TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

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

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 950 – 960
© TÜBİTAK
doi:10.3906/elk-1809-175

Prey-predator algorithm for discrete problems: a case for examination timetabling problem

Surafel Luleseged TILAHUN[*]
School of Mathematical Science, University of Zululand, KwaDlangezwa, South Africa

Received: 24.09.2018 • **Accepted/Published Online:** 06.12.2018 • **Final Version:** 22.03.2019

Abstract: The prey-predator algorithm is a metaheuristic algorithm inspired by the interaction between a predator and its prey. Initial solutions are put into three categories: the better performing solution as the best prey, the worst performing solution as a predator, and the rest as ordinary prey. The best prey totally focuses on exploiting its neighborhood while the predator explores the search space searching for a promising region in the search space. The ordinary prey will be affected by these two extreme search behaviors of exploration and exploitation. The algorithm has been tested and found to be effective in solving different problems arising from different disciplines including engineering, tourism, and management. Originally, the algorithm was designed to deal with continuous problems. However, many problems arising from real aspects are not continuous. Hence, in this paper the prey-predator algorithm will be extended to suit discrete problems. Examination timetabling is used to test the approach. The simulation results with appropriate statistical analysis show that the approach is as good as the cumulative best performance of results recorded in the literature for the selected benchmark problems.

Key words: Prey-predator algorithm, discrete prey-predator algorithm, metaheuristic, combinatorial optimization, examination timetabling

1. Introduction

A mathematically formulated decision-making problem of finding the best action in order to optimize a given objective(s) is called an optimization problem. It can be categorized in different ways. These include as categories based on the number of objectives; as single-objective and multiobjective problems, based on the behavior of the decision variables; as continuous, discrete, and mixed problems; and also based on the nature of the variables as deterministic and nondeterministic problems. Due to resource limitations (for example, the number of staff and number of products like cars), many real problems can be formulated as constrained discrete optimization problems. A combinatorial optimization problem is an optimization problem that has a finite and discrete, usually large, solution space [2]. Different real scenarios that are formulated as combinatorial optimization problems can be mentioned, including the exam timetabling problem [3], travel salesman problem [1], and bin packing problem [4].

Different exact optimization methods are introduced to solve optimization problems based on their properties; for instance, if the problem is linear and continuous, then one can use a simplex algorithm. However, these exact methods are shorthanded when dealing with difficult real problems that do not fall into the categories introduced for exact methods. In such cases, a metaheuristic algorithm will become an ideal choice. Even though

[*]Correspondence: surafelaau@yahoo.com

metaheuristic algorithms do not guarantee optimality, they are tested and found to give reasonable solutions under appropriate implementations. Since the introduction of evolutionary algorithms, different metaheuristic algorithms have been introduced. The prey-predator algorithm is one of the these metaheuristic algorithms, which mimics the scenario of a predator that runs after its prey. Adjusting the degree of exploration and exploitation is one of the challenging issues and has been in the forefront of research issues in the field [5]. The prey-predator algorithm gives a clear way of controlling exploration and exploitation [15, 17]. It has also been studied and tested in different problems and is found to be effective [6–10, 14, 15, 18, 20, 21]. In addition, it is a generalized swarm-based algorithm where some well-known algorithms occur as a special case of this algorithm, including the particle swarm algorithm and modified firefly algorithm [11]. Furthermore, the success of the algorithm motivates researchers to propose a generalized version of the algorithm by incorporating a hyperheuristic approach [17], which in turn leads researchers towards proposing a generalized swarm framework [19].

Along with continuous optimization problems, metaheuristic algorithms are also used for combinatorial optimization problems. Many of these algorithms were originally proposed for continuous problems and later modified to be used for discrete optimization problems. The "no free lunch" theorem proves that there is no superior metaheuristic algorithm for combinatorial optimization problems. Therefore, modifying the newly introduced prey-predator algorithm for the use of solving constrained combinatorial optimization problems is one contribution to the solution methods of these problems. Hence, in this paper, the prey-predator algorithm will be extended to suit discrete problems and used to solve an examination timetabling problem. The simulation results along with the statistical analysis suggest that the discrete prey-predator algorithm performs as well as the collection of the best performances of algorithms from the literature. For the simulation six uncapacitated examination timetabling problems are used. The advantage of easily controlling capability and tuning of the degree of exploration and exploitation of the prey-predator algorithm gives the extended version a significant performance improvement over other approaches.

The paper is organized as follows. In the next section basic concepts will be discussed on the constrained combinatorial optimization problem along with a brief explanation of the prey-predator algorithm. In Section 3 the proposed approach will be given, followed by a discussion on the timetabling problem in Section 4. Finally, the conclusion will be presented in Section 6, after a simulation-based discussion in Section 5.

2. Preliminaries

2.1. Combinatorial optimization problem

A combinatorial optimization problem is an optimization problem with discrete and finite solution space [12]. It has an objective function, $f(x)$, along with the feasible region, S. If the values that can be assigned to the decision variables are from a discrete set E then the problem can be given as shown in Eq. (1).

$$\begin{aligned} &min\ f(x) \\ &s.t.\ x \in S \subseteq E^n \end{aligned} \tag{1}$$

Note that a solution x is a vector of n dimensions with entries from E; for example, E can be integers between 1 and 100. In constrained optimization problems, the variables may be restricted so as not to take some values from set E. Hence, a general formulation for a constrained optimization problem can be given by:

$$\begin{aligned} &min\ f(x) \\ &s.t.\ x \in S \subset E^n \end{aligned} \tag{2}$$

A particular example will be the examination timetabling problem. For each exam you can assign a time slot, which is arranged as 1, 2, and so on. However, there should not be a clash between exams, which means that exams with common students should not be arranged in the same slot. Hence, assigning the exams to the slots randomly may possibly create a clash by assigning two exams with common students in the same slot. In addition to the exam clash constraint, exam room capacity or number of available invigilators, etc., can be used as constraints to be considered while generating initial solutions as well as solving the problem.

A vector $x*$ is said to be a solution for a combinatorial optimization problem if and only if $x^* \in S$ and $f(x^*) \leq f(x)$ for all $x \in S$.

2.2. Prey-predator algorithm

The prey-predator algorithm (PPA) is one of the swarm-based metaheuristic algorithms introduced for continuous problems [11, 16]. It mimics the interaction between a predator (which runs after its prey) and its prey in the natural environment. In the algorithm a randomly generated set of feasible solutions will be put into three categories: a predator, ordinary prey, and the best prey, based on their performances in the objective function. In each iteration of the algorithm the predator tends to explore the solution space with bigger step length and also run after the weak prey, which is the prey with the worst performance in the objective function. However, the best prey will only do a local search and totally focuses on the exploitation of the neighborhood. On the other hand, the ordinary prey will run away from the predator or follow better prey, prey with better performance in the objective function. This updating move of an ordinary prey is done based on the algorithm parameter called probability of follow-up.

Hence, probability of follow-up is one of the algorithm parameters that control the movement of the ordinary prey. High probability follow-up means 'following better prey' will have a higher chance than 'randomly run away from the predator'. If a randomly generated number is under the probability of follow-up then it will follow better prey and do a local search as well, but if the randomly generated number is greater than the probability of follow-up then the prey will randomly run away from the predator. The other algorithm parameter is step length. There are two step lengths, λ_{max} and λ_{min}. λ_{max} is a step length for exploration, whereas λ_{min}, which is shorter than λ_{max}, is a step length for exploitation. m is another algorithm parameter, which determines the number of random unit directions used for exploitation or local search purposes. The algorithm is summarized in Table 1.

3. Discrete prey-predator algorithm

As presented in Section 2, a combinatorial optimization problem is an optimization problem with discrete values for the decision variables. When there are cases that need to be fulfilled, the problem becomes challenging and finding a feasible solution by itself will be another challenging task because a random solution within the boundary may not always feasible. In order to construct a feasible solution to proceed with the PPA, different problem-specific methods can be used. Even after generating initial feasible solutions, the other biggest challenge in applying the PPA for discrete problems is in the updating process. Unlike the continuous case, while updating

Table 1. Prey-predator algorithm.

Step 1	Generate random and feasible N solutions
Step 2	Sort the solutions based on their performance in the objective function from worst to best and Assign x_1 to be the predator, x_N to be the best prey, and the rest ordinary prey
Step 3	Move the predator randomly and also towards x_2 i.e. $x_1 := x_1 + (\lambda_{max} rand)u + (\lambda_{min} rand)u_{1,2}$ where $rand$ is a random number from uniform distribution between 0 and 1, u is a random unit vector, and $u_{1,2} = \begin{cases} \frac{x_2 - x_1}{\Vert x_2 - x_1 \Vert}, & if x_1 \neq x_2 \\ 0, & otherwise \end{cases}$
Step 4	Move the best prey using a direction chosen from randomly generated m unit directions and zero vector i.e. $x_N := x_N + (\lambda_{min} rand)u$ where $u = \underset{u_i \in u_1, u_2, \ldots, u_m, 0}{min} f(x_N + (\lambda_{min} rand)u_i)$
Step 5	Update the ordinary prey either by following better performing prey if probability of follow-up is met or randomly away from the predator i.e. if $rand \leq$ Probability of follow-up $x_i := x_i + (\lambda_{max} rand)u_f + (\lambda_{min} rand)u$ where $u_f = \frac{\sum_{j=i+1}^{N}(x_j - x_i)}{\Vert \sum_{j=i+1}^{N}(x_j - x_i) \Vert}$ and u is similar to the one in step 4 else (if $rand >$ Probability of follow-up) $x_i := x_i + (\lambda_{max} rand)u_{rand}$ where $u_{rand} = \begin{cases} u & if \Vert x_1 - (x_i + u) \Vert \geq \Vert x_1 - (x_i - u) \Vert \\ -u & otherwise \end{cases}$ for a random unit direction u
Step 6	If a termination criterion is met stop else go back to step 2

a solution, using the updating equations presented in Table 1, the updated solution may become infeasible. Hence, the updating process should be modified while mimicking the same scenario. In this paper we propose a version of the PPA for constrained combinatorial optimization problems with a new updating and local search mechanism. Consider the following problem:

We want to minimize $f(x)$ by choosing n dimensional vector value for x in such a way that x fulfills the conditions given as set S, feasibility. Without loss of generality we can also assume that each component or entry of vector x can be assigned an integer between 1 and a natural number D, for $D > 1$.

First, by using problem domain-based methods (like saturation degree for exam timetabling problems [13]), a random set of feasible solutions will be generated, say $x_1, x_2, ...x_N$. The number of solutions should be more than two as we have three categories of solutions in the PPA. Among these solutions, based on their performance in the objective function, $f(x)$, they will be put into three categories: the one with the worst performance will be called the predator, represented by x_p; the one with the best performance is called the best prey; represented by x_b, and the rest are ordinary prey. Unlike in the continuous case, the local search will be done by changing only a number of entries of the solution based on a new algorithm parameter called probability of change (p_c). In this context a local search is when solutions try to look for another better solution by changing only a small number of its components. Hence, for each component, if a randomly generated number is less than or equal to p_c, then that entry will be changed as long as the solution remains feasible. However, if the change results in infeasibility, it will be rejected. Hence, a small number of p_c means exploitation, whereas a larger number of p_c leads to exploration. As discussed in the previous section, the other algorithm parameter in the continuous optimization case is m, the number of random directions for the local search of the best prey. Similarly, we can do m local updates and choose the one with the best performance; we add the original best prey in this comparison. The updating process of the best prey is summarized in Table 2.

Table 2. Updating process for the best prey ($randperm(D)$ is a random arrangement of D natural numbers).

```
Input $p_c, m, D$
for $k = 1 : m$
    $y_k = x_b$
    for $i = 1 : n$
        if $rand \leq p_c$
            $T = randperm(D)$
            for $j = 1 : D$
                if $y_k(i) \neq T(j)$
                    if making $y_k(i) = T(j)$ does not make $y_k$ infeasible
                        then $y_k(i) = T(j)$
                        break the $j$ loop (or update $j = D$)
                    end if
                end if
            end for
        end if
    end for
end for
Compare $y_1, y_2, ..., y_m, x_b$ and take the best to update $x_b$
```

The updating process of an ordinary prey x_i depends on the probability of follow-up (p_f). If a randomly generated number between zero and one from a uniform distribution is at most equal to the probability of follow-up then a ‘following-up' updating will be performed. The update is done for each component based on the probability of change (p_c), and here p_c should be higher than the value used for the local search. That is, for each component, a random number is generated and compared with the probability of change to check if that component needs to be updated by moving it towards promising solutions. The updating equation is given in Eq. (3).

$$x_i(k) := round[x_i(k) + (2rand)sign(\sum_j (x_j(k) - x_i(k)))] \tag{3}$$

For each better prey j and components k, $rand$ is a random number between 0 and 1, with $round$ being a rounding function to the nearest integer and $sign$ a function that will return –1 if the input is negative, 1 if the input is positive, and 0 otherwise.

Similarly, if the probability of follow-up is greater than a randomly generated number, then the prey will randomly run away from the predator; that will be done again component-wise as given in Eq. (4).

$$x_i(k) := round[x_i(k) + (2rand)(x_j(k) - x_p(k))] \tag{4}$$

For the predator x_p and components k, $rand$ is a random number between 0 and 1.

In Eqs. (3) and (4), a magnifying value of 2 is used, which allows the updating process to explore beyond the point of interest, i.e. to explore regions other than those between the two points. The updating process of an ordinary prey is summarized as follows in Table 3.

The predator focuses on exploration. That is done with a similar updating mechanism as used for the best prey but with bigger p_c. That leads the predator to have high probability of changing most of its components to end up in a new solution neighborhood.

Table 3. Updating of an ordinary prey x_i.

```
Input $p_f, p_c, D$
if $rand \leq p_f$
    for $i = 1 : n$
        if $rand \leq p_c$
            if updating $x_i(k)$ using Eq. (3) does not make $x_i$ infeasible
                then update $x_i(k)$ using Eq. (3)
            end if
        end if
    end for
else
    for $i = 1 : n$
        if $rand \leq p_c$
            if updating $x_i(k)$ using Eq. (4) does not make $x_i$ infeasible
                then update $x_i(k)$ using Eq. (4)
            end if
        end if
    end for
end if
```

Note that the step lengths, λ_{min} and λ_{max}, in the standard prey-predator algorithm are replaced by p_c.

4. Examination timetabling problem

The examination timetabling problem is a problem of assigning a given number of exams to available time slots in such a way that exams with common students should be arranged in different slots. There are two types of exam timetabling problems, capacitated and uncapacitated. In capacitated exam time tabling problems the capacity of the exam room will be taken into consideration, whereas in the uncapacitated case the room capacity is not considered. With the constraint being that there is no clash between exams with common students, the objective will be to spread the exams in such a way that students will have good gap of slots between exams, i.e. the objective is to minimize a penalty for two exams with common students scheduled with a smaller number of slots in between and is given by Eq. (5).

$$f(x) = \frac{1}{S}\sum(floor(2^{5-|x_i-x_j|}))N_{ij} \tag{5}$$

Here, S is total number of students, N_{ij} is number of students taking exam i and exam j, and the $floor$ function rounds up the result to the smallest and nearest integer.

This means that the maximum penalty is when two exams with common students are scheduled in the same slot and it will be 32 times the number of common students. If there is a slot in between then the penalty will decrease by half. In a similar manner, the ideal will be to have the maximum possible slots between the two exams, which is set to be 5, resulting in a zero penalty.

5. Simulation

5.1. The benchmark problems

Carter benchmark problems are well-known uncapacitated exam timetabling benchmark problems [22]. Six benchmark problems from Carter uncapacitated timetabling problems found in [23] are selected. These benchmark problems are named HEC92, STA83, UTE92, TRE92, CAR92, and UTA92. The properties of these datasets or problems are given in Table 4.

Table 4. Properties of the benchmark problems [23].

Problem	Exam number	Student number	Enrollment	Density	Time slots
HEC92	81	2823	10,632	0.42	18
STA83	139	611	5751	0.14	13
UTE92	184	2750	11,793	0.08	10
TRE92	261	4360	14,901	0.18	23
CAR92	543	18,419	55,522	0.14	32
UTA92	622	21,266	58,979	0.13	35

5.2. Wilcoxon signed-rank test

The Wilcoxon signed-rank test is a nonparametric test that does not assume the normality of the data, used to test whether the population means of two data pools are the same or different. Suppose there are data A and B with N observations, a_is for A and b_is for B. The null hypothesis is that the median difference between the pairs of observations is zero. The test can be conducted as follows:

1. Step 1. For each pair of data compute the absolute difference and also the sign, i.e. $c_i = |b_i - a_i|$ and $s_i = sign(b_i - a_i)$, where the $sign$ function returns –1, 1, or 0 if the entry is negative positive, or zero, respectively.

2. Step 2. Sort the resulting differences based on the absolute difference from smallest to largest and exclude those entries with absolute difference of zero. Suppose there are t entries with absolute difference of zero.

3. Step 3. For the entries with nonzero difference, rank them starting from 1 increasingly. If there is a tie the entries take the average ranks they span. That is, after an entry with rank k, if there are p entries with the same absolute difference then their rank will be computed using Eq. 6.

$$R = k + \frac{p+1}{2} \tag{6}$$

4. Step 4. Calculate the test statistics by taking the sum of the product of the sign, s_i, and the rank, R, using Eq. 7.

$$T = \sum_{i=1}^{N-t} s_i R_i \tag{7}$$

5. Step 5. Reject the null hypothesis if Eq. 8 is true, with a significance level of α.

$$|T| \geq T_{\alpha,N-t} \tag{8}$$

Here, $T_{\alpha,N-t}$ will be read from the reference table.

5.3. Simulation results and discussion

The simulation is done using MATLAB R2011b on an Intel Core i5, 2400 CPU and 3.10 GHz, with 32-bit operating system desktop machine. The algorithm parameters are tuned based on recommendations from [11] and [16]. The total number of initial solutions was set to be 20, with probability of follow-up, probability of change for local search, and probability of change for exploration being 0.75, 0.2, and 0.6, respectively. Furthermore, the number of random directions for local search, m, is set to be 100. The maximum number of iterations, which was set to be 50, was used as a termination criterion. In addition, the number of best prey was increased and set to be 8, to increase the local search behavior of the algorithm.

In order to generate feasible initial solutions, saturation degree was used with minimum penalty. The simulation is done 30 times in order to compute the mean and standard deviation of the results and is reported in Table 5.

Table 5. Simulation results of the PPA for the benchmark problems.

Problem	HEC92	STA83	UTE92	TRE92	CAR92	UTA92
Best result	11.5432	151.1554	24.0767	9.6676	5.9065	5.0671
Mean	11.9801	154.3231	24.3879	9.7568	6.0203	5.1418
Standard deviation	0.0231	2.0369	0.1692	0.0496	0.0824	0.0631
Average CPU time	52.6511	69.5811	203.5500	1331.1000	1145.0000	44448.0906

Exam timetabling has been one of the applications of combinatorial optimization problems and different studies have been conducted especially using Carter benchmark problems. Table 6 summarizes the solution ranges of the selected problems in the literature.

Table 6. Solutions for the benchmark problems from the literature.

Problems	Best results using PPA	Best results from literature		Worst results from literature	
		Results	References	Results	References
HEC92	11.5432	9.2	[26]	13.8	[29]
STA83	151.1554	157.3	[24]	168.3	[25]
UTE92	24.0767	24.4	[26]	29	[27]
TRE92	7.9679	7.87	[28]	10	[29]
CAR92	5.9065	3.9	[30]	6.2	[22]
UTA92	3.5671	3.1	[30]	4.2	[29]

Hypothesis testing based on the Wilcoxon signed-rank test is used to compare the simulation results with previously known best results from the literature. Hence, there are two data sets, as given in Table 7, for the statistical analysis.

Table 7. The set of data for hypothesis testing (A is a datum from simulation and B is the best result from the literature for each problem).

A	11.5432	151.1554	24.0767	7.9679	5.9065	3.5671
B	9.2	157.3	24.4	7.87	3.9	3.1
$B - A$	2.3432	6.1446	0.3233	0.0979	2.0065	0.4671
sign	–1	1	1	–1	–1	–1

The null hypothesis states that the median difference between the two datasets is zero, i.e. the performance of the proposed approach when compared to the best results known in the literature has same median.

The rank of the data entries is given in Table 8.

Table 8. Ranked datasets.

A	7.9679	24.0767	3.5671	5.9065	11.5432	151.1554
B	7.87	24.4	3.1	3.9	9.2	157.3
c	–1	1	–1	–1	–1	1
R	1	2	3	4	5	6

Based on the discussion given in Section 5.2, the test statistics T is found to be –5. For $\alpha = 0.05$:

$$T_{\alpha,6} = 21 \tag{9}$$

Since $|T| = 5 < 21 = T_{0.05,6}$ (as presented in Eq. (8)), the null hypothesis is accepted under the given level of significance. Hence, there is no significant difference between the medians of the collective results of the best performing algorithms and the proposed approach.

The proposed algorithm indeed outperforms existing algorithms for some of the problems and is outperformed by some for the other problems. It should be noted that based on the 'no free lunch theorem' there does not exist a single algorithm to outperform all existing algorithms over all sets of problem domains. Furthermore, the comparison here is done not with each of the proposed algorithms separately but with the best performance of the algorithms over the problems. Hence, the proposed algorithm is a promising algorithm to be among the pool of solution methods of discrete problems. Testing it on problems from other problem domains can be done in the future along with enhancing and modifying to boost the performance of the algorithm.

6. Conclusion

With the success of solving different problems and the advantage of easily controlling the ratio of exploration and exploitation, the prey-predator algorithm has been applied and used in different applications. These applications mainly focus on continuous problems, and hence in this paper the discrete version of the algorithm with a suitable local search mechanism to suit discrete search space is introduced. Examination timetabling has been used to test the proposed approach. Six uncapacitated timetabling problems are taken from Carter benchmark problems. Based on the statistical analysis, with a two tailed t-test, the simulation results show that the proposed approach is as good as the collective best results of the benchmark problems from the literature. In addition, the simulation results show that the approach is stable with smaller standard deviation. Hence, the algorithm can be a promising addition for discrete problems. The performance of the approach can further be checked by applying it on problems from other problem domains as future work.

References

[1] Little DL, Murty GK, Sweeney WD, Karel C. An algorithm for the traveling salesman problem. Oper Res 1963; 11: 972-989.

[2] Schrijver A. On the history of combinatorial optimization (till 1960). In: Aardal K, Nemhauser GL, Weismantel R, editors. Discrete Optimization, Volume 12 of Handbooks in Operations Research and Management Science. Amsterdam, the Netherland, Science Direct, 2005, pp. 1-68.

[3] Ayob M, Malik A, Abdullah S, Hamdan AR, Kendall G, Qu R. Solving a practical examination timetabling problem: a case study. In: Gervasi O, Gavrilova M, editors. ICCSA-2007, Part III LNCS, Vol. 4707. Heidelberg, Germany: Springer, 2007, pp. 611-624.

[4] Korf RE. A new algorithm for optimal bin packing. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence; 28 July–1 August 2002; Edmonton, Canada. pp. 731-736.

[5] Yang XS. Nature-Inspired Metaheuristic Algorithms. 2nd ed. London, UK: Luniver Press, 2010.

[6] Bahmani-Firouzi B, Sharifinia S, Azizipanah-Abarghooee R, Niknam T. Scenario-based optimal bidding strategies of GENCOs in the incomplete information electricity market using a new improved prey-predator optimization algorithm. IEEE Syst J 2015; 99: 1-11.

[7] Dai W, Liu Q, Chai T. Particle size estimate of grinding processes using random vector functional link networks with improved robustness. Neurocomputing 2015; 169: 361-372.

[8] Tilahun SL, Goshu NN, Ngnotchouye JMT. Prey predator algorithm for travelling salesman problem: application on the Ethiopian tourism sites. In: Vasant P, Kalaivanthan M. editors. Handbook of Research on Holistic Optimization Techniques in the Hospitality, Tourism, and Travel Industry. Hershey, PA, USA: IGI Global, 2017, pp. 400-422.

[9] Hamadneh N, Tilahun SL, Sathasivam S, Ong HC. Prey-predator algorithm as a new optimization technique using in radial basis function neural networks. Research Journal of Applied Sciences 2013; 8: 383-387.

[10] Tilahun SL, Ong HC. Comparison between genetic algorithm and prey-predator algorithm. Malay J Fund App Sc 2013; 9: 167-170.

[11] Tilahun SL. Prey predator algorithm: a new metaheuristic optimization approach. PhD, Universiti Sains Malaysia, Penang, Malaysia, 2013.

[12] Korte B, Vigen J. Combinatorial Optimization: Theory and algorithm. 2nd ed. Heidelberg, Germany: Springer-Verlag, 2002.

[13] Ross P, Hart E, Corne D. Some observations about GA-based exam timetabling. In: Burke EK, Carter M. editors, LNCS - 1408, Practice and Theory of Automated Timetabling II. Toronto, Canada: Springer-Verlag, 1997, pp. 115–129.

[14] Tilahun SL, Ngnotchouye JMT. Prey predator algorithm with adaptive step length. Int J Bio-Ins Comp 2016; 8: 195-204.

[15] Tilahun SL, Ong HC, Ngnotchouye JMT. Extended prey-predator algorithm with a group hunting scenario. Adv Oper Res 2016; 2016: 1-14.

[16] Tilahun SL, Ong HC. Prey predator algorithm: a new metaheuristic optimization algorithm. Int J Info Tech Dec Mak 2015; 14: 1331-1352.

[17] Tilahun SL. Prey predator hyperheuristic. App Soft Comp 2018; 59: 104-114.

[18] Hamadneh N, Khan W, Tilahun S. Optimization of microchannel heat sinks using prey-predator algorithm and artificial neural networks. Machines 2018; 6: 1-18.

[19] Tilahun SL, Tawhid MA. Swarm hyperheuristic framework. J Heur (in press).

[20] Ong HC, Tilahun SL, Lee WS, Ngnotchouye JMT. Comparative study of prey predator algorithm and firefly algorithm. Intell Aut Soft Comp 2017; 2017: 1-8.

[21] Tilahun SL, Matadi MB. Weight minimization of a speed reducer using prey predator algorithm. Int J Man Mat Mech Eng 2018; 8: 19-32.

[22] Carter MW, Laporte G, Lee SY. Examination timetabling algorithmic strategies and applications. J Oper Res Soci 1996; 47: 373-383.

[23] Qu R, Burke EK, McCollum B, Merlot LTG, Lee SY. A survey of search methodologies and automated system development for examination timetabling. J Sched 2009; 12: 55-89.

[24] Merlot LG, Boland N, Hughes B, Stuckey P. A hybrid algorithm for the examination timetabling problem. In: Burke E, Causmaecker P. editors. Practice and Theory of Automated Timetabling IV. Heidelberg, Germany: Springer, 2003, pp. 207-31.

[25] Burke EK, Newall JP. Enhancing timetable solutions with local search methods, In: Burke EK, Causmaecker P. editors. Practice and Theory of Automated Timetabling. Heidelberg, Germany: Springer, 2003. pp. 195-206.

[26] Caramia M, Dell'Olmo P, Italiano GF. New algorithms for examination timetabling. In: Naher S, Wagner D. editors. 4th International Workshop on Algorithm Engineering (WAE 2000); 5–8 September 2000; Saarbrücken, Germany. Berlin, Germany: Springer. pp. 230-242.

[27] White GM, Xie BS. Examination timetables and tabu search with longer-term memory. In: Burke EK, Erben W. editors. PATAT - 2000: LNCS Vol. 2079; 16–18 August 2000; Konstanz, Germany. Berlin, Germany: Springer. pp. 85-103.

[28] Weng FC, Asmuni HB. An automated approach based On bee swarm in tackling university examination timetabling problem. Int J Eng Comp Sci 2013; 13: 8-23.

[29] Gaspero LD, Schaerf A. Tabu search techniques for examination timetabling. In: Burke EK, Erben W, editors. Practice and Theory of Automated Timetabling III, Third International Conference - PATAT 2000, LNCS 2079; 16–18 August 2000; Konstanz, Germany. Berlin, Germany: Springer. pp. 104-117.

[30] Sabar NR, Ayob M, Kendall G. Solving examination timetabling problems using honeybee mating optimization (ETP-HBMO). In: Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009); 10–12 August 2009; Dublin, Ireland. pp. 115-129.

[31] Snedecor WG, Cochran WG. Statistical Methods. 8th ed. Ames, IA, USA: Iowa State University Press, 1989.