A modified artificial bee colony for N-Queens problem

De Long Sia
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
TP060810@mail.apu.edu.my

Kong Zee Xin Emerson
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
TP061400@mail.apu.edu.my

Sheng Jian Lim
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
TP056549@mail.apu.edu.my

Jin Han Ling
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
TP059609@mail.apu.edu.my

Zailan Arabee bin Abdul Salam
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
zailan@apu.edu.my

***Abstract*— This paper is to solve the N-Queen problems with the implementation of the Artificial Bee Colony Algorithm. A modified Artificial Bee Colony algorithm is introduced in this paper to improve the efficiency and the effectiveness on solving 64-Queens problem. In this paper, a few parameters of the algorithm are changed and compared to each other during solving of the N-Queens problem that leads to a better result. Parameters changed including colony population, trial limit and shuffle range which directly affect the algorithm; hence a best parameters combination is found for a 64-Queens problem with improvement.**

Keywords—Artificial Bee Colony, ABC, N-Queens, Parameter, Complexity, Time taken, Success rate

I. Introduction

The N-Queen problem is a NP-Complete problem. It is to place the N number of queens on a chess board that is N x N size big. When the N number of queens are placed on the N x N size chess board, they should not be a threat to each other. Since the queens on the chess board could move either two horizontal ways, two vertical ways, or two diagonal ways and hence lead to finding the place for NP-complete problem sets[1]. The main task is to find a way for the Artificial Bee Colony (ABC) algorithm to work in a more efficient way in solving the N-Queen problems when the number of queens increases on the chessboard.

This paper will consist of 4 sections. This is the first section of the paper which includes a literature review of the topic. Section II is Methods and Material, it is about the ways to obtain the required materials and algorithm used. Section III is about the discussion of implementation and explanation for the results. Section IV is about the conclusion for this paper and possible future enhancements.

A. Literature Review on Topic

In this paper [2], a comparison is made between using the ABC algorithm and also the Differential equation to solve the N-Queen problem. The enhanced convergence speed of basic is also implemented in the Differential Equation without any extra supporting functions. Multiple different metrics are used for comparison such as, Number of function calls, SP-Success performance conjugation of NFC and SR metrics, the time taken for the run to be completed on a single N-value and the SR-success Rate-a metric derivative of NFC. The results in this paper shows that the modified Differential Equation has outlined the basic DE. But the ABC algorithm solves the n-queen problem in a faster and more efficient way. By using the ABC algorithm, problems that are at the higher-dimensional level had been solved with a somewhat lesser number of epochs and also greater success rate.

II. Material and Methods

This section shows the details of how to obtain the materials and the definition for methods used for solving the problem.

A. Sources of Material

First of all, the source code is from an open source cloud storage GitHub[3], the source code is downloaded from there in java format. Besides that, an IDE NetBeans 8.2 version is used for easier reading and executing the source code. The algorithm for solving the N-Queens problem is using ABC as mentioned.

B. Algorithm Implementation

In the case of this algorithm study, the ABC concept is used to solve the N-Queens problem. The ABC algorithm is defined as a meta-heuristic which is also one of the most recently introduced swarm-based algorithms where it will simulate the intelligent foraging behaviour of the honey bees where the honey bees will be based on the resources space discovered and search for quality food sources.

This algorithm concept is used to solve numerical problems where the ABC algorithm has successfully applied into various practical problems to produce an optimal solution of the particular problem. In this ABC algorithm, every solution that is in the search space will be having a set of optimization parameters which is to represent the food source position. In this case, the number of food sources will be equal to the number of employed bees[4].

The algorithm uses a set of computational agents as the algorithm materials which are called “honeybees” to find the optimal solution. In this ABC algorithm, the honey bees will be categorized into three types: employed bees, onlooker bees, and scout bees. Each type of the bees will be having different tasks and objectives in order to combine every result to obtain the final optimal solution. The process of bee advertising, swarm seeking, and eventually selecting the best known food source will be the process that is used to search for the optimal solution. The optimization part of the algorithm will improve the current bee by choosing a random neighbour bee. The changes are a randomly generated number of times to try and improve the current solution[5].

For describing the task for every type of bees, the employed bees will have the objective to investigate the food sources that are found, known as the fitness values and sharing the information with the onlooker bees. The onlooker bees will be making a decision to choose a food source based on the value of the food source. This means that the food source that is having a higher quality will be having a larger probability of being selected by the onlooker bees. The scout bees will be finding new random food positions. If they found a quality food source, it is called the "fitness value" and also it is associated with its position.

For describing the usage of the parameters, N size is the parameter of determining the complexity of the N-Queens problem as it increases the board size and number of queens. It is to be compared and relating to the parameters of the ABC algorithm. The recommended N size is 4, 8, 12, 16 and 40. Colony population is the total population of every type of bee in this current colony and it has a direct relation with the number of food sources since it is defined as the half of the colony population. A population of 20 is set to default where it may be well suited for low complexity problems but not for high complexity. Trial limit is the available time for worker bees and onlooker bees to improve their carrying solution for the problem. The limit is predefined and is set to 50 on default for the use of escape from the local optima solution where the scout bees will reinitialise the solution when the trial limit is reached for that particular solution. There is also a predefined shuffle time range before starting to find the solution to the problem. It is used for limiting the shuffle time for scout bees when a bee could not improve their carrying solution within a trial limit given. This is achieved by getting a random number in the shuffle range, then the scout bee will randomly swap the numbers of positions with another on the board, so the process may repeat on the same position. A recommended range for shuffle is given from 8 to 20 in the source code but since the experiment will be using a larger N for problems out of the range of recommendation.

III. Result and Discussion

This section shows the hardware used on the implementation and modification on parameters of the algorithm made along with the explanation for the results.

A. *Discussion on Implementation*

The implementation is done on a device with Windows 10 operating system using Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz as central processing unit, 16GB ram installed and GeForce Nvidia RTX2060 as the graphic processing unit. One single execution will expect for a 50 time success finding a 0 conflict solution within 1000 epoch each time. If the maximum epoch reaches but the solution still consists of more than 0 conflict then that attempt will be considered as a failure.

The main focus of N size chosen for the problem is 64 as it is having an intermediate complexity and improvement can be easily noticed after modification of parameters. To avoid other elements from affecting the result, each shuffle range is tested individually by taking the average result from 10 times execution while colony population and trial limit are tested together, then a best configuration will be given for the 64-Queens problem. Therefore, all recommended N size is involved in comparison while 64 and 128 will both act as intermediate complexity and high complexity respectively while colony population and trial limit testing only carry out for 64 N size.

For colony population and trial limit parameters. The swarm size that was being tested includes 10, 50, 100, 200, 300, 500, and 1000 while the limit comprises 10, 50, 100, 250, and 500. The ratio between the colony population and number of food sources is remained the same due to many of the other studies configured it as number of food sources is the half of the colony population

For shuffle range parameters, an expectation is made that the best shuffle range will be around the N size for N-Queens problem while minimum should be at least 1 time shuffle, therefore range width is chosen to be 10 for every range class from 1 to 80. The reason to have 80 is to prove that shuffle time exceeding N size is no longer beneficial for the algorithm.

B. *Result*

TABLE I. Time Taken For Shuffle Range

Shuffle Range\N	4	8	12	16	40	64	128
1-10	0sec	0sec	0sec	0sec	11.9sec	94.5sec	312.1sec
11-20	0sec	0sec	0sec	0sec	10.1sec	79.7sec	305.7sec
21-30	0sec	0sec	0sec	0sec	9.4sec	69.3sec	292.8sec
31-40	0sec	0sec	0sec	0sec	8.9sec	67sec	291.5sec
41-50	0sec	0sec	0sec	0sec	8.6sec	62.6sec	290.9sec
51-60	0sec	0sec	0sec	0sec	8.7sec	60sec	295.8sec
61-70	0sec	0sec	0sec	0sec	9.1sec	57.2sec	298.1sec
71-80	0sec	0sec	0sec	0sec	9sec	63.6sec	304.2sec

Table I shows that shuffle range will not be very significant affecting the quality of the result when the N size is small since the board is too small and does not require a large amount of shuffle. If the complexity reaches a point that shuffle is matter, then the higher the shuffle range is, the lower the time taken for the algorithm to solve the problem but that is not the case for too high complexity problem such as 128 N size for this case. However, it also can conclude that if the shuffle range is too large than the N size, the time taken will be higher from 64 N size.

TABLE II. Success Rate For Shuffle Range

Shuffle Range\N	4	8	12	16	40	64	128
1-10	100%	100%	100%	100%	99.03%	43.56%	0%
11-20	100%	100%	100%	100%	99.22%	51.20%	0%
21-30	100%	100%	100%	100%	99.61%	56.81%	0%
31-40	100%	100%	100%	100%	99.61%	61.25%	0%
41-50	100%	100%	100%	100%	99.61%	64.69%	0%

51-60	100%	100%	100%	100%	99.80%	64.92%	0%
61-70	100%	100%	100%	100%	99.22%	67.41%	0%
71-80	100%	100%	100%	100%	99.61%	62.66%	0%

Table II shows that problems with low complexity where N size is small do not affect by the shuffle range significantly. However, problems with too high complexity, modification on shuffle range also could not make the algorithm successful to find a solution for the problem such as 128 as the N size. For a problem which is solvable with current parameters, the higher the shuffle range is, the higher the success rate for the ABC algorithm to solve the problem. However, it also can conclude that if the shuffle range is too large than the N size, the success rate will be lower.

TABLE III. RESULT FOR COLONY POPULATION AND TRIAL LIMIT

Swarm Size	Limit	Average runtime (sec)	Number of failures
10	10	103.8	100
	50	144.8	81.1
	100	131.8	76.6
	250	131.5	77.6
	500	130.6	76.3
50	10	556.5	100
	50	209.2	5.5
	100	206.5	3.5
	250	194.6	1.8
	500	192.7	2.5
100	10	1006.2	100
	50	326.3	0.9
	100	310.3	0.2
	250	310.4	0.1
	500	326.8	0.1
200	10	1985.3	100
	50	584.8	0
	100	541.2	0
	250	552.3	0
	500	535	0
300	10	926.5	100
	50	246.2	0
	100	239.5	0
	250	234	0
	500	246.3	0
500	10	1508.9	100
	50	381.7	0
	100	430.5	0
	250	366.2	0
	500	366.4	0
1000	10	3060.1	100
	50	685.7	0
	100	656.8	0
	250	654.5	0
	500	656.9	0

From Table III, it can be concluded that an increase in colony population can increase performance and decrease as the number of failures decreases. When the colony population increases to 200, the number of failures is zero, except for an extreme limit, which is 10. This shows that all foraging for one process of different food sources is successful.

Out of the range of N-queen solutions in epoch. However, in the process of increasing the colony population, in addition to the colony population, the time taken has also increased significantly by 300, where the time taken is similar to the average. The time taken of the colony population is 10 and 50, but it has a better correlation with them and the result failed. Therefore, a colony population of 300 was determined to be the best to set the colony population in this question. Once the required colony population is reached, that is, 200. In this case, the increase in colony population will not make any improvement. Therefore, the optimal setting must be known to avoid setting a huge value for the cluster size. This does not help to improve, but only reduces performance.

The results show that a set of optimal parameters can be determined to solve the highest N-Queens problem efficiency and performance, which is when the colony population is set to 300 and the limit is set to 250. The trial limit has a significant impact on the size of all clusters when the value exceeds a certain value. This study is not without its limitations. Although the optimal group size and limiting parameters are determined to be the problem after 64, there are other parameters that can be studied to further reconstruct the performance of ABC on this problem.

In the end, by concluding the finding from every parameter, the most suitable parameter configurations for 64-Queens problem are given as shown on table IV while others remained the same as default from the source. Colony population is set to 300 while the limit is 250 because it is showing the best performance among others. Shuffle range is chosen to be 61-69 as it is around the N size of the problem.

TABLE IV. PARAMETER CHANGES

Parameters	Default ABC	Modified ABC
Colony Population	20	300
Trial Limit	50	250
Shuffle Range	8-20	61-69

Both of default ABC and modified ABC will be tested as the same way of parameter testing. The result is shown as table V at below.

TABLE V. COMPARISON OF ABC

Result	Default ABC	Modified ABC	Improvement
Time Taken	86.8sec	304.7sec	-251.04%
Success Rate	47.32%	100%	111.33%

From Table V, it can be concluded that the success rate of finding the global optima solution is increased, while the time taken is also increased. This is because the greedy selection from the parameter testing where only the best performance of parameter chosen. Therefore, further research can be carried out in the future to get a deep understanding between the parameter and the performance to balance the time taken and success rate. However, the success rate is benefit significantly from the greedy selection where it reaches the maximum success rate which mean the algorithm will always success to find a global optima solution.

IV. CONCLUSION

In a nutshell, this paper proposed a modified ABC algorithm for the 64-Queens problem. The modification of parameters to the ABC algorithm is impactful for the solution to get out of local optima and increase the chance of getting global optima. The modified ABC enhanced the success rate to be better than the default ABC but required a higher time taken, which was tested in a specified way. Although it required a higher time taken, the result showed that the improvement of success rate is significant and without failure. Therefore, further improvement can be made to reduce the time taken with the same level of success rate.

REFERENCES

[1] CodesDope. "N-Queens Problem." Backtracking. https://www.codesdope.com/course/algorithms-backtracking/ [accessed Aug. 20, 2021].

[2] P. N. Sharief & B. S. Saini. "Metaheuristic Techniques On N-Queen Problem: DE Vs ABC". International Journal of Applied Engineering Research, X(55), pp. 4240-4244, 2015. ResearchGate. https://www.researchgate.net/publication/283092799_Metaheuristic_techniques_on_N-Queen_problem_DE_VS_ABC. [accessed Aug. 23, 2021]

[3] jimsquirt. "Java code implementing the Artificial Bee Colony (ABC) algorithm in solving the N-Queens problem." GitHub. https://github.com/jimsquirt/JAVA-ABC [accessed Aug. 30, 2021].

[4] P. Melin et al. "Foundations of Fuzzy Logic and Soft Computing." LNAI 4529 ed. Mexico: 12th International Fuzzy Systems Association World Congress, IFSA 2007, Cancun, Mexico, June 18-21, 2007, Proceedings. Google Book. https://books.google.com.my/books?hl=en&lr=&id=daypg0c1t00C&oi=fnd&pg=PR4&dq=Foundations+of+Fuzzy+Logic+and+Soft+Computing&ots=-DW3XHKb9G&sig=OzNUKfS3iqXq9ZQYZSS2AW1Uass#v=onepage&q=Foundations%20of%20Fuzzy%20Logic%20and%20Soft%20Computing&f=false [accessed Aug. 26, 2021].

[5] X.S. Yang. "Nature-Inspired Computation and Swarm Intelligence: Algorithms, Theory and Applications". Cambridge: Academic Press. ScienceDirect. https://www.sciencedirect.com/book/9780128197141/nature-inspired-computation-and-swarm-intelligence [accessed Aug. 22, 2021].