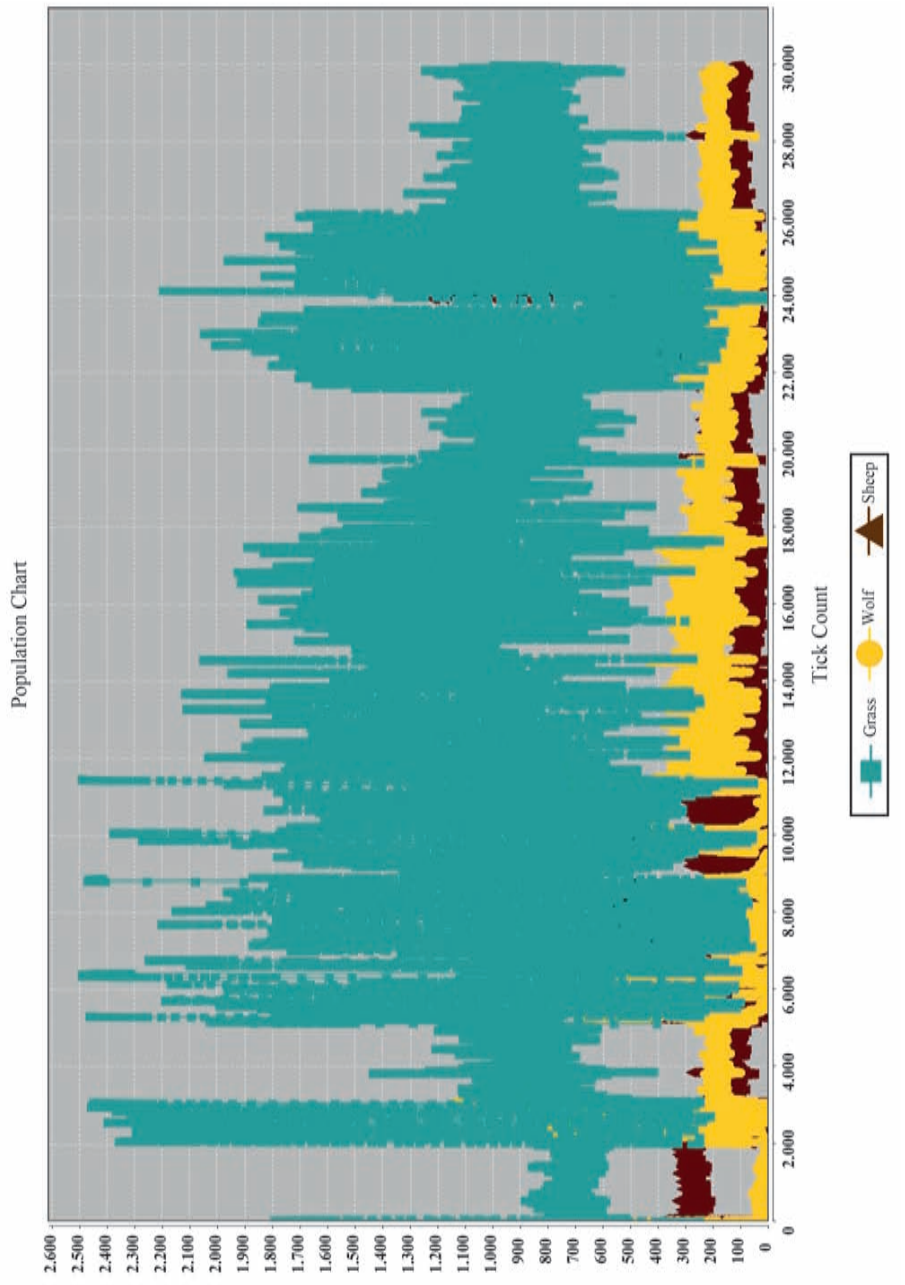

Figure 6. Parameter setting via the GA in the prey predator model.

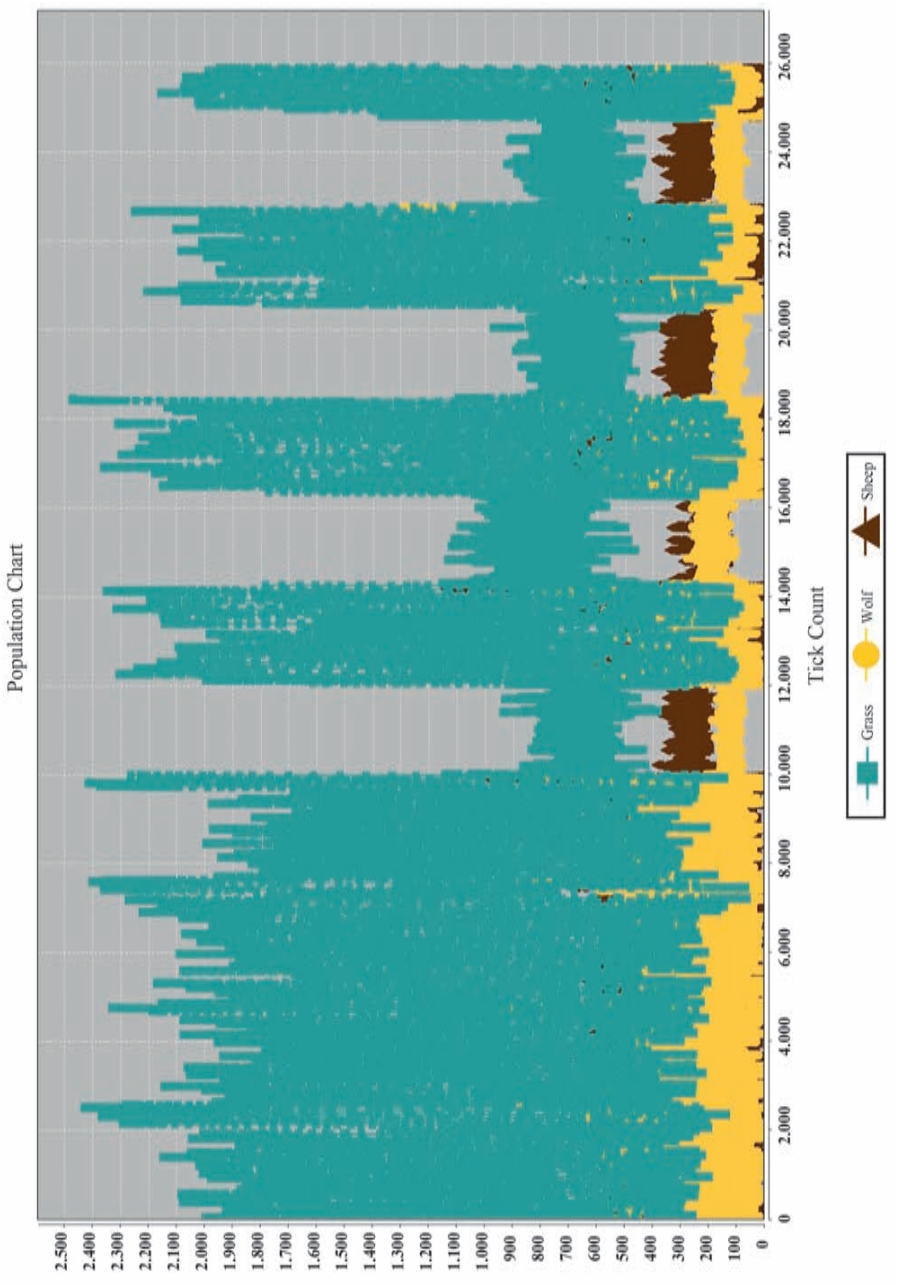

Figure 7. Parameter setting via the MABC-50 algorithm in the prey predator model.

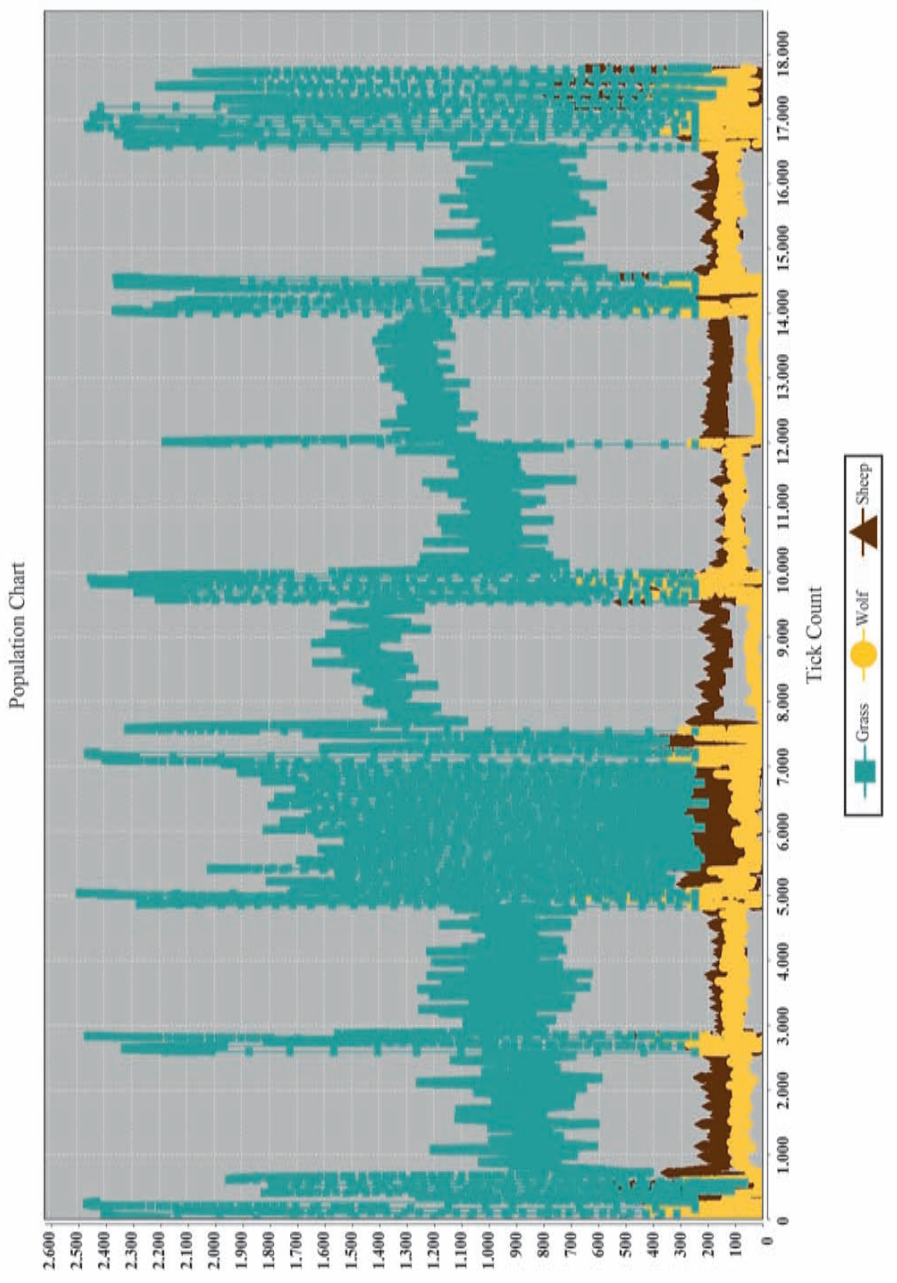

Figure 8. Parameter setting via the MABC-100 algorithm in the prey predator model.

Population changes in the number of wolf, sheep, and grass agents in the time frame specified to adjust the parameters in the prey predator model are shown in Figure 4. The goal in terms of setting the parameters is to ensure the continuity of the wolf, sheep, and grass species. The parameter values of the model need to be adjusted. The ORJ-ABC algorithm that was rerun at 20,000 ticks and also run for a total of 20 iterations failed to find the parameter values to ensure the continuity of the species. The principal goal is to prevent the extinction of each of the species. For each of the iterations, a further goal is to become extinct especially after creating the wolf population. In conclusion, the purpose of the parameter-setting operation in an agent-based model and simulation cannot be achieved by the ORJ-ABC algorithm.

Figure 5 shows the process of the prey predator model parameters-setting of MABC. Unlike the ORJ-ABC algorithm, the MABC algorithm has an MR parameter that directly affects the performance of the solution of the problem. The graphic shown in Figure 4. was obtained at the end of the parameter-setting operation by manually assigning a value of 10 to the MR parameter. As can be seen in Figure 4, parameters that ensure the continuity of the species in the prey predator model were found. These parameters can be clearly seen between 10,000 and 14,000 ticks. Yet, the value of 10 assigned to the MR parameter is not suitable for the parameter-setting problem in the prey predator model. Instead, this algorithm must be run through a much greater number of iterations in order to find a closer to optimum solution set for ensuring the continuity of the species such that more time is needed for this algorithm to be effective.

Figure 6 shows the process of the prey predator model parameters-setting of the GA. As can be seen from Figure 5, the wolf, sheep, and grass populations do not run out at certain intervals. This demonstrates the success of the GA in parameter setting. After the first few iterations, it is seen in the graph that the parameters that provide the continuity of the species for a long time are produced.

Figure 7 shows the graphic obtained by manually assigning a value of 50 to the MR parameter. This graphic evolves toward better parameters after a specific iteration in comparison with the graphic in Figure 4. After 10,000 ticks, parameter values that ensure the continuity of the species can be found. An MR parameter value of 50 is shown to be considerably more successful than an MR parameter value of 10 in the parameter-setting operation.

Figure 8 shows the parameter-setting graphic obtained by assigning a value of 100 to the MR parameter value of the MABC algorithm. It is evident that the best parameter values can be found via the first iterations. The species do not deplete at certain intervals. In this case, the MR parameter value of 100 is ideal for the solution of this problem. However, running the simulation by manually changing the parameter value setting continuously is both difficult and time consuming. As the number of parameters that need to be adjusted increases, it is almost impossible to find the best fit parameter by manually setting these critical parameter values. Therefore, there is a need for a mechanism that can adjust the existing critical parameter value.

When Figure 9 is examined, it is evident that the AGA finds suitable parameter values after the first 3 iterations (i.e. after 6000 ticks) since a certain number of species do not deplete at certain intervals. This shows the success of the AGA in finding suitable parameter values.

The graphic in Figure 10 shows the result of the parameter-setting procedure of AMABC algorithm. The uncertainty continues in almost 4000 ticks until the proper MR value is found in the first iterations. The proper parameter values are found (solution sets) that ensure the continuity of the species over the working time via the proper MR parameter value in the following iterations. Without the need to manually set the MR parameter value of MABC algorithm, the model parameters were adjusted by the AMABC algorithm by receiving a value fit for the problem. Thus, the best fit parameter set that can reach the model targets was obtained.

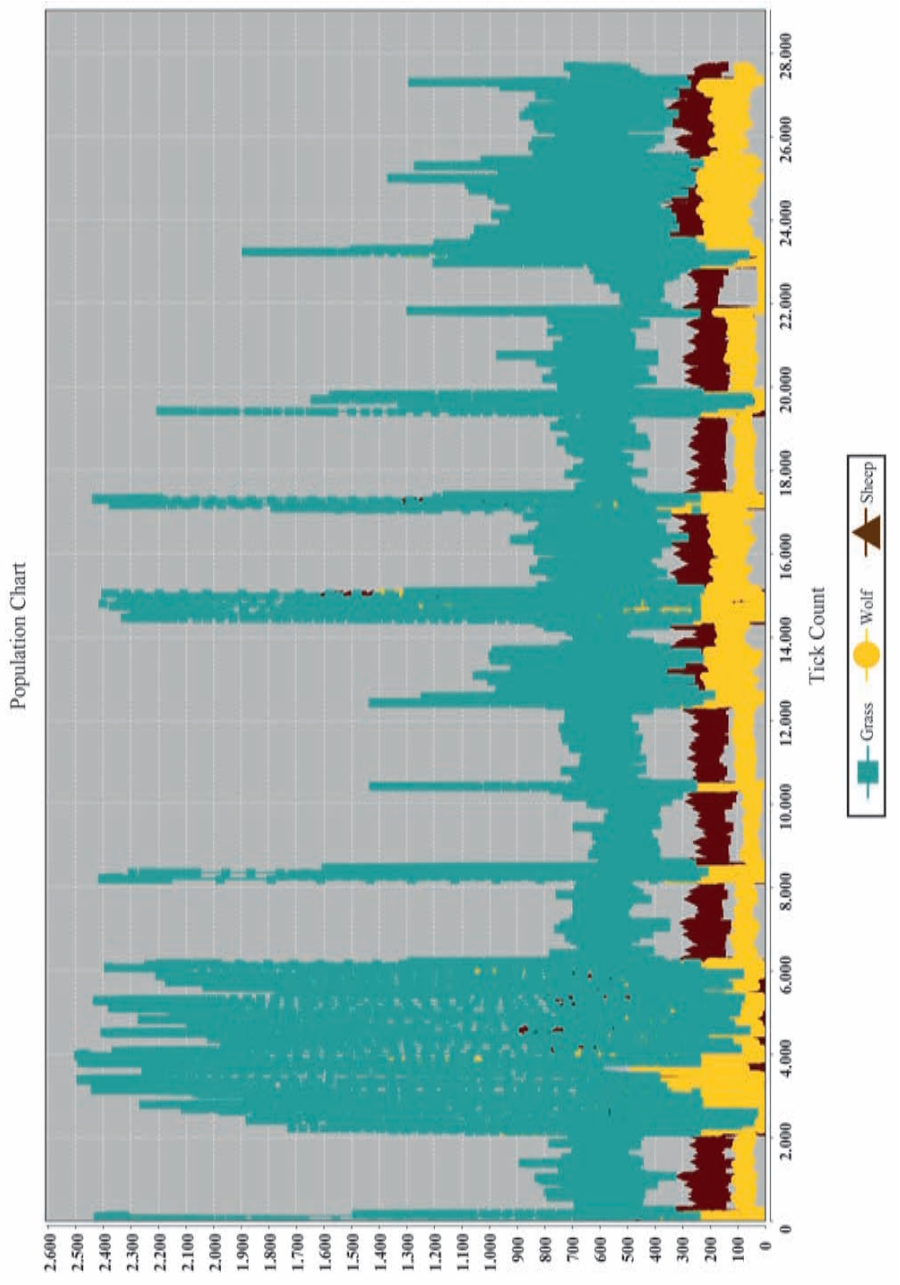

Figure 9. Parameter setting via the AGA in the prey predator model.

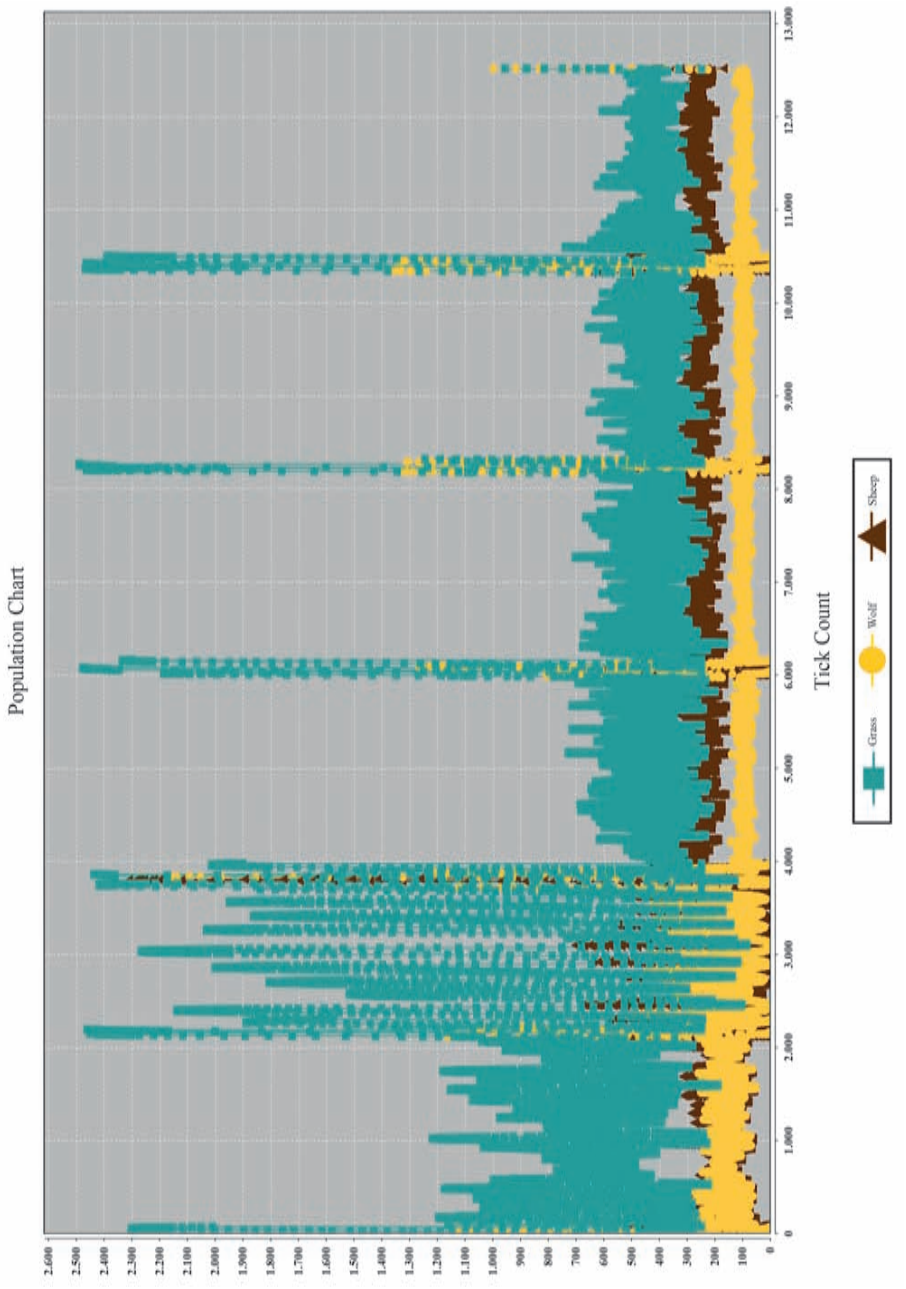

Figure 10. Parameter setting via the MABC-OACPS algorithm in the prey predator model.

Figure 11 shows the graphic of the best prey predator model found by the AMABC algorithm in the simulation. Figure 12 shows the graphic of the best prey predator model found by the AGA in the simulation. These graphics show the distribution of the wolf, sheep, and grass populations when the simulation is run with parameters manually by 20,000 ticks. The purpose of parameter setting is to find the best fit parameter values to ensure the continuity of the species. This research achieved the parameter setting operation targeted.

3.3. Numerical test functions used for the optimization operation

The numerical test function formulas are used to evaluate the performance of the optimization algorithm. Some test functions, i.e. Rastrigin and Griewank, are multimodal, meaning that each has a large number of local optimum points in addition to the global optimum. However, the test functions used herein are the Sphere and Rosenbrock functions, which are unimodal with no local optimum values with the exception of the global optimum. These test functions, which differ from each other in terms of their characteristics, were used to determine the relative success of the ORJ-ABC, MABC, and AMABC algorithms in optimization. The algorithm that performed best in the model parameter setting performed best in the optimization procedure of the test functions as well. Table 4 shows the name, formula, and minimum point (i.e. the best solution value), and the search range.

The objective of the optimization operations of the numerical test functions is to come as close as possible to the min small point, as shown in Table 5. In particular, given the success of the AMABC algorithm in terms of setting model parameters, determining whether it can be successful on the optimization functions is of great interest. The optimization results of each numerical test function are shown in Table 5, and then with the GA, AGA, ORJ-ABC, MABC, and AMABC algorithms in Table 6. The data obtained by running the GA, AGA, ORJ-ABC, MABC, and AMABC algorithms five times on each of the functions are shown in Table 6. An evaluation of the data presented in Table 6 indicates that the AMABC and AGA perform well for model parameter setting and in the optimization operations of the numerical test functions. Different parameter values of the MABC algorithm also performed well for each test function, which demonstrates the importance of the algorithm's critical parameter value in optimization operations. However, in comparison with the MABC algorithm, the AMABC algorithm is more successful in the optimization of each of the four test functions because in the latter algorithm the critical parameter values are adapted to the problem. When the GA is examined, it is seen that Griewank test function is very successful in the optimization process. However, it was found that the AMABC algorithm was better than the GA in optimization of other test functions. When the success of the AMABC algorithm and the AGA in test functions is compared, it is seen that the AMABC algorithm is more successful in optimization of Rastrigin and Rosenbrock functions, whereas the AGA is more successful in optimization of Sphere and Griwank test functions. This shows that the adaptation of the MHAs to the problem has directly affected the performance of the algorithm. All MHAs developed can be adapted to problems by using the OAPS algorithm developed in the present study. Thus the performance of the algorithms will increase as shown herein.

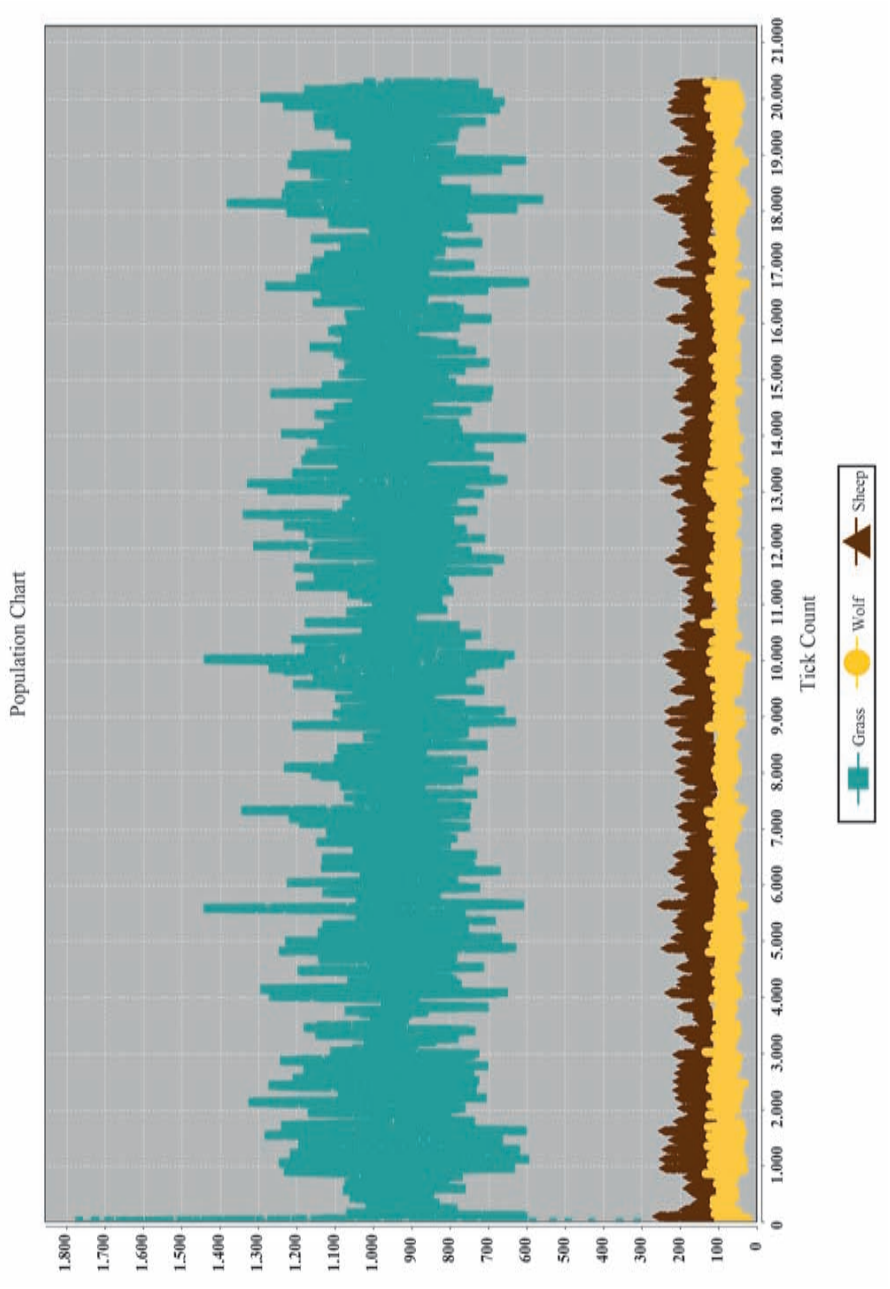

Figure 11. Manual run of the best parameter values obtained with the AMABC algorithm for the prey predator model.

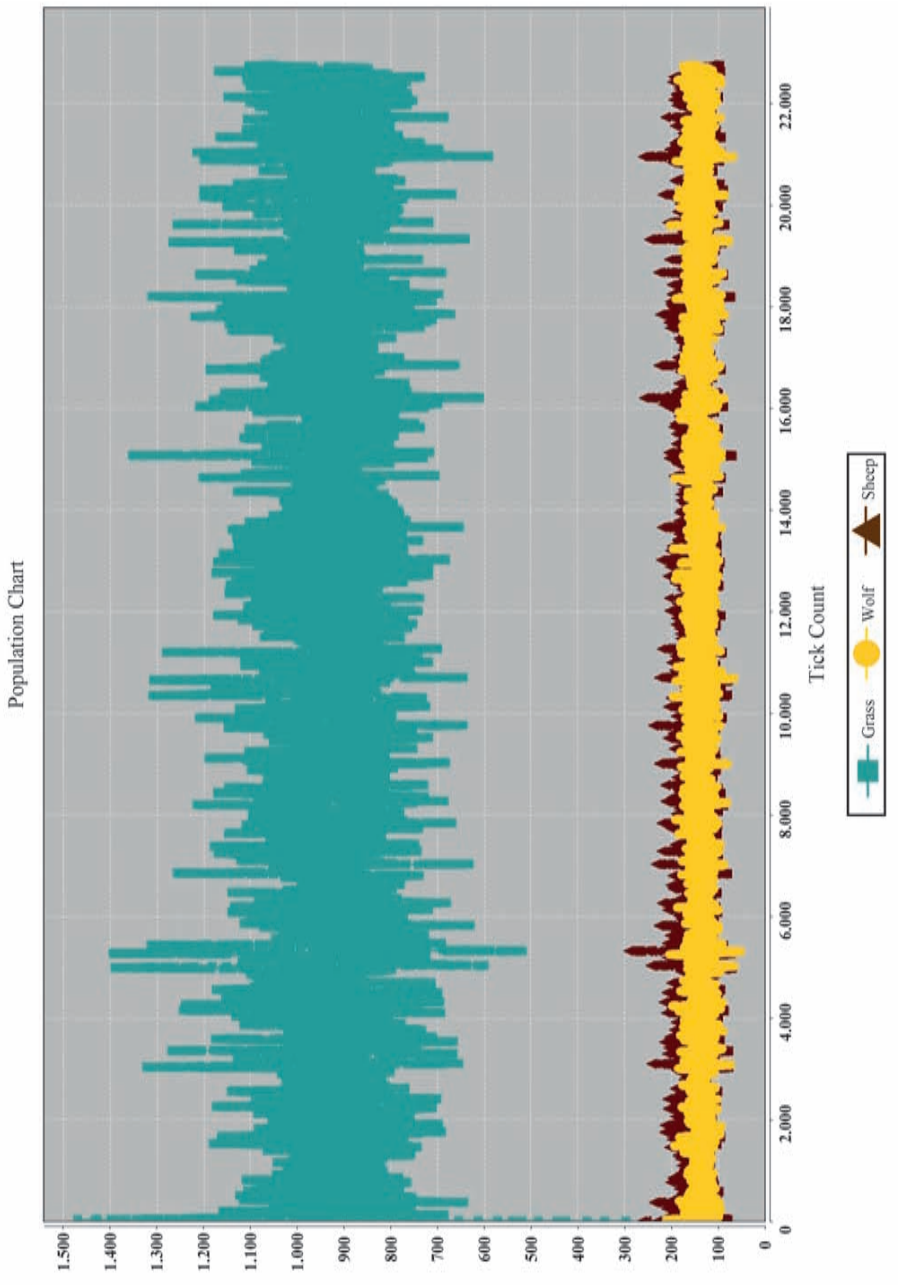

Figure 12. Manual run of the best parameter values obtained with the AGA for the prey predator model.

Table 5. Numerical test functions used.

Name of function	Formula	Min small point	Search interval
Rastrigin	f(x)=An+$\sum_{i=1}^{n}[x_i^2 - Acos(2\pi x_i)]$,A=10	f(0,0) = 0	$-5,12 \leq x,y \leq 5.12$
Sphere	f(x)=$\sum_{i=1}^{n} x_i^2$	f(0 , … ,0) = 0	$-\infty \leq xi \leq \infty$, $1 \leq i \leq n$
Rosenbrock	f(x)=$\sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	f(1,…,1) = 0	$-\infty \leq xi \leq \infty$, $1 \leq i \leq n$
Griewank	f(x)=1 + 1/400$\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(x_i/i)$	f(0,…,0)=0	$-600 \leq xi \leq 600$, $1 \leq i \leq n$

4. Discussion and conclusion

In the present study, the parameter-setting operation in the prey predator model was performed using the GA, AGA, ORJ-ABC, and MABC algorithms. The results obtained prove that the MABC algorithm is good at parameter setting in complex systems. However, this same success was not shown by the ORJ-ABC algorithm. In reference to the other findings, the value of the critical parameter for the GA and the MABC algorithm affected the optimum result. There is a need to vary the critical parameter value used in accord with the structure of each of the problems. However, meeting this need by manually setting the critical parameter value of the MABC algorithm is difficult and time consuming. Therefore, there is a need for approaches that can adjust critical parameters of MABC and other MHAs. This is because the adaptive algorithms (AMABC and AGA) presents the best solutions to each of the problems investigated. The AMABC algorithm is much more successful than either the GA or the ORJ-ABC. In addition, it is seen in the present study that the AGA is more successful in this problem. The MABC algorithm can be used in both setting the model parameters for complex systems modeled in and close to the optimum in numerical test functions. Thus, the AMABC algorithm is significant in terms of its ability to adapt to the problem, which means this algorithm generates the best solutions to the optimization problems investigated. In the present study, the OAPS algorithm was developed into the AMABC algorithm to the problem. The OAPS algorithm is important in that it can be adapted to existing ABC algorithms. Thus, algorithms that can be adapted to the problem are developed. This will increase the problem-solving success of the algorithms.

Table 6. Algorithms and optimization results used to optimize the test functions.

Function	Algorithm	Working 1	Working 2	Working 3	Working 4	Working 5	Average
Rastrigin	GA	51,441694	26,240791	42,912781	37,892368	46,646883	41,026900
	AGA	0,0911682	0,1656174	0,1342727	0,0634769	0,1370721	0,1183215
	ORJ-ABC	51,098063	65,145178	37,602278	50,545854	68,904444	54,659163
	MABC-10	0,59434	0,632331	0,599457	0,729784	0,340679	0,5793183
	MABC-50	0,758938	1,397699	0,592751	0,508514	1,170497	0,8856798
	MABC-100	0,929969	1,01314	1,576475	0,852173	0,449002	0,9641519
	AMABC	0,0400609	0,0700237	0,0584063	0,0726403	0,0809132	0,0644089
Sphere	GA	9,251076	15,26091	4,763601	9,72903	13,11957	10,42484
	AGA	0,0372451	0,0172696	0,1120942	0,0681774	0,0188995	0,0507372
	ORJ-ABC	15,9802	10,3251	19,04103	9,302802	11,07244	13,144314
	MABC-10	0,891958	0,259195	0,231424	0,566871	1,100572	0,6100039
	MABC-50	0,891958	0,259195	1,710638	0,231424	0,566871	0,7320171
	MABC-100	1,125044	0,825749	0,507663	1,089862	0,185659	0,7467955
	AMABC	0,1721976	0,0597481	0,1109223	0,158777	0,1721455	0,1347581
Rosenbrock	GA	194,2464	82,97482	335,1893	402,0784	47,45999	212,3898
	AGA	0,1340479	0,0948623	0,033258	0,0348619	0,0541642	0,0702389
	ORJ-ABC	177,0827	368,4447	88,56046	765,1089	53,08777	290,45691
	MABC-10	0,20209	0,062965	0,11898	0,022972	0,124672	0,1063359
	MABC-50	0,06319	0,110253	0,046529	0,165417	0,030051	0,0830883
	MABC-100	0,179807	0,250297	0,066753	0,037296	0,062862	0,1194029
	AMABC	0,0191021	0,0129505	0,022575	0,0153738	0,0325735	0,020515
Griewank	GA	0,010298	0,003482	0,003722	0,020253	0,010115	0,009574
	AGA	0,1293177	0,1000618	0,1324447	0,2947363	0,0969213	0,1506963
	ORJ-ABC	0,020238	0,018766	0,015367	0,014199	0,019891	0,0176922
	MABC-10	0,928191	0,434277	1,02874	5,158047	0,689556	1,6477622
	MABC-50	2,969948	2,2972	2,056117	2,361782	0,371162	2,0112421
	MABC-100	1,318057	3,813994	1,85888	1,534777	0,96373	1,8978876
	AMABC	0,3612823	0,1465825	0,7043986	0,7648852	0,1113145	0,4176926

References

[1] Di Marzo Serugendo G, Gleizes MP, Karageorgos A. Self-organising Software From Natural to Artificial Adaptation. Berlin, Germany: Springer-Verlag, 2003. pp 7-32.

[2] Kaddoum E, and George J-P. Collective Self-Tuning for Complex Product Design (short paper). In: IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO); Lyon, France; 2012. pp. 193-205.

[3] Guivarch V, Camps V, Peninou A. AMADEUS: an adaptive multi-agent system to learn a user's recurring actions in ambient systems: Advances in Distributed Computing and Artificial Intelligence Journal 2012; 3: 1-10. doi: 10.14201/ADCAIJ2012131110

[4] Korkmaz Tan R, Bora Ş. Parameter tuning algorithms in modeling and simulation. International Journal of Engineering Science and Application 2017; l: 58-66.

[5] Korkmaz Tan R, Bora Ş. Modelleme ve benzetim ortamında parametre optimizasyonu ve kullanılan teknikler. Mühendislik Bilimleri ve Tasarım Dergisi 2017; 5: 685-697 (in Turkish). doi: 10.21923/jesd.307125

[6] Calvez B, Hutzler G. Automatic tuning of agent based models using genetic algorithms. In: Proceedings of the 6th International Workshop on Multi-Agent Based Simulation (MABS'05); Utrecht, Netherlands; 2005, pp. 41-57.

[7] Imbault F, Lebart K. A stochastic optimization approach for parameter tuning of support vector machines. Proceedings of the 17th International Conference on Pattern Recognition ICPR; Cambridge, UK; 2004. pp. 597-600.

[8] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 2007; 39: 459-471. doi: 10.1007/s10898-007-9149-x

[9] Bhambu P, Sharma S, Kumar S. Modified gbest artificial bee colony algorithm. In: Soft Computing: Theories and Applications. Berlin, Germany: Springer, 2018. pp 665-677. doi: 10.1007/978-981-105687-1_59

[10] Agarwal SK, Yadav S. A Comprehensive Survey on Artificial Bee Colony Algorithm as a Frontier in Swarm Intelligence. In: Hu YC, Tiwari S, Mishra K, Trivedi M (editors). Ambient Communications and Computer Systems. Advances in Intelligent Systems and Computing Vol. 904. Singapore: Springer, 2019, pp 125-134. doi: 10.1007/978-981-13-5934-7_12

[11] Neelima S, Satyanarayana N, Murthy PK. A Comprehensive Survey on Variants in Artificial Bee Colony (ABC). International Journal of Computer Science and Information Technologies 2016; 7 (4): 1684-1689.

[12] Ponton J, Klemes J. Alternatives to neural networks for inferential measurement. Computers and Chemical Engineering 1993; 17 (10): 991-1000. doi: 10.1016/0098-1354(93)80080-7

[13] Karaboga N. A new design method based on artificial bee colony algorithm for digital IIR filters. Journal of the Franklin Institute 2009; 346 (4): 328-348. doi : 10.1016/j.jfranklin.2008.11.003

[14] Zhu G, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Compution 2010; 217 (7): 3166-3173. doi: 10.1016/j.amc.2010.08.049

[15] Bansal JC, Joshi SK, Sharma H. Modified global best artificial bee colony for constrained optimization problems. Computers and Electrical Engineering 2018; 67: 365-382. doi: 10.1016/j.compeleceng.2017.10.021

[16] Karaboga D, Gorkemli B. A quick artificial bee colony -qABC- algorithm for optimization problems. 2012 International Symposium on Innovations in Intelligent Systems and Applications; Trabzon, Turkey; 2012. pp. 1-5.

[17] Karaboga D, Gorkemli B. A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Applied Soft Computing 2014; 23: 227-238. doi: 10.1016/j.asoc.2014.06.035

[18] Li X, Yang G. Artificial bee colony algorithm with memory. Applied Soft Computing 2016; 41: 362-372. doi: 10.1016/j.asoc.2015.12.046

[19] Aslan S, Badem H, Karaboga D. Improved quick artificial bee colony (iqABC) algorithm for global optimization. Soft Computing 2019; 23: 13161-13182. doi: 10.1007/s00500-019-03858-y

[20] Eiben AE, Hinterding R, Michalewicz Z. Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 1999; 3 (2): 124–141. doi: 10.1109/4235.771166

[21] Adenso-Diaz B, Laguna M. Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 2006; 54 (1): 99-114. doi: 10.1287/opre.1050.0243.

[22] Bartz-Beielstein T, Parsopoulos K, Vrahatis M. Analysis of particle swarm optimization using computational statistics. Proceedings of the International Conference of Numerical Analysis and Applied Mathematics (Icnaam 2004). Chalkis, Greece: Wiley, 2004, pp. 34-37.

[23] Nannen V, Eiben AE. Efficient Relevance Estimation and Value Calibration of evolutionary algorithm parameters. In: IEEE Congress on Evolutionary Computation; Singapore; 2007. pp. 103-110.

[24] Dobslaw F. A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. Proceeding of the International Conference on Computer Mathematics and Natural Computing; Rome, Italy; 2010. pp. 213-216.

[25] Korkmaz Tan R, Bora Ş. Parameter tuning of complex systems modeled in agent based modeling and simulation. World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering 2017; 11: 1314-1323. doi: 10.5281/zenodo.1314905

[26] Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization. Information Sciences 2010; 192: 120-142: doi: 10.1016/j.ins.2010.07.015

[27] Singh A. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Applied Soft Computing 2009; 9: 625-631. doi: 10.1016/j.asoc.2008.09.001

[28] Korkmaz Tan R, Bora Ş. Adaptive parameter tuning for agent-based modeling and simulation. Simulation: Transactions of the Society for Modeling and Simulation International 2019; 95 (9): 771–796. doi: 10.1177/0037549719846366

[29] North MJ, Tatara E, Collier N, Ozik J. Visual agent based model development with Repast Simphony. Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence; Argonne National Laboratory, Argonne, IL USA; 2007. pp. 1-20.