World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:11, No:12, 2017

Parameter Tuning of Complex Systems Modeled in Agent Based Modeling and Simulation

Rabia Korkmaz Tan, Şebnem Bora

***Abstract*—**The major problem encountered when modeling complex systems with agent-based modeling and simulation techniques is the existence of large parameter spaces. A complex system model cannot be expected to reflect the whole of the real system, but by specifying the most appropriate parameters, the actual system can be represented by the model under certain conditions. When the studies conducted in recent years were reviewed, it has been observed that there are few studies for parameter tuning problem in agent based simulations, and these studies have focused on tuning parameters of a single model. In this study, an approach of parameter tuning is proposed by using metaheuristic algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colonies (ABC), Firefly (FA) algorithms. With this hybrid structured study, the parameter tuning problems of the models in the different fields were solved. The new approach offered was tested in two different models, and its achievements in different problems were compared. The simulations and the results reveal that this proposed study is better than the existing parameter tuning studies.

***Keywords*—**Parameter tuning, agent based modeling and simulation, metaheuristic algorithms, complex systems.

I. Introduction

AGENT based modeling and simulation technique is often used in modeling complex systems [1]. Modeling of these systems has necessitated parameter tuning. Parameter tuning is an optimization problem. Mathematical methods were initially used in the solution of these problems. In order for the mathematical method to be used, the problem must be defined by mathematical functions. This process is almost impossible in complex systems. The heuristic algorithms, formed by being inspired from the events in nature, are frequently used in optimization problems in our day. Moreover, it has been supported by studies that metaheuristic algorithms are the most suitable methods that can be used in the tuning of these parameters of these systems [2]-[4]. The most successful algorithms, used in the field of optimization, were used in this study. These algorithms are GA, PSO, ABC, FA algorithms.

GA is an intuitive search algorithm. It aims to offer solutions using mutation and crossover methods, by being inspired from natural selection and genetic issues [5].

The PSO is a swarm-based heuristic optimization algorithm. It was developed by being inspired from the movements of swarm, especially from covey. The movements of birds in covey according to the position of those that are close to the feed underlie the basis of the algorithm [6].

The ABC is a swarm-based heuristic optimization algorithm. The ABC was constituted by being inspired from the nutritional search behaviors of bees. The nutrition search and processing behaviors of employed, onlooker, and scout bees were adapted to the algorithm [7].

FA is also a swarm-based heuristic optimization algorithm. It was developed by taking fireflies' brightness-sensitive social behaviors into consideration. Algorithm treats fireflies without taking their sex into consideration. In the other words, all fireflies can go towards each other. More brilliant fireflies are more attractive. Less bright fireflies go towards those that are attractive. Since the effect of brightness will decrease as distance increases, the more the distance from bright fireflies increases, the less fireflies are impressed by them. If a firefly becomes unable to find a firefly that is brighter than itself, it moves randomly [8].

The algorithms used in this study were tested using different models and numerical test functions. According to the results obtained, it can be said that the success of each algorithm depends on the needs of the problem. While GA performs parameter tuning process more successfully than others in one model, a different algorithm can produce better results in a different model. When the studies conducted were reviewed, it is seen that while working at high speed with acceptable parameter values is more important for some models in some studies, the speed has secondary importance and working with best parameter values is more important for the model in some studies. This approach differs from other studies from the point of satisfying both situations.

Afterwards, the algorithms will be explained with their main lines in the study, and then the algorithms tested using two different models will be compared. The success and performance of the algorithm on each model and function will be compared.

II. Methods

A. Genetic Algorithm (GA)

GA aims at 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, multi-dimensional search space, according to the principle of survival of the best [9]. While GA parameters refer to the genes in the 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 as a population. The fitness of the population is maximized or

Rabia Korkmaz Tan is with the Ege University, Izmir, Turkey (e-mail: rabia_korkmaz@yahoo.com).

minimized within certain rules. Each new generation is acquired by combining survivors within the sequences created by random information exchange.

There were 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 steps of the GA are as follows:

1) An initial population is created in which the possible solutions are coded. There is no standard for the number of individuals to be determined in the population, it may vary according to the type of the problem. The gene values of each individual can be assigned according to (1). The number of gene is also the number of parameters of the problem to be solved.

$$P_{i_k} = minP_j + rand(0,1) \times (maxP_j - minP_j) \quad (1)$$

i refers to the index number of the selected solution set, k refers to the index number of the selected parameter, P_{ik} refers to the valid parameter number, *rand(0,1)* refers to the random number between 0 and 1, $minP_j$ refers to the minimum number that can be assigned to the parameter, $maxP_j$ refers to the maximum number that can be assigned to the parameter.

2) The appropriateness value of each chromosome is calculated using the fitness function (the function related to problem, which is created by user, and meanwhile, critical parameter variables are considered). The success of the GA is often dependent on the good detection of this function. The best individual in this step is recorded as a local solution.
3) It is important to select the individuals with the best fitness value, as well as the percentage of the selection. The selection is made using Tournament and Roulette Wheel method mostly preferred in GA.

a. Tournament Method: In this method, two for each fitness value that are included in the random sequences, selected from the population at every turn, are compared, and assigned to the new sequence. Each individual participates in the tournament for 2 times.
b. Roulette Wheel Method: The percentage of area covered by the individuals with the best fitness value in the roulette wheel is higher, thus increasing the probability of the random numbers generated to be yielded in the large area, which increases the probability of selection of the individuals with the best fitness value.

4) The individuals obtained after the selection process are randomly matched, and the part replacement process (crossover) is performed between the chromosomes of two selected individuals. One-Point Crossover Method or Two-Point Crossover Method can be used for the crossover process.

a. One-Point Crossover Method: One of the parameter points is randomly selected. A new individual is created by taking the parameter values from the first individual up to that point, and from the second individual after that point.
b. Two-Point Crossover Method: In this method, two points are selected without including the first and last parameters. The genes between these two points are taken from the 1st individual, and the rest of the genes are taken from the 2nd individual, thus a new individual is created.

5) The migration effect can be used during the selection phase [10], which allows the algorithm to perform a large area search by preventing it from focusing on the overall optimum solution. This feature is extracted from the individual population where the worst individuals selected by the inverse roulette method (i.e., the selection probability of the worst individuals selected is higher), instead, random individuals are produced externally and included in the population. This event can be performed once in each generation. This rate can be increased depending on the request.
6) The next step is mutation. Once or more times, the parameter of individual is modified, depending on the percentage of mutation of the new individual created.
7) In step 1, the parameter value range was set, and parameter values were assigned. It can be checked whether the determined range has been exceeded.
8) The number of generations initially determined as a stop criterion, or any ideal global solution can produce a solution. The stop criterion is tested. If the criterion is satisfied, the algorithm is stopped, and the final attained local solution is regarded as the global solution. If the criterion is not satisfied, it is proceeded to the next step and continued to seek for new solutions. If the criterion is satisfied, it is returned to the step 2, and the processes are repeated.

B. Particle Swarm Optimization (PSO)

It is an algorithm based on swarm intelligence. It has been observed that birds that move in swarm can easily reach to food sources, with their random position changes. PSO is based on the social information sharing of each individual in the swarm. The search process is performed until the number of generation specified as the criterion is reached. Each individual is called as a particle, the population, consisting of the community of particles, is called as a swarm [6].

The work steps of PSA are as follows:

1) PSO first constitutes the solution set through the randomly generated start positions and speeds. The solution set (particle) consists of N elements. ith particle is referred to as $x = x=[x_{i1}, x_{i2},..., x_{iN}]$. Each element in the solution set corresponds to either the problem dimension, or to the problem parameters. Each element of the particle can be assigned using (1) in the GA.
2) After the values of the elements have been assigned, the fitness value of each particle is calculated using the fitness

function which is associated with the model or with the fitness function that will be adapted to another problem (this function varies depending on the model, problem, and expectations of users).

3) It is attempted to reach to the optimum solution by continuous updating of the speeds and positions of the particles. The particle positions are updated at each iteration according to their optimum position (pbest), and to the best position (gbest) of swarm, and stored in memory. After finding the pbest and gbest values of position at each iteration, the speed and position of particle are updated according to (2) and (3), respectively.

$$v_{ik} = w.v_{ik} + c_1.rand_{1k}\,(pbest_{ik} - x_{ik}) + c_2.rand_{2k}(gbest_k - x_{ik}) \quad (2)$$

$$l_{ik} = l_{ik} + v_{ik} \quad (3)$$

As can be seen, the equations used to update the position value of particles consist of simple sum and multiplication, and do not require derivative information. $pbest_{ik,}$ is the coordinate that provides the best solution for ith particle until that moment. $gbest_{k,}$ is the coordinate value that provides the best solution found so far. The c1 and c2 values, the learning factors, are the constants that determine the approach ratio for the pbest and gbest values, respectively. Rand1 and rand2 in the equation refer to the randomly assigned numbers between 0 and 1; k refers to the iteration value; i refers to the index number of the selected solution set, l_{ik} refers to the position of the particle; v_{ik} refers to the speed value of particle, newly constituted; and w refers to the inertia weight [11].

Intertia Weight (w)

Intertia weights are an important PSO parameter. It has been proven by the studies that intertia weights have greatest effect on the convergence of the optimum value in the PSO process. There are many strategies of inertia weight. Linear decreasing intertia weight strategy yields more approximate results to the optimal results, compared to another strategy [12]. Linear decreasing intertia weight equation is defined as in (4):

$$w_k = w_{max} - ((iter) / maksIte) * (w_{max} - w_{min}); \quad (4)$$

C. Artificial Bee Colony (ABC) Algorithm

The ABC, a swarm-based heuristic algorithm, was developed (written) by being inspired from the nutritional search behaviors of bees [7]. The nutrition search and processing behaviors of employed, onlooker, and scout bees were adapted to the algorithm. The ABC algorithm has been widely used for solving many optimization problems due to the fact that it has limited number of control parameters, simple, and easy-to-develop [13], [14].

There are three types of bee in ABC algorithm as employed, onlooker, and scout bees in ABC algorithm. Employed bees are responsible for calculating the amount of nectar of each food source, and the number of employed bees in the population is equal to the number of food sources in the area of nutrition. Onlooker bees are responsible for choosing a food source that has a good amount of nectar, and the number of onlooker bees in the population equals the number of employed bees. Scout bees are responsible for discovering new food sources. The employed bees, of which nutrition sources have been consumed, become scout bees, and the old food source can be replaced by a new food source found by scout bees.

The position of the nutrition sources represents the possible solution of the optimization problem, intended to be solved. The high nectar content of nutrition source means that the possible solution of optimization problem is good. Therefore, the quality of the possible solution is represented by the amount of nectar, and this value is called as the *fitness value* in the ABC algorithm.

ABC's work steps are given below;

1) First, random food sources, that is to say, solution sets are created. The number of elements in each set varies depending on the size of the problem. As well as the values of these elements can be randomly produced, they can also be optionally assigned using (1), specifying the minimum maximum limits.
2) After the solution sets have been constituted, the fitness value of each solution set is calculated using the fitness function of the model, or that of the problem. This function may vary depending on the user's expectation of the model. The obtained fitness value is applied to the fitness function (5) of the ABC algorithm to calculate the final fitness value.

$$\begin{aligned} &\text{if } f_i \geq 0 \rightarrow \frac{1}{1+f_i} \\ &\text{if } f_i < 0 \rightarrow 1 + |f_i| \end{aligned} \quad (5)$$

f_i is the fitness value of the ith solution set of the optimization problem or of the model.

3) Equation (6) is applied to each of the produced solution sets that have been created, thus allowing for new candidate solutions to be obtained. The fitness value of the obtained solution set is calculated, and compared with the fitness value of the old solution set. If there is an improvement in the fitness value, the solution set is replaced by the old one. The goal in this step is to improve the nutrition source, that is, the solution set.

$$\begin{aligned} &\text{if } R_{ik} < MR \text{ then } V_{i_k} = X_{i_k} + \varphi_{i_k} \times \left(X_{i_k} - X_{j_k}\right) \\ &\text{else } V_{i_k} = X_{i_k} \end{aligned} \quad (6)$$

V_{ik} refers to the new solution to be obtained, X_{ik} the old candidate solution, φ_{i_k} the random number in the range of [-1,1], X_{kj} the selected neighbor solution, R_{ik}, randomly produced value, MR the ratio that determines how many elements are needed to be replaced, instead of single element replacement, in the production of neighbor solution. Afterwards, onlooker bees attempt to find new good food sources in a similar way as employed bee. Unlike employed bees, the possibility of selection of the solution sets of onlooker bees are determined according to their fitness values.

Then, the solution sets with the best fitness value are selected using the roulette wheel method. Then, by applying (7) to the selected solution sets, a better solution is tried to be created. If the fitness value is improved, then, the old solution set is replaced with the new solution set. If no improvement is observed, then the limit value of that solution set is increased by one.

$$O_i = \frac{F_i}{\sum_j F_j} \quad (7)$$

O_i refers to the probability of selection of the solution set for onlooker bees, F_i the fitness value of the selected solution set, $\sum_j F_j$ the sum of the fitness values of all solution sets.

4) If the nutrition source is exhausted in the last step, employed bees become scout bees, and begin to search for nutrition source. If nutrition source cannot be improved along the specified limit value, it is deleted. Instead of that, a new set of solutions is created randomly, thus, allowing the algorithm to avoid from the best local solutions.

D. Firefly Algorithm (FA)

FA [8] is another swarm-based heuristic optimization algorithm. It was developed by discussing the social behaviors of fireflies. It is used in many optimization problems because of its advantages such as having limited number of parameters, being easy to understand, easy to implement.

The FA is an algorithm, being fictionalised on the brightness-sensitive behaviors of fireflies. The important criteria in the algorithm are the ability of all fireflies to impress each other regardless of sex, their *attractiveness,* and brightness. The less bright firefly comes close to the brighter firefly. The more the distance increases, the more the influence of brightness decreases. Therefore, a firefly that cannot find a brighter firefly than itself moves randomly, which increases the possibility of finding a brighter firefly. N fireflies, that is, solution sets are used in an optimization problem. Each solution set contains elements as much as the problem size. At the beginning, it takes values at the specified intervals, or random values without specifying any interval.

The fitness value of the obtained solution set is calculated through the fitness function. The fitness value represents brightness. The firefly with the best fitness value is the brightest one.

The work steps of the FA Algorithm are as follows:

1) As the first step, a set of random solutions are created at the specified intervals (the number of the elements depends on the size of the problem) for each element of each solution set. These values can optionally be assigned using (1).
2) In this step, the light intensity of fireflies, the distance between fireflies and their light intensities are calculated using (8)-(10) respectively.

$$I = 1/r^2 \quad (8)$$

I refers to the intensity of light, r, the distance between two fireflies.

$$r_{ij} = |x_i - x_j|^2 = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \quad (9)$$

r_{ij} refers to the distance between i th and jth fireflies, d the element number, $x_{i,k}$, k th. element of the solution set of ith firefly, and $x_{j,k}$ k th. element of the solution set of jth firefly

$$I(r) = I_o e^{-\gamma r^2} \quad (10)$$

I_0 refers to the initial light intensity, the distance between two fireflies, γ refers to light absorption coefficient, r, the distance between two fireflies.

3) The attractiveness of a firefly depends on the brightness and distance of other firefly. Therefore, the attractiveness is calculated with (11):

$$\beta = \beta_o e^{-\gamma r^2} \quad (11)$$

β is the attractiveness value of a firefly. β_0 is the attractiveness value when the distance between two fireflies is 0. β_0 can take values between 0 and 1.

4) The movement of fireflies towards brighter fireflies: each firefly controls all other fireflies, and they move towards fireflies that are brighter than themselves, according to (12):

$$x_{i,p} = x_{i,p} + \beta(x_{j,p} - x_{i,p}) + \alpha\varepsilon_{i,p} \quad (12)$$

$x_{i,p}$ refers to the pth element value of ith firefly, $x_{j,p,}$ pth element value of jth firefly . α random variable usually takes a value between [0,1], Ɛi is determined by Gaussian distribution, but generally represents a random value in the range of [-0.5,0.5]

5) If the firefly fails to find a firefly brighter than itself, move randomly according to (13):

$$x_{i,p} = x_{i,p} + \alpha\varepsilon_{i,p} \quad (13)$$

6) The cycle is repeatedly operated from the 2nd step until the maximum iteration is reached. By comparing the results obtained, the best solution set is found.

III. Experimental Studies

Through this hybrid approach developed using metaheuristic algorithms, predator-prey which is a factor-based model and the model parameters belonging to eight queens were set.

The parameter interface of this approach is shown in Fig. 1. The users can select the algorithm to be used for each model, and can manually enter the parameters of the algorithm used. Besides, the parameters of model can be optionally manually entered. Thus, manually entering the best parameter set produced by the algorithms allows for retesting the performance of this parameter set. A distinctive parameter

tuning process is performed on each algorithm model. Thus, we can observe which metaheuristic algorithm is more successful in which problem.

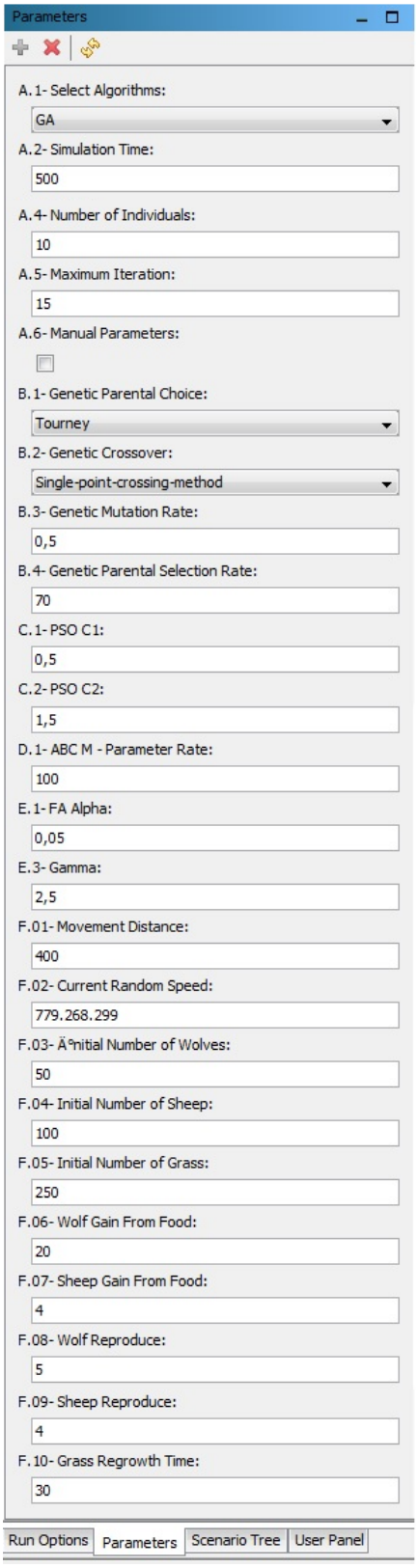

Fig. 1 Parameter Interface

The models that we use to demonstrate the effectiveness of our approach that we recommend to solve the problem of parameter tuning are introduced below.

A. Used Models

1. Predator-Prey Model

The goal of the model is to determine the reasons, having an effect on the continuity of predator-prey ecosystem [15]. Wilensky suggests that the huntee–hunter binaries of predator-prey model, included in an ecosystem, are associated with nourishment (hunting), reproduction and death behaviors. In this simulation model used to test the developed approach, the wolf-lamb ecosystem was investigated. The values of the parameters such as the relation of the predator-prey population with each other, and the influence of nutrition included in the ecosystem on the population were set through the approach. The objective is to ensure the continuity of the population in ecology, with the most appropriate parameter values. There are three agents in the predator-prey simulation model: wolf, sheep and grass, and sheep agents (agent) move randomly in the simulation environment. But, there is a cost for both wolf and sheep agents to walk in the simulation environment. Thus, they lose energy as long as they continue to move in the environment. The wolf and sheep agents are born having a certain initial energy. But, they need hunting to sustain their lives in the environment. If their energy values decreases to 0 level, they will die.

If a wolf meets with a sheep in the same coordinate as itself, hunts it, and increases its energy according to the specified parameter value (*wolf gain from food*). Similarly, if a sheep agent meets with an edible and live grass agent, eats it, and increases its energy according to the specified parameter value (*sheep gain from food*). Grass agents are found at every coordinate in the environment. After agents are created, they grow until the specified value (grass regrow time). After the grass is eaten by a sheep, its condition is updated to be *dead*. The life cycle of grass is provided by a randomly assigned life cycle *(countdown*). Grass die at the end of their lifetime, even if they are not eaten in the life cycle of the model. During the life cycle of grass, it becomes edible again after growth time value has been exceeded, and its state value is updated to be *alive* . Thus, these parameter values are set with this approach, and the best parameter set allowing for the population continuity is reached [16].

a. Eight Queens Model

It is aimed to place eight queens to a chess board of 8x8 in such a way that they cannot get each other. This model has eight parameters in total. Each of these parameters has the position value of a queen on the chessboard. Each queen can be placed on a chess board in 64 different ways. Through this approach that we have developed, eight queens in the model were positioned on the chessboard without overlapping.

2. Simulation Details

The information about the PC features used for the experiments, and about the tool and programming language for which the approach is developed are given in Table I.

The common parameter values for all algorithms, that have an important effect on the operation of the model used for all algorithms and on the parameter tuning, given fixedly, and can be changed manually by user, are given in Table II. The duration of the simulation gives the tick count for which model is operated for each set of solution, and this value can be increased optionally. The solution set is the population that is created by each algorithm, based on the problem. The number of iteration shows how many generations will be created and simulated. Increasing the number of iteration will help to obtain the optimum solution.

TABLE I
PC DETAILS USED FOR EXPERIMENTS

System	Windows 7 professional 64 bit
RAM	8 GB
CPU	Intel(R) Core(TM) i7 2.10 GHz
Algorithm	GA, PSO, ABC, FA
Prog. Language	Repast Symphony 2.3.1-Eclipse-Java

TABLE II
COMMON PARAMETERS USED FOR ALL ALGORITHMS

Parameter Variable	Predator-Prey Model	Eight Queens Model
Simulation Time	500 Tik	1 Tik
Number of solution sets	10	20
Maximum Iteration	15	1000

The constant parameter values of the algorithms, used in the developed approach, are given in Table III. These values can be changed optionally, or based on the model, and their effect on the results can be observed.

TABLE III
PARAMETER VALUES USED IN ALGORITHMS

GA	Predator-Prey	Eight Queens
Selection Method	Tournament	Tournament
Crossover Method	One-Point Crossover	B One-Point Crossover
Parent Percentage	70	70
Percent Mutation	0.5	0.5
PSO		
C1	0.5	0,5
C2	1,5	1,5
ABC		
MR	100	10
FA		
Alpha	0.05	0,05
Gama	2,5	2,5

3. Results

a. Predator-Prey Model

The data in Table IV are the average of the results obtained after 10 runs of each algorithm in the predator-prey model used.

TABLE IV
METAHEURISTIC ALGORITHM RESULTS IN HUNTEE HUNTER OPTIMIZATION

	Tick	Time	Average fitness value	Initial best fitness value	Best fitness for all simulation process	Improvement Difference	Average local fitness value	Average General fitness value
FA	70089.2	249.20	0.42	0.48	0.08	0.40	0.23	0.19
PSO	54432.4	190.14	0.57	0.61	0.12	0.39	0.31	0.23
GA	25265.4	44.93	0.40	0.32	0.07	0.25	0.14	0.13
ABC	38768.4	93.75	0.73	0.45	0.10	0.35	0.39	0.24

Explanation of Columns in Table IV:

Tick: The average of tick counts in each operation of simulations

Time: The average of the elapsed times when simulations are over.

Average Fitness Value: The average of fitness values of solution sets in each iteration.

Initial Best Fitness Value: The average of the best fitness value of randomly created solution sets in the first generation in simulations.

Best Fitness Value: The average of the fitness values of the best solution sets yielded after simulations**.**

Improvement Difference: The difference between the " Initial Best Fitness" and "Best Fitness" values.

Average Local Fitness Value: The average of local values in each iteration in simulations.

Average General Fitness Value: The average of general values in each iteration in simulations.

When Table IV is reviewed, it is observed that GA is much faster than other algorithms. It is based on the structure of the algorithm. There is the percentage of selection of parents in GA. This parameter determines how many individuals will be created in each iteration. This value is identified as 70% for GA, and seven new individuals are created in each iteration. 10 individuals are created for each iteration in other algorithms. This feature allows GA to operate faster. The second important reason is that a set of solutions similar to each other is created more than other algorithms, since the solution sets, generated in GA, are crossed. The fitness values are recorded to reduce the calculation burden in this approach, and the fitness value of the solution sets, the same as each other, are not recalculated. This feature contributes to the faster run of GA. Moreover, it was observed that the FA algorithm runs very slowly compared to other algorithms. The reason for this is that the FA algorithm uses more processors than other algorithms; because, FA contains more cycles and operations than other algorithm. Each firefly has to control the brightness of all fireflies so that it can produce a new set of solutions.

When Table IV is reviewed with respect to the optimum result, it is observed that GA again produces better results than the other algorithms. But, when considered that FA's best fitness value is very approximate to the GA's, and FA's improvement difference is higher than GA's, it can be concluded that FA algorithm is better than GA algorithm in terms of achieving optimum solution.

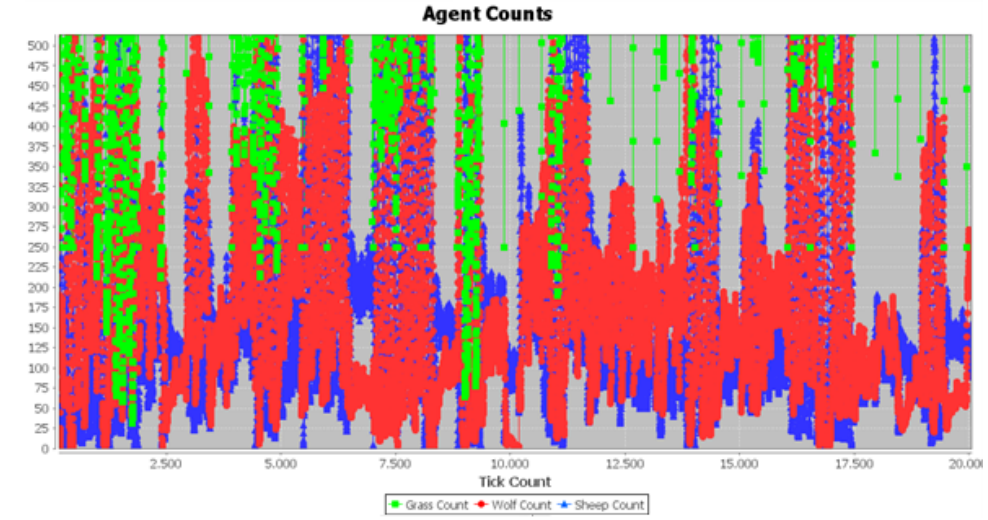

Fig. 2 Population Distribution of Predator-Prey Model in Parameter Tuning Process

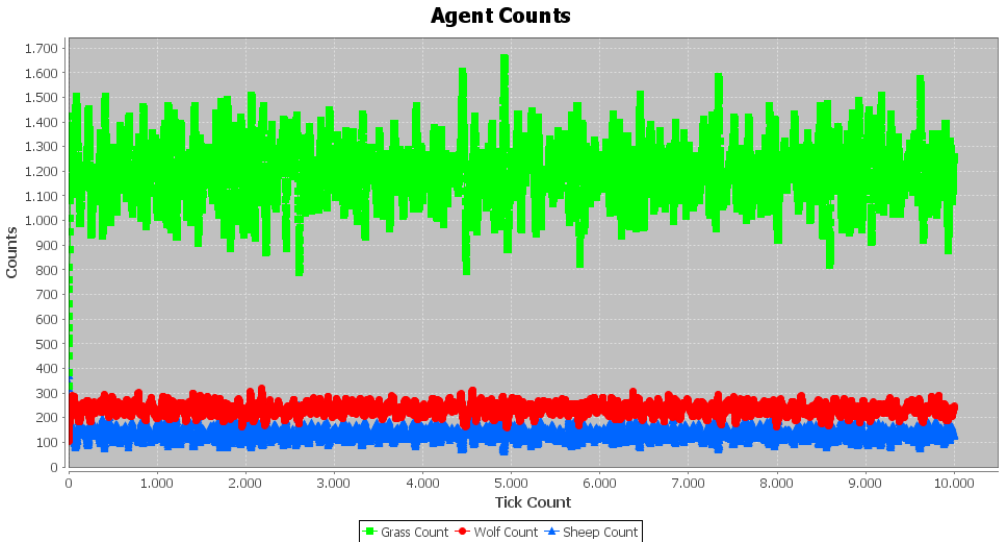

Fig. 3 Population Graph obtained from GA's adjusted parameter values

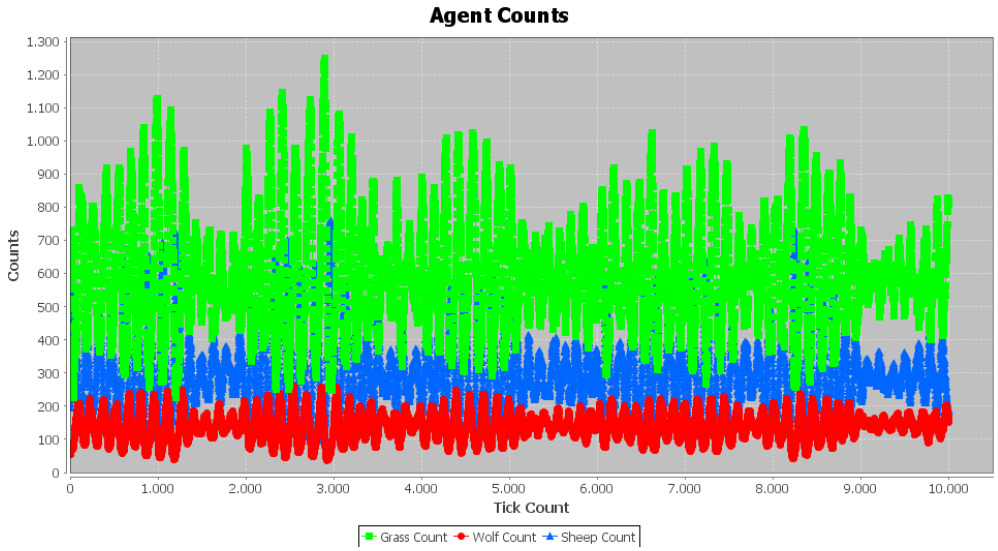

Fig. 4 Population Graph obtained from PSO's adjusted parameter values

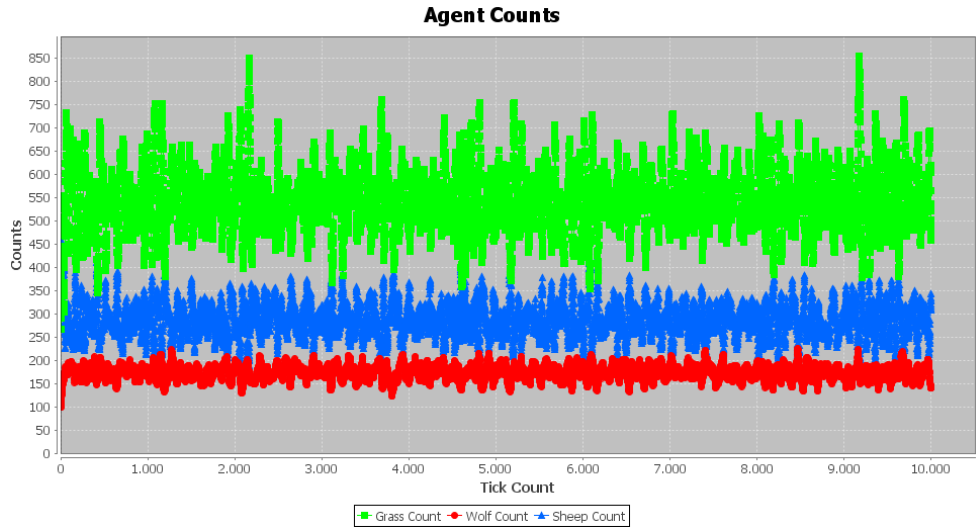

Fig. 5 Population Graph obtained from ABC's adjusted parameter values

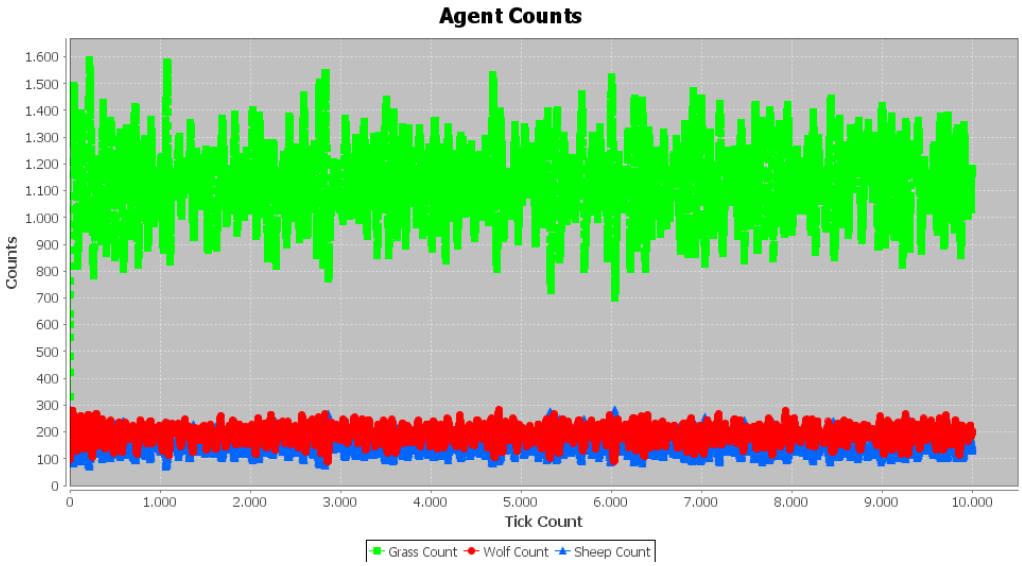

Fig. 6 Population Graph obtained from FA's adjusted parameter values

Fig. 2 shows the graph obtained after the simulations of all solution sets created while attempting to find the appropriate parameter value for the predator-prey model. As seen in the graph, the simulation results of good and bad solution sets are available.

Figs. 3-6 are the graphs obtained by manually running the best parameter values, found through GA, PSO, ABC and FA algorithms, respectively. When the graph obtained from these four algorithms is analyzed, it is seen that all these algorithms contain solution sets that provide the continuity of the simulation.

b. Eight Queens Model

The data in Table V are the average of the results obtained after 10 runs of each algorithm in the used eight queens model. According to the data obtained from the eight queens problem, it is seen that the fitness values of ABC and GA algorithms are much better than those obtained from PSO and FA algorithms. In the model where each algorithm was run for 1000 ticks, unlike other algorithms, the best value was obtained before reaching to 1000th tick in the ABC algorithm. 10 was assigned to the parameter change percentage value (MR) of the ABC algorithm in the eight queens problem. This algorithm, trying to improve ABC solution sets by changing one each parameter, has yielded quite successful results in this problem. When we review other algorithms, it is observed that PSO and FA are not successful at solving this problem, contrary to huntee-hunter problem, since the structures of these algorithms are not suitable enough to solve this problem. If all queens' places are changed at the same time, achieving a solution is almost impossible in the problem of eight queens. All parameters' values (queens' positions) change in each iteration in the FA and PSO algorithm. It should be noted that as this problem is not associated with a mathematical formula, replacing overlapping queens one by one while not moving

those that do not overlap is the best method to approach the solution. Therefore, better results were obtained with ABC and GA algorithms. The fact that GA keeps good solution sets and performs crossing through these solution sets, and that ABC algorithm attempts to reach to the optimal solution through good solution sets by changing the parameters in the solution sets one by one, which enabled them to achieve quite successful results at solving this problem.

TABLE V
EIGHT-QUEENS OPTIMIZATION PROBLEM AND METAHEURISTIC ALGORITHM RESULTS

	Generation	Best Fitness Value	Tick	Duration
GA	1000	0.0625	20000.0	2577.564
PSO	1000	0.25	20000.0	3199.03
ABC	432	0.0	8610.0	478.464
FA	1000	0.125	20000.0	2935.888

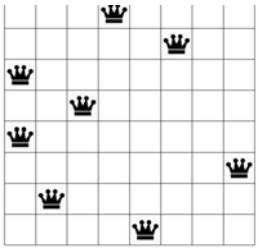

Fig. 7 The most suitable sequence as a result of 1000 iterations-GA

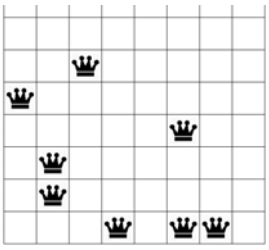

Fig. 8 The most suitable sequence as a result of 1000 iterations-PSO

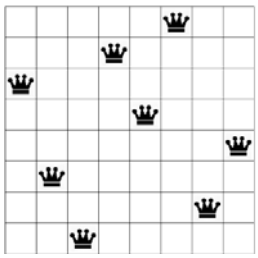

Fig. 9 The most suitable sequence as a result of 1000 iterations- ABC

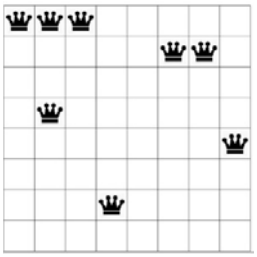

Fig. 10 The most suitable sequence as a result of 1000 iterations-FA

IV. CONCLUSION

In this study, a solution was searched for the problem of metaheuristic algorithms, with regard to tuning the parameters of complex systems modeled by agent based simulation and simulation technique. Their success on the model was tested using different metaheuristic algorithms. The different algorithms yield better results in different problems, which is one the important results of this approach developed. Besides, the other one is to gain time by taking over the burden of the determination of parameter values from model users, and giving it to the model itself. It allows for the user to make the necessary adjustments manually in a way that will satisfy the user's expectations, in the cases where the most approximate parameter value or speed comes into forefront. Since each algorithm can be run independently on the model, we can observe which algorithm yields better result in which model. Since the fitness function, which has a vital feature for each model, differs for each model and according to the model expectation of user, the fitness functions were not mentioned in this study. This study aims to offer a solution for the problem of parameter tuning of the original models created by users, rather than creating model-specific fitness functions.

In the next steps of this approach, it is aimed to develop a system that allows for algorithm parameters to adapt themselves, depending on the model or problem. Thus, the algorithms used can create better parameter sets specific to problem. Besides, the system to be developed will be tested on more models and problems.

As well as the possibility of attaining optimum solution can be increased by selecting the parameters according to the manually entered simulation duration and the maximum iterations, this approach also offer the opportunities such as attaining acceptable parameters very quickly.

REFERENCES

[1] Di Marzo Serugendo, Giovanna, Gleizes, Marie-Pierre, Karageorgos, Anthony, "Self-organising Software from Natural to Artificial Adaptation", 1st editör, Heidelberg, Berlin: Springer-Verlag, 2011, pp. 7-32.

[2] B. Calvez, G. Hutzler, "Automatic tuning of agent-based models using genetic algorithms", *Proceedings of the 6th International Workshop on Multi-Agent Based Simulation (MABS'05)*, Springer, Utrecht, The Netherland, 2005, pp. 41-57.

[3] D. S. Bolme, J. R. Beveridge, B. A. Draper, P. J. Phillips, Y. M. Lui. "Automatically Searching for Optimal Parameter Settings Using a Genetic Algorithm", *Computer Vision Systems - 8th International Conference, {ICVS}*, Sophia Antipolis, 2011, pp. 213-222.

[4] F. Dobslaw, "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, 2010.

[5] J. H. Holland, Adaptation in natural and artificial System, Ann Arbor: The University of Michigan Press, MA USA, 1992, ch 3.

[6] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", *Proc. of the IEEE Int. Conference on Neural Networks*, Western Australia, 1995, 1942-1948.

[7] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Global Optimization*, Springer US, 39, November 2007, pp. 459-471.

[8] X. S. Yang, Firefly algorithm, Nature-Inspired Metaheuristic Algorithms, 2nd Ed., Luniver Press, Ed.: Xin-She Yang, Frome UK, 2008, PP. 79-90.

[9] K. F. Man, K. S. Tang and S. Kwong, Modifications to Genetic

Algorithms, *Genetic Algorithms,* Springer London, Hong Kong, 1999, pp. 23-44.
[10] N. Adar1, G. Kuvat, "Paralel Genetik Algoritmalarda Farklılık Ve Geçirgenlik*", Dumlupınar üniversitesi, fen bilimleri ensititüsü dergisi*, vol. 27, April 2012, pp. 55-66.
[11] S. Tamer, C. Karakuzu, "Parçacık Sürüsü Optimizasyon Algoritması ve Benzetim Örnekleri", ELECO 2006 Elektrik-Elektronik-Bilgisayar Sempozyumu, Elektronik Bildirileri Kitabı, Bursa, 2006, pp. 302-306.
[12] J. C. Bansal, P. K. Singh, M. Saraswat, "Inertia Weight Strategies in Particle Swarm Optimization", *2011 Third World Congress on Nature and Biologically Inspired Computing*, Salamanca, 2011, pp. 633-640.
[13] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem", *Applied Soft Computing*, University of Hyderabad, Andhra Pradesh, vol. 9, 2009, pp. 625- 631.
[14] F. Kang, J. Li and Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Comput. Struct. 87, Dalian, 2009, 861-870 pp.
[15] U. Wilensky, NetLogo wolf sheep predation model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997.
[16] İ. Çakırlar, Çakırlar, "Etmen Temelli Benzetimler İçin Test Güdümlü Bir Yaklaşım Geliştirilmesi", Ege University, Computer Engineering Department, PhD Thesis, İzmir, 2014.