Implemantation of firefly algorithm on Arduino Uno

Denda Dewatama[1], Oktriza Melfazen[2], Mila Fauziyah[1]
[1]Department of Electrical Engineering, State Polytechnic of Malang, Malang, Indonesia
[2]Department of Electrical Engineering, Faculty of Engineering, University of Islam Malang, Malang, Indonesia

Article Info

Article history:

Received Nov 28, 2022
Revised Feb 23, 2023
Accepted Apr 24, 2023

Keywords:

Arduino Uno
Dynamic memory
Firefly algorithm
Iteration
Population

ABSTRACT

Not only getting the optimal solution of a problem, embedding the algorithm on the microcontroller is also expected to work optimally without burdening the system and fast response. Getting a microcontroller specification that matches the complexity of an algorithm is necessary so that the system can execute the algorithm perfectly. Values for the basic parameters of optimization algorithms inspired by nature such as the firefly algorithm (FFA) which are interpreted into variables greatly affect the performance of the microcontroller in obtaining the expected optimal solution. The observed performance of the Arduino Uno microcontroller in running the FFA includes execution time and memory capacity required to obtain optimal values based on changes in absorption coefficient, random parameters, iterations, and population. Changes in the absorption coefficient and random parameters affect the optimal value but do not significantly affect the execution time and memory capacity of Arduino Uno. Iteration changes greatly affect execution time and population changes most affect the performance of Arduino Uno. With a dynamic memory capacity of 2 Kb, the FFA can be run with a maximum range of 50 populations and up to 20 iterations.

Corresponding Author:

Denda Dewatama
Department of Electrical Engineering, State Polytechnic of Malang
Soekarno Hatta no. 9, Jatimulyo, Malang, East Java 65141, Indonesia
Email: denda.dewatama@polinema.ac.id

1. INTRODUCTION

Engineering problems are not always able to be solved manually or with ordinary mathematical calculations. Certain methods are needed in the form of algorithms to get the best results for solving a problem through optimization [1]–[3]. Nature-based metaheuristic algorithms by imitating animal behavior in colonies have been developed and chosen to solve global optimization problems [4]–[6] such as war strategy optimization [7], giant Trevally optimizer [8], artificial rabbits optimization [9], Bat algorithm [10], and Firefly algorithm (FFA) [11]. FFA is also one of the nature-inspired algorithms to find the optimal value and is still relatively new, introduced in 2010 [12], [13]. As compared to other algorithms, FFA has been shown in a number of studies to achieve optimal values for various cases due to the fact that it can handle multimodality and automatic subdivision. FFA has a much quicker convergent program and a straightforward process in the computer system [14], [15]. When used to solve many optimizations, especially routing optimization problems and tracking problems, firefly's method is very efficient and goes beyond traditional algorithms [16], [17].

Problems were found when using this optimization algorithm on the microcontroller. Sometimes the complexity of the formula of the algorithm in question cannot be fully supported by the specifications of the microcontroller used. Therefore when run sometimes it takes a very long time to execute or in other conditions the maximum value is not reached because the memory capacity of the microcontroller is not suitable to execute an algorithm program perfectly. In 2017, Mahmood and Jawaherry [18] tested the performance of the Arduino mega microcontroller in executing the FFA only based on changes in the number of iterations, while other basic parameters were kept the same. By only making changes to one parameter, the performance of the microcontroller has not been clearly described. Therefore, in this study, observations were made on the performance of the Arduino Uno microcontroller which was implanted with the FFA with more varied test parameters. Arduino Uno was chosen because it is one of the popular microcontrollers, in addition to programming that is easy to do, it also has a large library and adequate capacity to test various parameter conditions of the FFA.

2. METHOD

2.1. Research specification

This study focuses on the effectiveness of the performance of the FFA-implanted Arduino Uno to determine the optimal value by varying the parameter values. This study provides an appropriate parameter solution for FFA applied to the Arduino Uno. The system is tested to find the maximum value of the Sphere mathematical equation where i=1..n and x=1..10 is a random variable.

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{1}$$

Arduino Uno is an open-source microcontroller-based electronic device that uses an ATmega328P chip with surface-mount device (SMD) or dual in-line package (DIP) that functions as a controller of the electronic system [19]–[21]. To be able to use serial communication, this microcontroller is equipped with an ATmega16U2 chip. Arduino Uno has 14 digital input and output pins as well as 6 analog input pins and works at a frequency of 16 MHz, an ICSP header, a USB connection, a reset button, and 32 Kb flash memory capacity to store coding or sketch embedded in it. While SRAM is available at 2 Kb and 1 Kb EEPROM [22]. Considering its specifications, Arduino Uno was chosen to be used in this research.

As we known, FFA is a method of finding the optimal value by imitating the blinking pattern, behavior of fireflies, and bioluminescent communication phenomena [13], [23]. The inspiration for FFA comes from the behavior of fireflies, which use the light emitted from within their own bodies when they are active. A firefly with a less bright will always seek and approach the brighter one, then the position of the lighter fireflies will be represented as a solution. The fitness value represents the brightness intensity of the firefly. This repetitive activity becomes the firefly's unique behavior, inspiring FFA. In algorithm language, FFA is formulated as follows [24]: fireflies are unisexual animals, so their attraction to each other is irrespective of the gender between them. Fireflies attraction rises in direct proportion to their brightness. However, when the distance between the two fireflies goes up, the attractiveness reduces.

Several fireflies with the same brightness level will move randomly. This random movement generates a new solution. The objective function of the problem is based on the brightness level of the fireflies. The fireflies form small groups according to their attractiveness, then each subgroup swarms around local models. FFA is a very effective engineering algorithm due to its ability to resolve optimization issues even in dynamic domains. In applied mathematics, this algorithm works with simple mathematical logic [25]. Global and local optimization can be found simultaneously because FFA works based on global communication between fireflies in optimizing, especially by using real random values. Fireflies outside the subgroups work autonomously and lend themselves well to parallel implementations [15], [26], [27].

In (2) shows the light intensity of the firefly. Where the absorption coefficient is denoted by γ and the initial value is denoted by (I_0) at (r=0) [19]–[21].

$$I(\gamma) = I_0 e^{-\gamma r_{ij}} \tag{2}$$

In (3) states the attractiveness. Where the firefly attractiveness value denoted by $\beta 0$ at (r=0).

$$\beta(\gamma) = \beta_0\, e^{-\gamma r^2} \tag{3}$$

The distance between two fireflies, i and j, in locations x_i and x_j respectively is calculated in (4), where the problem's dimension is represented by D and x_{ik} is the k_{th} element of the spatial coordinate x_j of the i_{th} firefly.

$$\mathrm{r}_{ij} = |\mathrm{r}_i - \mathrm{r}_j| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2} \tag{4}$$

In (5) illustrates the transition of a firefly at location x_i to a brighter firefly at position x_j.

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}(x_i - x_j) + \alpha(rand - 0.5)$$

Where xi (t+1) is the displacement of firefly i at iteration t+1. The position of the firefly at iteration t is represented by the first term on the right side of (5). The second term is related to attraction and the final term represents randomization (blind flying in the absence of light) where α is the random parameter α ∈ [0,1]. To use the FFA in this research, it is necessary to assign certain values to several basic parameters in the FFA which will work to get the expected optimal value in question, including:

Random parameter (α)	: 0.1; 0.3; 0.6
Absorption coefficient (γ)	: 0.1; 0.3
Population (n)	: 10; 25; 50; 65
Iteration	: 3; 5; 10; 15; 20

After setting the basic parameters, the FFA is coded on the Arduino Uno following the pseudocode of FFA:

```
Initialize parameter of FA algorithm: n, α, γ, iteration
Generate initial fireflies' population xi, (i=1, 2, …, n)
Calculate intensity I(i)=f(xi)
While (t < iteration)
   For i = 1 : n all n fireflies
      For j = 1 : j all n fireflies
         Calculate the distance r between xi and xj
         If (Ij > Ii)
            Move firefly I towards j in all d dimensions
         End if
         Evaluate new solution and update light intensity
      End for j
   End for i
   Rank the fireflies and find fireflies with best intensity gives controller parameter
End while
```

Execute the program on the Arduino Uno and through the serial monitor the results can be observed on the PC. Observations were made on the best value generated by FFA (GBest), execution time, and Arduino Uno memory capacity used to produce optimal values based on the values determined for each parameter of the FFA.

2.2. Research stages

Based on several predetermined specifications, steps were developed to test the performance of Arduino Uno in optimization using this FFA, starting with initializing initial conditions for the basic parameters of the FFA (γ, α, n, iterations, and β). These parameters are started to be generated based on the initial conditions according to (2). FFA started to work on calculating the light intensity and evaluating the weight of each firefly population according to (3) and (4). The fireflies with low intensity will move closer to the lighter fireflies, while the fireflies with lighter intensity will continue to move randomly within a predetermined random space limit as in (5). When the movement of all individuals in the population has reached its best position in a certain number of iterations, then an evaluation and weighting of all individuals in the population is carried out. The individual with the highest score will be updated as GBest. The research steps are depicted in a flow chart as shown in Figure 1.

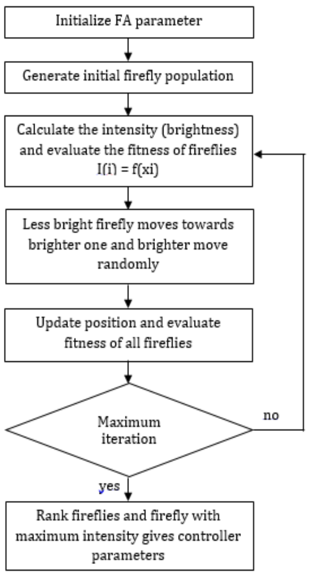

Initialize FA parameter
Generate initial firefly population
Calculate the intensity (brightness) and evaluate the fitness of fireflies I(i) = f(xi)
Less bright firefly moves towards brighter one and brighter move randomly
Update position and evaluate fitness of all fireflies
Maximum iteration
no
yes
Rank fireflies and firefly with maximum intensity gives controller parameters

Figure 1. Flowchart of firefly algorithm implementation in Arduino Uno

3. RESULTS AND DISCUSSION

The basic parameters (an absorption coefficient or γ, random parameter or α, population or n, and the first iteration) began to be generated with initial conditions as shown in Figure 2. Each firefly is randomly generated in a maximum population of 25 fireflies (shown on the x-axis) and a random value of 0 to 100 is the maximum achievable position of the fireflies (shown on the y-axis). Several tests were carried out on the system to analyze Arduino Uno's performance related to execution time and memory capacity to obtain optimal values based on the embedded FFA, as shown in Table 1. More complex observations were made compared to research conducted by Melfazen *et al.* [11] in 2017 which only tested the performance of the Arduino mega microcontroller in executing the FFA based on changes in the number of iterations only. In this study, we examine the effect of changes in firefly parameters (γ, α, n, and iterations) on the best value (GBest), the time required for processing, and dynamic memory requirements (dm). Based on the test results listed in Table 1, it will be explained one by one the effect of changing the work area value of each parameter on the optimal value that can be generated, execution time, and Arduino Uno dynamic memory capacity usage.

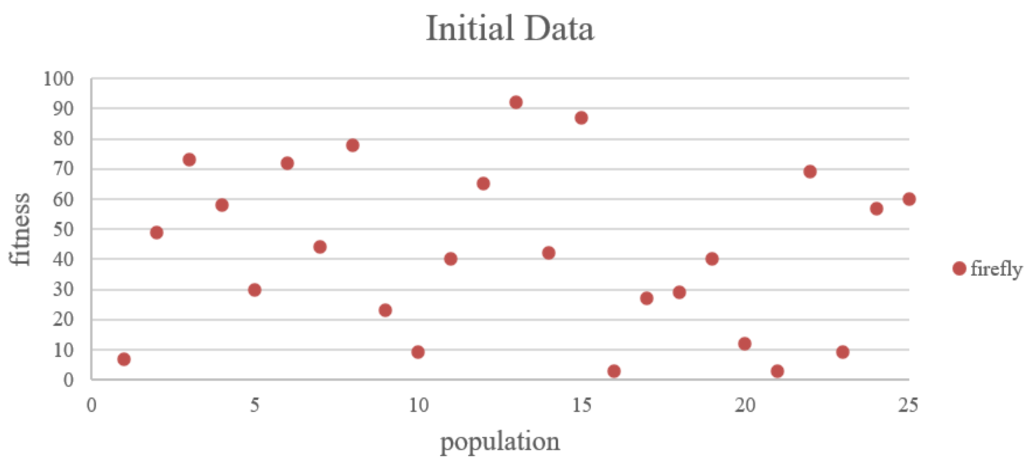


Figure 2. Generate initial firefly population

Table 1. Arduino Uno's performance in getting the best value on the execution of the firefly algorithm

			Iteration														
γ	n	α		3			5			10			15			20	
			GBest	time	dm	GBest	time	dm	GBest	time	dm	GBest	time	dm	GBest	time	dm
0.1	10	0.1	78.00	0.26	518	78.07	0.30	518	78.26	0.40	518	78.47	0.50	518	78.67	0.70	518
		0.3	78.00	0.26	518	78.22	0.30	518	78.77	0.40	518	79.40	0.50	518	80.00	0.70	518
		0.6	78.00	0.26	518	78.44	0.30	518	79.53	0.40	518	80.82	0.50	518	81.95	0.70	518
	25	0.1	92.03	4.07	938	92.06	4.26	938	92.27	4.74	938	92.48	5.22	938	92.69	5.70	938
		0.3	92.08	4.07	938	92.18	4.26	938	92.73	4.74	938	93.41	5.22	938	94.04	5.70	938
		0.6	92.16	4.07	938	92.36	4.26	938	93.46	4.74	938	94.81	5.22	938	96.15	5.70	938
	50	0.1	99.07	4.78	1638	99.16	8.11	1638	99.33	16.31	1638	99.56	24.58	1638	99.80	32.89	1638
		0.3	99.20	4.78	1638	99.47	8.11	1638	100.07	16.31	1638	100.73	24.58	1638	101.45	32.89	1638
		0.6	99.43	4.78	1638	99.95	8.11	1638	101.19	16.31	1638	102.59	24.58	1638	104.00	32.89	1638
	65	0.1	err	err	err	err	err	err	err	err	err	err	err	err	err	err	err
		0.3	err	err	err	err	err	err	err	err	err	err	err	err	err	err	err
		0.6	err	err	Err	err	err	err	err	err	err	err	err	err	err	err	err
0.3	10	0.1	78.03	0.26	518	78.11	0.30	518	78.11	0.40	518	78.48	0.50	518	78.67	0.70	518
		0.3	78.10	0.26	518	78.23	0.30	518	78.32	0.40	518	79.43	0.50	518	79.99	0.70	518
		0.6	78.21	0.26	518	78.63	0.30	518	78.63	0.40	518	80.90	0.50	518	82.05	0.70	518
	25	0.1	92.03	4.07	938	92.06	4.26	938	92.06	4.74	938	92.45	5.22	938	92.68	5.70	938
		0.3	92.08	4.07	938	92.18	4.26	938	92.18	4.74	938	93.56	5.22	938	94.24	5.70	938
		0.6	92.16	4.07	938	92.36	4.26	938	92.36	4.74	938	94.88	5.22	938	96.20	5.70	938
	50	0.1	99.07	4.78	1638	99.11	8.11	1638	99.11	16.31	1638	99.11	24.58	1638	99.79	32.89	1638
		0.3	99.22	4.78	1638	99.33	8.11	1638	99.33	16.31	1638	100.66	24.58	1638	101.32	32.89	1638
		0.6	99.43	4.78	1638	99.66	8.11	1638	99.66	16.31	1638	102.15	24.58	1638	103.49	32.89	1638
	65	0.1	err	err	err	err	err	err	err	err	err	err	err	err	err	err	err
		0.3	err	err	err	err	err	err	err	err	err	err	err	err	err	err	err
		0.6	err	err	err	err	err	err	err	err	err	err	err	err	err	err	err

3.1. The effect of changes in absorption coefficient (γ) and random parameter (α) on GBest, execution time, and dynamic memory

The amount of light that is absorbed (γ) determines how much light is lost. The γ parameter influences how quickly the FFA converges and explains the fluctuation in attraction. Testing on the γ parameter by changing the value to it shows that there is only a small effect on changes in the resulting GBest value and has no effect on execution time and dynamic memory requirements.

The random parameter is one of the variables that affect the calculation of the level of interest between fireflies. It was determined that this random parameter had a value from 0 and 1. In this study, the random step size was determined at 0.1, 0.3, and 0.6. Testing on changes in random parameter values and explains that the use of larger random parameter values will produce better GBest values. However random parameters do not affect execution time and dynamic memory requirements.

3.2. The effect of iteration change on GBest, execution time, and dynamic memory

The best fireflies have the highest light intensity in each iteration. It would be a potential solution. Based on the test results in Table 1, iteration changes have a beneficial effect on the resulting GBest value. The more iterations used, the better the GBest value will be. Conversely, a large number of iterations is also detrimental because the execution process takes longer to find the best value. Iteration changes do not affect Arduino Uno's dynamic memory usage. Figure 3 shows the fitness of the firefly obtained for each given iteration (3rd, 5th, 10th, 15th, and 20th iterations). Proper iteration provides a better chance for each individual firefly to reach the most optimal value at the predetermined maximum position.

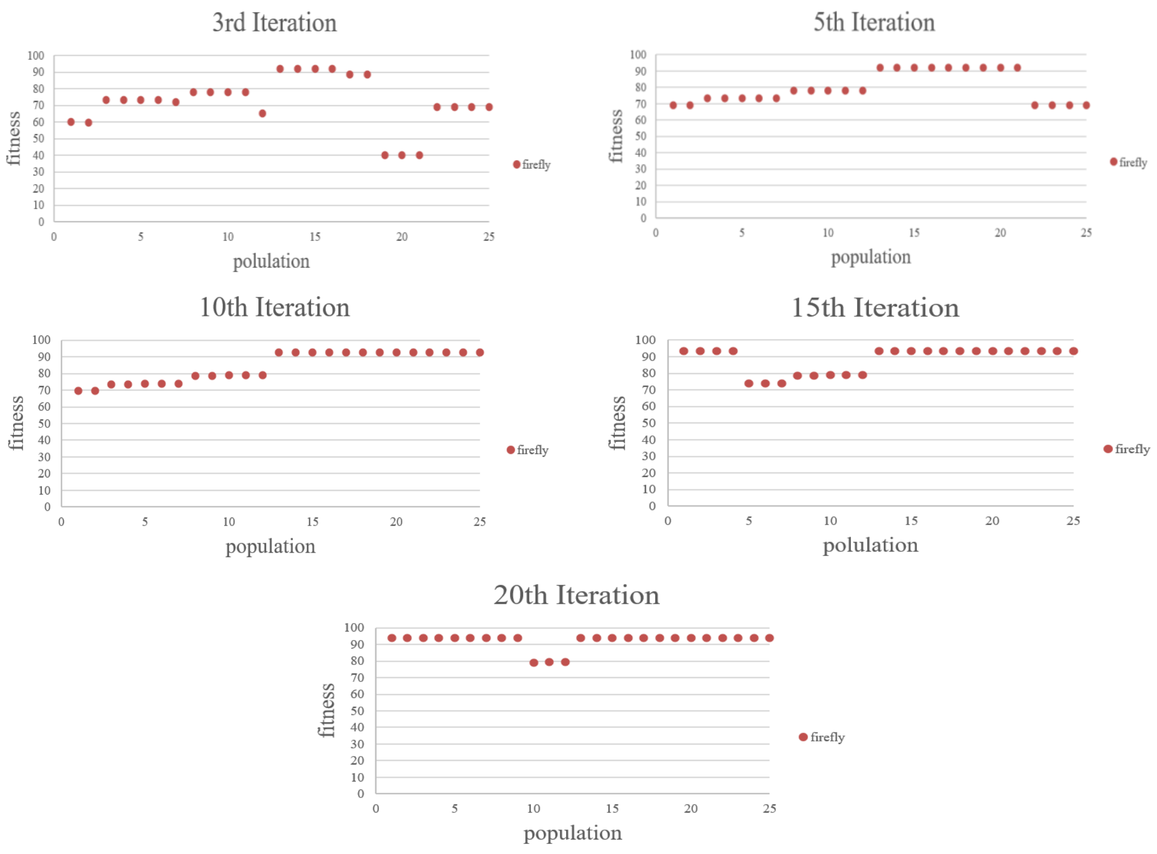
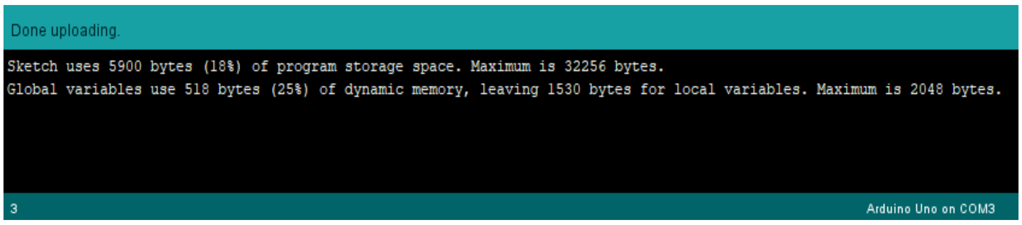


Figure 3. The effect of iteration change on GBest

3.3. The effect of population change on GBest, execution time, and dynamic memory

The population in the FFA is the total number of fireflies involved, which consists of a set of codes that represent the solution to the problem. In this study, changes were made to the number of population variables (10; 25; 50; and 65), based on Table 1. It can be seen that changes in population values have a significant effect on the resulting GBest value, the required execution time, and the amount of dynamic memory needed.

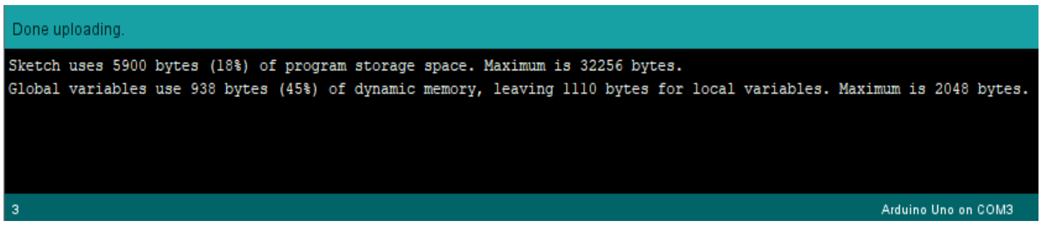
The first experiment was given an absorption value (γ) of 0.1, a population (n) of 10, and a variable value of 0.1; 0.3; 0.6. Observations were made on the optimal value, execution time, and dynamic memory of Arduino Uno used. The optimal value (GBest) is 78.00. The best value generated in this condition is only 78% of the fitness set. The longest execution time in this condition is 0.7 seconds. Arduino Uno's dynamic memory used is quite small, around 518 bytes, and only 25% of the total available dynamic memory is used. The results of program execution can be seen in Figure 4.



```
Done uploading.
Sketch uses 5900 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 518 bytes (25%) of dynamic memory, leaving 1530 bytes for local variables. Maximum is 2048 bytes.
3                                                                                           Arduino Uno on COM3
```

Figure 4. Population (n)=10

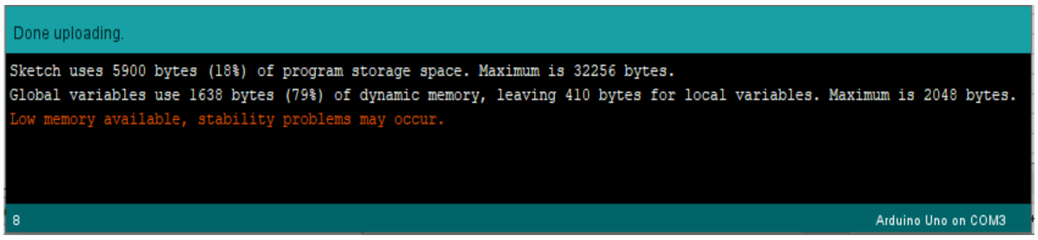
In the next experiment, an absorption coefficient of 0.1 was given. With random parameter values varying from 0.1; to 0.3; 0.6, the firefly population was enlarged to 25. It turns out that the increase in the firefly population increases the optimal value (GBest) that can be produced at the maximum achievable position of the fireflies that has been determined. Arduino Uno requires a longer execution time than if fireflies have a smaller population as shown in Table 1, it takes almost five times more execution time. Increase the memory capacity used up to 45% of the maximum available memory, as shown in Figure 5.



```
Done uploading.
Sketch uses 5900 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 938 bytes (45%) of dynamic memory, leaving 1110 bytes for local variables. Maximum is 2048 bytes.
3                                                                                           Arduino Uno on COM3
```

Figure 5. Population (n)=25

The next test was carried out by making the population twice as large as the second experiment (50) at an absorption coefficient of 0.1 and the random parameters varied by 0.1; 0.3; and 0.6 then the most optimal value (GBest) can be obtained close to the maximum position that the fireflies can reach that has been determined with a fairly short execution time of only 4.78 seconds. It can be seen that the execution time is not much different from the previous population value even though the dynamic memory capacity used is increased to 79% of the maximum available memory or about 1,638 bytes. As seen in Figure 6, Arduino Uno gives a warning that memory usage above 75% can cause stability problems, there may be corrupted data or data that is not executed.

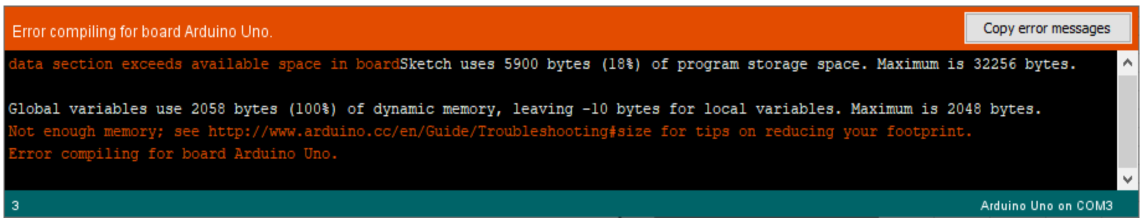


```
Done uploading.
Sketch uses 5900 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 1638 bytes (79%) of dynamic memory, leaving 410 bytes for local variables. Maximum is 2048 bytes.
Low memory available, stability problems may occur.
8                                                                                           Arduino Uno on COM3
```

Figure 6. Population (n)=50

When the program is executed by giving a population greater than 65 and an absorption coefficient of 0.1 and random parameters varying at 0.1; 0.3; and 0.6 then the words "error compiling for Arduino Uno board" appear as shown in Figure 7. In this condition, Arduino Uno can no longer work to find the optimal value of the FFA. The memory required is 2,058 bytes, exceeding the available capacity of 2,048 bytes.

It means the increase in population has an effect on increasing the amount of memory capacity usage because more variables need to be stored as the implementation of each individual firefly. It is necessary to attend to the warning given by Arduino Uno in determining the maximum population limit related to the use of dynamic memory capacity that has been used. Understandably from all tests, population changes are very influential not only to get the optimal value (GBest) but also on execution time and the most significant effect of this population change is on memory usage. Giving a large population causes a lot of memory to be used for data storage and reduces the ability of the Arduino Uno microcontroller to execute commands to search for optimal values.



```
Error compiling for board Arduino Uno.
data section exceeds available space in boardSketch uses 5900 bytes (18%) of program storage space. Maximum is 32256 bytes.

Global variables use 2058 bytes (100%) of dynamic memory, leaving -10 bytes for local variables. Maximum is 2048 bytes.
Not enough memory; see http://www.arduino.cc/en/Guide/Troubleshooting#size for tips on reducing your footprint.
Error compiling for board Arduino Uno.

3                                                                                            Arduino Uno on COM3
```

Figure 7. Population (n)=65

4. CONCLUSION

By observing the results of Arduino Uno performance testing in the problem-solving process using the FFA, it can be concluded that changes in absorption coefficient and random parameters are greatly affect in achieving the most optimal values but changes in absorption coefficient and random parameters have no significant effect on Arduino Uno's execution time and memory capacity. Iteration changes greatly affect the time needed to execute the program. Of all the parameters in the FFA, the population is the most influential on the performance of the Arduino Uno. The 2 Kb of dynamic memory on the Arduino Uno is capable of handling programming for the FFA with a maximum population range of 50 and up to 20 iterations. Setting a large population of fireflies can improve the ability of the algorithm to get the optimal value to the maximum position that can be reached by the fireflies, but it impacts on the amount of memory capacity used because more variables need to be stored as the whereabouts of each individual firefly.

To achieve the optimal value in solving a problem using the FFA in a microcontroller, it is necessary to estimate memory usage correctly. Thus, making it more flexible in determining the size of the population. Using a microcontroller with a higher working frequency will shorten the execution time.

REFERENCES

[1] J. O. Agushaka and A. E. Ezugwu, "Advanced arithmetic optimization algorithm for solving mechanical engineering design problems," *PLoS ONE*, vol. 16, no. 8, pp. 1–29, 2021, doi: 10.1371/journal.pone.0255703.

[2] B. Chen, H. Chen, and M. Li, "Improvement and optimization of feature selection algorithm in swarm intelligence algorithm based on complexity," *Complexity*, vol. 2021, pp. 1–10, 2021, doi: 10.1155/2021/9985185.

[3] S. Taheri, M. Mammadov, and S. Seifollahi, "Globally convergent algorithms for solving unconstrained optimization problems," *Optimization: A Journal of Mathematical Programming and Operations Research*, vol. 64, no. 2, pp. 249–263, 2012, doi: 10.1080/02331934.2012.745529.

[4] B. J. Saharia, H. Brahma, and N. Sarmah, "A review of algorithms for control and optimization for energy management of hybrid renewable energy systems," *Journal of Renewable and Sustainable Energy*, vol. 10, pp. 1–33, Sep. 2018, doi: 10.1063/1.5032146.

[5] K. C. Tan, L. Feng, and M. Jiang, "Evolutionary transfer optimization-a new frontier in evolutionary computation research," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 22–33, 2021, doi: 10.1109/MCI.2020.3039066.

[6] M. J. -Tehrani, O. B. -Haddad, and H. A. Loáiciga, "A review of applications of animal-inspired evolutionary algorithms in reservoir operation modelling," *Water and Environment Journal*, vol. 35, no. 2, pp. 628–646, 2021, doi: 10.1111/wej.12657.

[7] T. S. L. V. Ayyarao *et al.*, "War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization," *IEEE Access*, vol. 10, pp. 25073–25105, 2022, doi: 10.1109/ACCESS.2022.3153493.

[8] H. T. Sadeeq and A. M. Abdulazeez, "Giant trevally optimizer (GTO): a novel metaheuristic algorithm for global optimization and challenging engineering problems," *IEEE Access*, vol. 10, pp. 121615–121640, 2022, doi: 10.1109/ACCESS.2022.3223388.

[9] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: a new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105082, 2022, doi: 10.1016/j.engappai.2022.105082.

[10] J. Liu, H. Ji, Q. Liu, and Y. Li, "A bat optimization algorithm with moderate orientation and perturbation of trend," *Journal of Algorithms and Computational Technology*, vol. 15, pp. 1–11, 2021, doi: 10.1177/17483026211008410.

[11] O. Melfazen, M. T. Alawity, and D. Dewatama, "Firefly algorithm for optimizing single axis solar tracker," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, vol. 6, no. 4, pp. 313–320, 2021, doi: 10.22219/kinetik.v6i4.1338.

[12] X. S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Engineering with Computers*, vol. 29, no. 2, pp. 175–184, 2013, doi: 10.1007/s00366-012-0254-1.

[13] X. S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36–50, 2013, doi: 10.1504/ijsi.2013.055801.

[14] N. J. Cheung, X. M. Ding, and H. B. Shen, "A non-homogeneous firefly algorithm and its convergence analysis," *Journal of Optimization Theory and Applications*, vol. 170, no. 2, pp. 616–628, 2016, doi: 10.1007/s10957-016-0875-4.

[15] N. F. Johari, A. M. Zain, N. H. Mustaffa, and A. Udin, "Firefly algorithm for optimization problem," *Applied Mechanics and Materials*, vol. 421, pp. 512–517, 2013, doi: 10.4028/www.scientific.net/AMM.421.512.

[16] D. Dewatama, Siswoko, H. K. Safitri, and O. Melfazen, "MPPT using firefly algorithm for Cuk converter in photovoltaic," *ELKHA: Jurnal Teknik Elektro*, vol. 14, no. 1, pp. 34–39, 2022, doi: 10.26418/elkha.v14i1.52461.

[17] M. L. Gao, X. H. He, D. S. Luo, J. Jiang, and Q. Z. Teng, "Object tracking using firefly algorithm," *IET Computer Vision*, vol. 7, no. 4, pp. 227–237, 2013, doi: 10.1049/iet-cvi.2012.0207.

[18] R. Z. Mahmood and M. A. A. -Jawaherry, "Firefly algorithm implementation based on Arduino microcontroller," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 15, no. 12, pp. 1–5, 2017.

[19] B. Setiawan, E. S. Putra, I. Siradjuddin, M. Junus, D. Dewatama, and S. Wiyanto, "Study of LoRa (long range) communication for monitoring of a ship electrical system," *Journal of Physics: Conference Series*, vol. 1402, no. 4, pp. 1–6, 2019, doi: 10.1088/1742-6596/1402/4/044022.

[20] K. K. Khaing, K. S. Raju, G. R. Sinha, and W. Y. Swe, "Automatic temperature control system using Arduino," in *Proceedings of the Third International Conference on Computational Intelligence and Informatics*, 2020, pp. 219–226, doi: 10.1007/978-981-15-1480-7_18.

[21] Y. A. Badamasi, "The working principle of an Arduino," in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, 2014, pp. 1–4, doi: 10.1109/ICECCO.2014.6997578.

[22] R. H. Sudhan, M. G. Kumar, A. U. Prakash, S. A. R. Devi, and P. Sathiya, "Arduino ATMEGA-328 microcontroller," *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, vol. 3, no. 4, pp. 27–29, 2015, doi: 10.17148/ijireeice.2015.3406.

[23] Y. Zhang, L. Wu, and S. Wang, "Solving two-dimensional HP model by firefly algorithm and simplified energy function," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–10, 2013, doi: 10.1155/2013/398141.

[24] S. Arora and S. Singh, "The firefly optimization algorithm: convergence analysis and parameter selection," *International Journal of Computer Applications*, vol. 69, no. 3, pp. 48–52, 2013, doi: 10.5120/11826-7528.

[25] L. Zhang, L. Liu, X. S. Yang, and Y. Dai, "A novel hybrid firefly algorithm for global optimization," *PLoS ONE*, vol. 11, no. 9, pp. 1–17, 2016, doi: 10.1371/journal.pone.0163230.

[26] S. Yu, S. Su, Q. Lu, and L. Huang, "A novel wise step strategy for firefly algorithm," *International Journal of Computer Mathematics*, vol. 91, no. 12, pp. 2507–2513, 2014, doi: 10.1080/00207160.2014.907405.

[27] M. K. A. Ariyaratne and T. G. I. Fernando, "A comparative study on nature inspired algorithms with firefly algorithm," *International Journal of Engineering and Technology*, vol. 4, no. 10, pp. 611–617, 2014.

BIOGRAPHIES OF AUTHORS

Denda Dewatama received the S.T and M.T. degrees in Department of Electrical Engineering from the University of Brawijaya Malang, in 2004 and 2009. From 1999-2001, he worked at PT Hartono Istana Engineering as Design Quality Assurance. From 2005, he has taught in Department of Electrical Engineering at the State Polytechnic of Malang. He has written or co-written more than five research studies and one book. His areas of interest in research include renewable energy, artificial intelligence, control systems, and DC-DC converters. He can be contacted at email: denda.dewatama@polinema.ac.id.

Oktriza Melfazen received bachelor of engineering and master of engineering degrees in Department of Electrical Engineering from the University of Brawijaya Malang, in 2005 and 2012. She has worked as a lecturer in Department of Electrical Engineering at University of Islam Malang, since 2013. The scope of her research is the application of automatic control and renewable energy involving artificial intelligence. She has written one book. She can be contacted at email: oktriza.melfazen@unisma.ac.id.

Mila Fauziyah received her S.T. degree at Department of Electrical Engineering, University of Brawijaya Malang in 2000 and received her M.T. degree in Department of Electrical Engineering from the University of Gadjah Mada in 2007. Since 2000, she has taught in Department of Electrical Engineering at the State Polytechnic of Malang. She serves as a lecturer, academic adviser, and P3AI secretary at the Indonesian State Polytechnic in Malang. She has written or co-written for over 7 publications. Applications of NN, BP-NN, PID control, fuzzy logic, and image processing are among her areas of interest in the study. She can be contacted at email: mila.fauziyah@polinema.ac.id.