Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 6, December 2021, pp. 3333~3340
ISSN: 2302-9285, DOI: 10.11591/eei.v10i6.3206

Timetabling problem solving based on best-nests cuckoo search

Mohammed A. Jebur, Hasanen S. Abdullah
Computer Science Department, University of Technology, Baghdad, Iraq

Article Info

Article history:

Received Jul 6, 2021
Revised Sep 11, 2021
Accepted Oct 16, 2021

Keywords:

Improved cuckoo search
Levy flight
Timetabling problem

ABSTRACT

The university courses timetabling problem (UCTP) is a popular subject among institutions and academics because occurs every academic year. In general, UCTP is the distribution of events through slots time for each room based on the list of constraints for instance (hard constraint and soft constraint) supplied in one semester, intending to avoid conflicts in such assignments. Under no circumstances should hard constraints be broken while attempting to fulfill as many soft constraints as feasible. this article presented a modified best-nests cuckoo search (BNCS) algorithm depend on the base cuckoo search (CS) algorithm. BNSC algorithm was achieved by dividing the nests into two groups (best-nests and normal-nests). The BNCS algorithm selection was limited to the best-nests to generate new solutions. The comparison between BNCS and basic CS based on the experimental result is achieved. For performance evaluation, the BNCS has been tested on four variant-size datasets. It was observed that the BNCS has performed high performance and is faster at finding a solution from CS.

Corresponding Author:

Mohammed A. Jebur
Computer Science Department
University of Technology
Baghdad, Iraq
Email: cs.19.10@grad.uotechnology.edu.iq

1. INTRODUCTION

Timetabling is a control that has always been in existence. This problem can be found in a variety of fields, including education, sports, and transportation [1]. The educational timetabling problem, which is challenging for educational institutions, involves a group of events for instance courses that should be allocated to specific rooms and slots-time while adhering to a group of constraints [2]. Constraints are divided into hard constraints and soft constraints. Hard constraints must not be broken for a solution to be regarded as practical, whereas soft constraints just increase the quality of the solution but are not required to be satisfied [3]. It is a Nondeterministic Polynomial Hard (NP-Hard) task to solve the university course timetable problem (UCTP). Several strategies may be used to solve this problem, but no promise can be made that the best answer will be found [4]. As a result, solving this problem is extremely difficult, and it becomes increasingly more difficult as the issue size grows. In the last several years, researchers use different timetabling methods using population-based methods, constraint-based methods (ant colony optimization [5], genetic algorithms [6], [7] and memetic algorithms [8]), the metaheuristic methods (simulated annealing [9], tabu search [10], and great deluge [11]), variable neighborhood search, and hybrid and hyper-hybrid approaches, etc. As a result of this diversity in the use of algorithms, further consideration should be given to the UCTP study. UCTP is an issue that involves fitting a set of events for instance teaching, into a predefined (room and time) bundle while meeting all of the constraints of the problem. A bundle (b) of room (r) and time (t) is formulated in (1).

$$b = (r, t) \quad (1)$$

First, the UCTP resource is room, where that differ in (ownership, location, capacity, and specialization) for example the capacity size for (lecture rooms, lesson rooms, and lab rooms). The second resource is time, it may differ from one university to the next, and it may vary on every week, every day, every hour, or even every minute basis. For instance, a university may apply a (45) limit minute, while another university may apply a (90) limit minute. A lecturer (l) teaching a certain class (c) to a group of students (s) is defined as an event (e) in (2).

$$e = (l, c, s) \quad (2)$$

As a result, a timetable is a collection of all events that are divided into several or all bundles. A join (j) is a connection between a bundle and an event, as defined by (3).

$$j = (b, e) \quad (3)$$

According to previous research [3], this study only used one type of constraint: the Hard-Constraint. Hard-Constraint is a stipulation that must be fulfilled. Soft-constraint may be considered, but due to the scope of this study, they are not taken into account. When all events are given a time slot while not violating any of the hard constraints, the problem is considered solved. A solution (timetable) is considered acceptable when all of the following hard constraints are fulfilled:

- A time slot is assigned to each event in each course.
- No student group can hold two events at the same time.
- No lecturer should have two events going on at the same time.
- All events are held in the appropriate room.
- No two events take place at the same time in the same room.
- No event is held in a room that is smaller than the number of students expected to attend.

In this article, the UCTP is solved using the best-nests cuckoo search (BNCS) algorithm. The base cuckoo search (CS) algorithm is used by the BNCS algorithm. By dividing the nests into two groups, the BNSC algorithm was created (best-nests and normal-nests). To generate new solutions, the BNCS algorithm was limited to the best-nests. This article is divided into six sections, the remainder organized as; section 2 shows the related work, section 3 shows the CS, section 4 shows BNCS, section 5 shows how the testing has been carried out and the results, last but not least, section 6 talks about the conclusion suggest possible future work is presented.

2. RELATED WORKS

Some researchers have been inspired by the idea that research at high cost should be explored more and at a low cost more exploited. Therefore, the simulated annealing with reheating (SAR) algorithm for timeline problems was proposed by Babaei *et al*. [12]. It is through the initial temperature and reheating that the research exploration is directed. The methodology does not require setting the final temperature because the temperature is reheated if the research is stopped. Moreover, the algorithm does not need to run any set of different time limits. The method showed outstanding results without the need for strict tuning of some parameters (start temperature, end temperature, and Markov chain length), which are often required in conventional simulated Annealing.

For the SAR algorithm to work well, a manual setup of the neighborhood architecture is required to get good results. Goh *et al.* further developed the algorithm and proposed an algorithm called simulated annealing with improved reheating and learning [13]. The average cost changes were incorporated into the reheated temperature function. Furthermore, the reinforcement learning method was used to tune the composition of adjacent structures during the research. Overall, better results have been reported. Using a two-stage approach to the problem by Nagata [14]. In the first stage, an algorithm based on tabu search was used to build an initial solution. As for the second stage, reduce violations of the soft constraint by using random partial neighborhood search (RPNS). RPNS is based on Tabu Search except that the duration of Tabu is set to zero. It was found that varying neighborhood sizes during research work best for RPNS. The RPNS was configured for each data set in terms of neighborhood structure, proportion, update strategy, and several iterations. Very competitive results have been reported for the tested datasets especially International Timetabling Competition (ITC) 2007.

A parallel hybrid approach, proposed by Gozali *et al.*, to generate primary timescales for the application of a genetic algorithm (GA) improves the quality of the timelines by creating a hybrid algorithm

based on the parallel genetic algorithm and local search to solve the course schedule problem [15]. To ensure that the most fundamental restrictions are never broken, this combines a direct representation of the timetable with heuristic crossover operators. To solve the students' splitting, the multi-depth genetic algorithm (MDGA) was proposed by Alfian A. Gozali *et al.*, UCTP [16]. MDGA calculates GA using multiple stages, including multi-level mutagenesis and multi-depth constraint consideration. He demonstrates that MDGA can provide a viable timetable for student segmentation difficulties and that the outcomes are superior to earlier work and existing UCTP solutions.

3. CUCKOO SEARCH

Through the brood parasitism of some species of cuckoos, the CS was inspired, in which they lay their eggs in the nests of other host birds (of other species). The offending cuckoo has been observed to engage in head-to-head fights with some of the host birds. The host bird will either get rid of the foreign eggs or leave the nest and create a new nest elsewhere if it realizes that the eggs are not it is own. The host's nest is often the choice of the parasitic cuckoo in which to lay its eggs. Moreover, the timing of ovulation in some species is incredible. When the host bird has recently produced eggs at most, this nest is an option for the parasitic cuckoo. Its initial instinct for a cuckoo chick after hatching is to blindly push the host's eggs to expel from the nest so that the cuckoo chick's share of the food provided by the host bird is large. According to research [17], the cuckoo chick imitates the host chicks' sound to enhance the chances of foraging. Animals in the wild are haphazard in their quest for food. Because the next step is predicated on the present location/state and the transition probability to the next place, an animal's foraging path is practically a random walk. Its decision is based on an implicit probability that may be mathematically represented. The flying manner of numerous insects and animals has been proven in several investigations to exhibit the basic features of Levy flights [18]. Three idealistic rules [18] underpin CS:

a. Each cuckoo lays one egg at a time and deposits it in a nest that is picked at random.
b. The best nests with the highest quality eggs will be passed down to future generations.
c. Since the count of host nests is fixed, with a probability of pa (0, 1) the host bird correctly identifies the cuckoo egg, the host bird can either abandon it and start a new nest or discard the cuckoo egg. by the host, bird discovering act on a set of worst nests with a probability Pa (0, 1), and identified solutions ejected from further computations.

An important issue is the use of random walks and Levy flights (LF) in the generic equation to generate novel solutions. When generating new solutions Xi^(t+1) by (4).

$$\mathrm{Xi}\,(\mathrm{t}+1) \;=\; \mathrm{Xi}(\mathrm{t}) \;+\; \mathrm{a} \;*\; \mathrm{levy}(\lambda) \qquad (4)$$

The step size where it denotes a>0, which should be related to the problem metrics. In conditions of the great majority, use a=1. The LF simulates a random walk by using a Levy distribution to determine duration of each random stride is shown in (5).

$$\mathrm{levy}(\lambda) \;=\; \mathrm{t}(-\lambda)0.25 \;<\; \lambda \;\leq\; 2 \qquad (5)$$

It has an infinite mean and an unlimited variance. The steps effectively constitute a random walk process with a heavy tail and a power-law step-length distribution. Levy's roundup of the best solutions obtained so far should provide some new solutions, that will increase the speed of your local search. To protect that the system does not get stuck at the local optimum, a significant portion of the new solutions must be generated via remote domain randomization, with locations sufficiently far from the existing best solutions [19]. The CS can be summarized by basic steps in the pseudo-code presented by [17]: .

```
CS Algorithm:
BEGIN
  Objective function f(p), p= (p1,p2…pn)
  Generating initial population (P) of n host nests Pi (i=1 to n)
  While (t<MaxGenerations) and (!stop criterion)
     Get a cuckoo randomly via Lévy flights
     Evaluate its fitness Fi
     If( fj > Fi) Replace j by the new solution;
     Fraction pα of worse nests are abandoned and new nests are being built;
     Keep the best solutions or nests with quality solutions;
     Rank the solutions and find the current best
  End while
  Post process and visualize results
End
```

A cuckoo's sequential jumps/steps effectively create a random walk process with a heavy tail that follows a power-law step-length distribution. Furthermore, via random walks and mixing, a percentage of the poorest nests can be abandoned so that new nests may be formed at new sites. Random permutation can be used to mix the eggs/solutions based on their similarity/difference to the host eggs [17]. CS is successfully used to solve timetabling problems. Many applications, such as job scheduling [20], where CS algorithm was used to solve flexible-job scheduling problems, transposition ciphers attacks [21], and medical estimation systems [22], have used CS algorithm. CS is well suited to this type of behavior and can be applied to a variety of optimization problems [23], such as multi-objective optimization of distribution network configuration [24] and increasing DG capacity while minimizing system power loss by optimizing the location and capacity of distributed generators (DGs) [25].

4. BEST-NESTS CUCKOO SEARCH

Best-nests cuckoo search (BNCS) improved the CS algorithm by depending on the best solutions to create new solutions. In the basic CS generating solution, the selected of the worst solution was randomly in state space. This might result in a slower convergence to the optimal solution. Therefore, suggests splitting part of nests into two groups according to evaluation/fitness, the first group will be called the best-nests and the second will be normal-nests. The best-nests group size represents a ratio(m) of the total number of nests, while the rest of the nests are considered a normal-nests group. The new solution is created by selecting a nest at random from the group of best-nests. And then, compare with the group of normal-nests randomly. The fundamental phases in the pseudo-code described below summarize the BNCS.

```
BNCS Algorithm:
BEGIN
  Objective function f(p), p= (p1,p2…pn)
  Generate initial population of (P) host nests Pi (i=1 to n)
  m = ratio best-nests
  While (t < MaxGeneration) or (stop criterion)
     evaluate its quality/fitness Pi and sort from best to worst
     divide P into two groups of nests (best-nests(m) and normal-nests(n-m))
     Get a cuckoo randomly by Lévy flights from best-nests
     evaluate its quality/fitness Fi
     Choose a nest Fj among normal-nests randomly
     if (Fi > Fj) replace j by the new solution;
     Aggregation best-nests and normal-nests to P
     A fraction (pa) of worse nests is abandoned and new ones are built;
     Keep the best solutions (or nests with quality solutions);
     Rank the solutions and find the current best
  End while
  Postprocess results and visualization
END
```

A best-nests group is the major source of randomly selected new solutions (Fi), whereas a normal-nests group is the major source of randomly selected current solutions (Fj). Then comparing the new solution (Fi) with the current solution (Fj)., If the new solution (Fi) is better than the current solution (Fj), it will be replaced. Then, into the P nests, combine the two groups (best-nests and normal-nests). The rest of the steps in the algorithm as CS work.

5. EXPERIMENTAL RESULT

In this section, you'll find an explanation of datasets as well as their most notable features. Also, the setting of parameters for the CS and BNCS algorithms was demonstrated. Finally, there will be a discussion and analysis of the results.

5.1. Dataset description

To study the outcomes of algorithms' behavior, it is necessary to select various and appropriate datasets for testing. The dataset's input various data was provided by the Royal Institute of Technology (KTH) [3]. The number of student groups in each of the four input data files varies, as do the needed courses and classes. Each student group has two courses, with one to three events each course that must be arranged. The event density is calculated by dividing the number of events by the number of timeslots. The number of events to available timeslots density was taken into mind while creating the input files so that they did not differ too much. Also, the datasets are categorized into (kth_S, kth_M, kth_L, and kth_XL) depend on the size of (total events and total time slots). Table 1 summarizes the different data sets. The types of rooms are

lecture, lesson, and lab. A week is divided into five weekdays, Sunday to Thursday, and a weekday is divided into four timeslots of two hours each. This means that every room has a total of 20 timeslots. for instance, if the timetable has 8 rooms the total time slots equal 8x20=160.

Table 1. Summary of the different test data sets [3]

Input data file	kth_S	kth_M	kth_L	kth_XL
Lecture rooms	2	2	3	6
Lesson rooms	3	5	6	10
Lab rooms	3	5	7	11
Courses	12	15	21	29
Lecturers	9	12	15	21
Student groups	6	8	12	21
Total events	70	115	159	293
Total time slots	160	240	320	540
Event density	0.44	0.48	0.50	0.54

5.2. Parameter setting

The CS parameters are as follows:

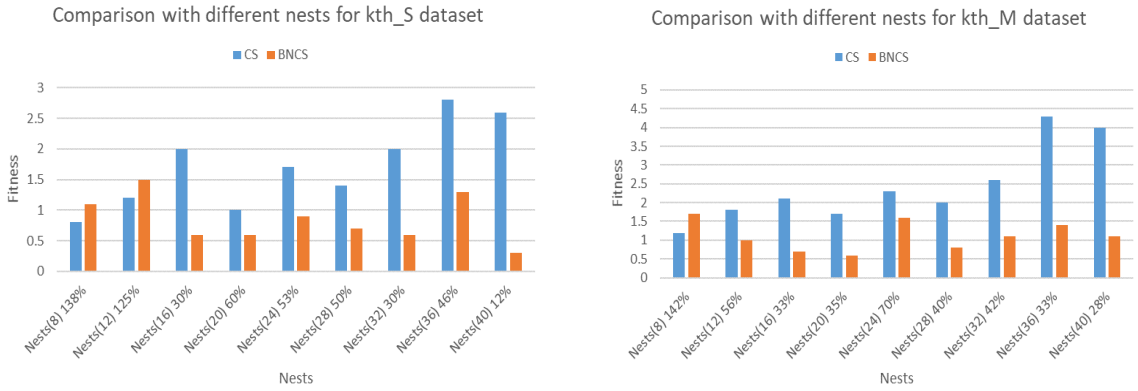
a. The size of the population or the number of nests, which ranges from 8 to 40 with an increment of
b. The number of iterations (10000, 20000, 40000, and 60000) is determined by the size of the dataset.
c. And the λ parameter that of Levy flights (LF) parameter, which ranges from 0.25 to 2.0.
d. probability of pa with a value of 0.25.
e. BNCS also has the same number of parameters as CS, in addition, the ratio best-nests parameter, which has a range of 15% to 35%.

Table 2 shows the values of the parameters that were considered. Then solve them by the obtained algorithms. Figure 1 (a)-(d) shows the results. As it can be seen, for BNCS and CS the best population size is 40.

The aim of the LF function is to return required the length of the step for a new solution from the current solution, where the length of the step integer value started from one-step and increments by one. The results of the length of the step were analyzed depending on a λ parameter change. It was found that length one step is repeated at different rates according to the value of λ. The best value is 1.0 because has a 50% of one step from a total of the result. Table 3 shows the values of the λ parameters vs percentage of one-step length.

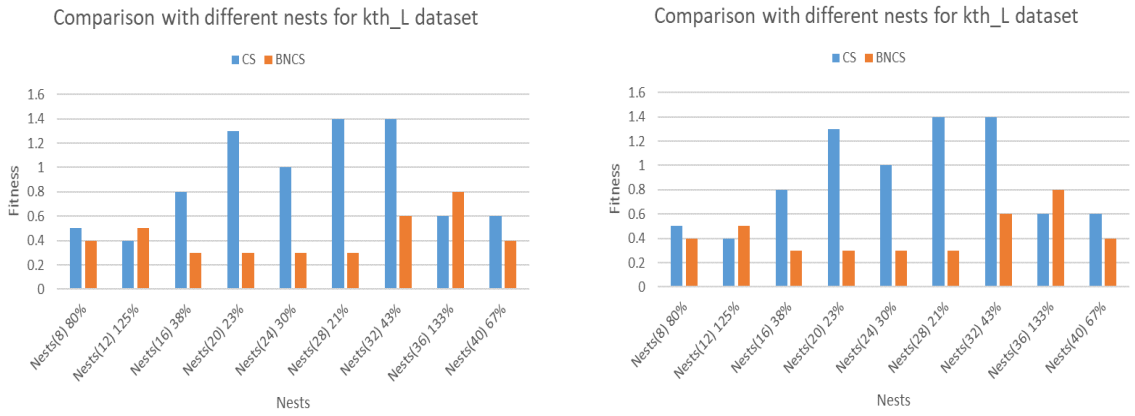
Table 2. Used parameter for testing

No.	Parameter Name	Value
1	Populations size (nests)	8, 12, 16, 20, 24, 28, 32, 36, 40
2	Iteration	10000, 20000, 40000, 60000
3	λ value	0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0
4	Pa	0.25
5	Raito best-nests	15%, 25%, 35%



(a) (b)

Figure 1. Average fitness value versus nests and express the difference as a percentage, (a) is small dataset, (b) is medium dataset



(c) (d)

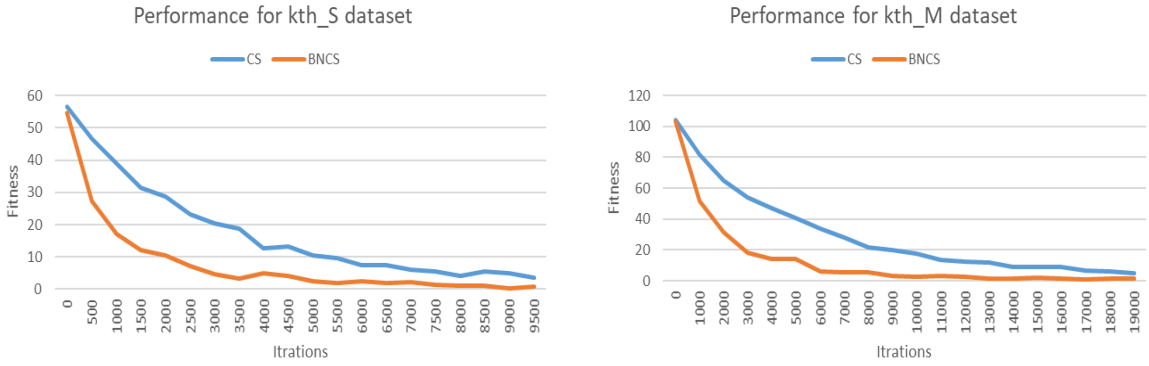
Figure 1. Average fitness value versus nests and express the difference as a percentage, (c) is large dataset, (d) is extra-large dataset

Table 3. The LF function and compute the ratio of impressions of a one-step length are called for the different testing values of the parameter for 500 iterations

λ Value	Total all step	Total One-Step	Percentage of One-Step
0.25	500	36	7%
0.5	500	116	23%
0.75	500	187	37%
1.0	500	246	49%
1.25	500	312	62%
1.5	500	405	81%
1.75	500	452	90%
2.0	500	500	100%

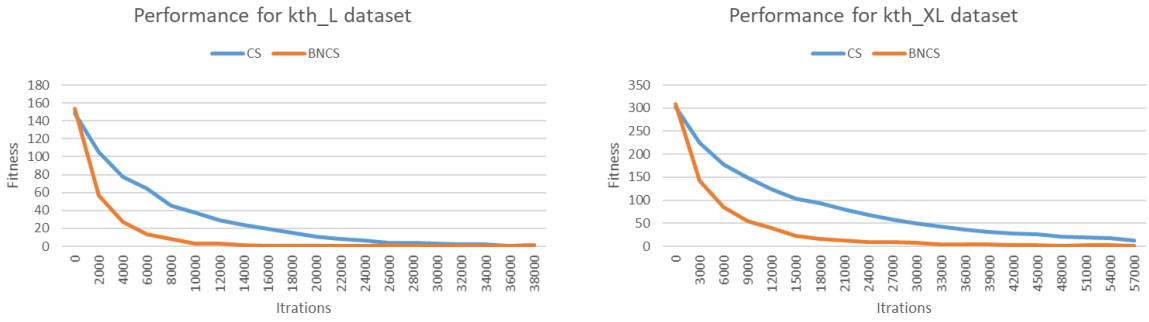
5.3. Discussion and analysis of the results

Many experiments were carried out according to the parameters mentioned in Table 2. Used a programming language C# and a specification Computer (CPU Core i5, RAM 8 MB, OS Windows 10 64-bit, HDD 512 GB). The result has shown BNCS's performance on CS has improved. As the figure contains the fitness values vertically and the number of iterations horizontally. This is accomplished by taking five readings of average fitness values on every iteration. BNCS accelerates to the optimal solution from CS. The difference in performance between CS and BNCS was clearly shown in Figure 2 (a)-(d). where (a) test with the kth-S dataset was considered as a small dataset. And (b) test with the kth-M dataset was considered as a medium dataset. Also, (c) and (d) was the test with the kth-L kth-XL datasets were considered large and extra-large datasets.



(a) (b)

Figure 2. Average fitness value versus iterations, (a) is small dataset, (b) is medium dataset



(c) (d)

Figure 2. Average fitness value versus iterations, (c) is large dataset, (d) is extra-large dataset *(continue)*

6. CONCLUSION

The UCTP is a fascinating and challenging academic topic; research in this field should focus on improving techniques to solving UCTP. The BNCS (BNCS) algorithm is presented in this research to solve the UCTP. The approach takes into consideration representations of both the basic CS and excellent nests to optimize the UCTP and to be the major source for selection. The BNCS algorithm's experimental findings are encouraging. Despite the problem's great complexity and Nondeterministic Polynomial Hard (NP-Hard), the performance of the BNCS is excellent. Within a short processing period, good results were obtained This is done through parameter values (Nests=40, λ=1, Pa=0.25, Raito Best-Nests=1.0). Even if the mathematical model can provide the ideal schedule for a specific instance of the issue, the needed process time to attain optimality in real-world settings would be unrealistic. As a result, heuristic strategies must constantly be employed. In this sense, other ways should be studied as well, despite the BNCS's strong performance. As a result, in the second supplement of this study, we would want to develop and experiment with different heuristic strategies to compare their performance.

REFERENCES

[1] A. Gunawan, K. M. Ng and K. L. Poh, "Solving the Teacher Assignment-Course Scheduling Problem by a Hybrid Algorithm," *International Journal of Computer, Information, and System Science, and Engineering*, vol. 1, no. 2, pp. 136–141. 2007.

[2] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007, doi: 10.1016/j.ejor.2005.08.012.

[3] A. Salman and R. Hanna, "A Comparative Study between GeneticAlgorithm, Simulated Annealing and a Hybrid Algorithm for solving a University Course Timetabling Problem," Degree Project in Computer Science, KTH Royal Institute of Technology School of Electrical Engineering and Computer Science, KTH, Stockholm, Sweden 2018.

[4] R. A.-M. Nabeel, "Hybrid genetic algorithms with great deluge for course timetabling," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 4, pp. 283–288, 2010.

[5] R. Karimpour, M. R. Khayyambashi and N. Movahhedinia, "Load balancing in grid computing using ant colony algorithm and max-min technique," *Malaysian Journal of Computer Science*, vol. 29, no. 3, pp. 196–206, 2016, doi: 10.22452/mjcs.vol29no3.3.

[6] M. Ghalehgolabi and A. Rezaeipanah, "Intrusion Detection System Using Genetic Algorithm and Data Mining Techniques Based on the Reduction Features, "*International Journal of Computer Applications Technology and Research*," vol. 6, no. 11, pp. 461–466, 2016, doi: 10.7753/IJCATR0611.1003.

[7] J. Makka, H. Monga and S. Baghla, "Reduction of Inter-Symbol Interference Using Artifical Neural Network System in Multicarrier OFDM System," *Electric Electron Tech Open Acc J.*, vol. 2, no. 1, pp. 20-23, 2018.

[8] A. Alkan and E. Ozcan, "Memetic algorithms for timetabling," *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, 2003, pp. 1796-1802 Vol.3, doi: 10.1109/CEC.2003.1299890.

[9] M. Nandhini and S. Kanmani, "A Survey of Simulated Annealing Methodology for University Course Timetabling 2," *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, pp. 1–3, 2009.

[10] C. H. Aladag, G. Hocaoglu and M. Basaran, "The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12349-12356, 2009, doi: 10.1016/j.eswa.2009.04.051.

[11] S. Abdullah, H. Turabieh, B. McCollum and P. McMullan, "A hybrid metaheuristic approach to the university course timetabling problem," *Journal of Heuristics*, vol. 18, pp. 1–23, 2012, doi: 10.1007/s10732-010-9154-y.

[12] H. Babaei, J. Karimpour and A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers and Industrial Engineering*, vol. 86, pp. 43–59, 2015, doi: 10.1016/j.cie.2014.11.010.

[13] S. L. Goh, G. Kendall and N. R. Sabar, "Improved local search approaches to solve the post enrolment course timetabling problem," *European Journal of Operational Research*, vol. 261, no. 1, pp. 17–29 2017, doi: 10.1016/j.ejor.2017.01.040.
[14] S. L. Goh, G. Kendall and N. R. Sabar, "Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem," *Journal of the Operational Research Society*, vol. 70, 873–888, 2019, doi: 10.1080/01605682.2018.1468862.
[15] Y. Nagata, "Random partial neighborhood search for the post-enrollment course timetabling problem," *Computers and Operations Research*, vol. 90, pp. 84–96, 2018, doi: 10.1016/j.cor.2017.09.014.
[16] A. A. Gozali, S. Fujimura, "Solving University Course Timetabling Problem Using Multi-Depth Genetic Algorithm," *The 2nd ACM Chapter International Conference on Educational Technology, Language and Technical Communication (ETLTC2020)*, vol. 77, 01001, 2020, doi: 10.1051/shsconf/20207701001.
[17] A. A. Gozali, B. Kurniawan, W. Weng and S. Fujimura, "Solving university course timetabling problem using localized island model genetic algorithm with dual dynamic migration policy," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 15, no. 3, pp. 389–400, 2020, doi: 10.1002/tee.23067.
[18] X. Yang and Suash Deb, "Cuckoo Search via Lévy flights," *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210-214, doi: 10.1109/NABIC.2009.5393690.
[19] N. Bacanin, "Implementation and performance of an object-oriented software system for cuckoo search algorithm," *International Journal of Mathematics and Computers in Simulation*, vol. 6, pp. 185–193, 2012.
[20] R. Hilton, "Automated Cryptanalysis of Monoalphabetic Substitution Ciphers Using Stochastic Optimization Algorithms," Ph.D. dissertation, Department of Computer Science and Engineering, University of Colorado, Denver, 2012.
[21] A. T. Saadeq Al-Obaidi and S. A. Hussein, "Two Improved Cuckoo Search Algorithms for Solving The Flexible Job-Shop Scheduling Problem," *International Journal on Perceptive and Cognitive Computing*, vol. 2, no. 2, pp. 25-31, 2016, doi:10.31436/ijpcc.v2i2.34.
[22] A. T. Sadiq, L. Ali and H. Kareem, "Attacking Transposition Cipher Using Improved Cuckoo Search," *Journal of Advanced Computer Science and Technology Research*, vol. 4, no. 1, pp. 22–32, 2014.
[23] A. T. Sadiq Al-Obaidi, "Cuckoo Search-Based Bayesian Networks for Medical Estimation System," *2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, 2015, pp. 92-102, doi: 10.1109/ACSAT.2015.55.
[24] T. T. Nguyen, "Optimization of distribution network configuration with multi objective function based on improved Cuckoo search algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp: 1685–1693, 2020, doi: https://doi.org/10.11591/eei.v9i4.1886.
[25] T. T. Nguyen, T. T. Ngoc, T. T. Nguyen, T. P. Nguyen and N. A. Nguyen, "Optimization of location and size of distributed generations for maximizing their capacity and minimizing power loss of distribution system based on cuckoo search algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp: 1769–1776, 2021, doi: https://doi.org/10.11591/eei.v10i4.2278.

BIOGRAPHIES OF AUTHORS

Hasanen S. Abdullah received a B.Sc. degree in Computer Science from the University of Technology, Baghdad, Iraq, in 2000. he graduated from the University of Technology, Baghdad, Iraq, in 2004 with an M.Sc. degree in Computer Science. he graduated from the University of Technology, Baghdad, Iraq, in 2008 with an Ph.D. degree in Computer Science. His research interests include the artificial intelligence techniques and applications. He can be contacted at email: hasanen.s.abdullah@uotechnology.edu.iq.

Mohammed A. Jebur received a BSc degree in Computer Science from the University of Technology, Baghdad, Iraq, in 2009. His research interests include the application of optimization techniques as solutions to combinatorial and optimization problems. He can be contacted at email: cs.19.10@grad.uotechnology.edu.iq.