An SHO-based approach to timetable scheduling: a case study

Van Du Nguyen\textsuperscript{a} and Tram Nguyen\textsuperscript{a,b}

\textsuperscript{a}Faculty of Information Technology, Nong Lam University, Ho Chi Minh City, Vietnam; \textsuperscript{b}Faculty of Electrical Engineering and Computer Science, Department of Computer Science, VSB -- Technical University of Ostrava, Ostrava, Czech Republic

ABSTRACT
University timetable scheduling, which is a typical problem that all universities around the world have to face every semester, is an NP-hard problem. It is the task of allocating the right timeslots and classrooms for various courses by taking into account predefined constraints. In the current literature, many approaches have been proposed to find feasible timetables. Among others, swarm-based algorithms are promising candidates because of their effectiveness and flexibility. This paper investigates proposing an approach to university timetable scheduling using a recent novel swarm-based algorithm named Spotted Hyena Optimizer (SHO) which is inspired by the hunting behaviour of spotted hyenas. Then, a combination of SA and SHO algorithms also investigated to improve the overall performance of the proposed method. We also illustrate the proposed method on a real-world university timetabling problem in Vietnam. Experimental results have indicated the efficiency of the proposed method in comparison to other competitive metaheuristic algorithm such as PSO algorithm in finding feasible timetables.

1. Introduction
In the recent decades, the rapid development of Information and Communication Technology (ICT) has extensively impacted all sectors including education. Indeed, it has significantly facilitated the learning and teaching of students and teachers. To date, much research has investigated to find feasible solutions to some common optimization problems in the sector of education such as timetable scheduling, tasks scheduling and tests extraction using metaheuristic algorithms (Abayomi-Alli et al., 2019; Bui et al., 2018; Chu et al., 2006; Matijaš et al., 2010; Montero et al., 2011; Nguyen et al., 2019). Metaheuristic algorithms have been widely applied to solve a variety of sophisticated optimization problems in the real world because of their effectiveness and flexibility (Yang, 2020). The most popular swarm-based algorithms have emerged recently including Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995) and Ant Colony Optimization (ACO).
(ACO) (Dorigo et al., 2006), Firefly Algorithm (FA) (Yang, 2009), Bat Algorithm (BA) (Yang, 2010), Cuckoo Search (CS) (Yang & Deb, 2009). Although many publications have documented their positive roles in finding optimal solutions to a wide range of optimization problems, one optimizer algorithm cannot be effective in solving all optimization problems because each optimization problem has different nature (Wolpert & Macready, 1997). For such a motivation, much research attention investigates introducing modified versions of the standard algorithms to improve their overall performance (Ebadifard & Babamir, 2018; Fu et al., 2018; Rani et al., 2019). Besides, other research focuses on proposing novel algorithms by taking into account the interaction of individuals of a swarm and their environment (Korayem et al., 2015; Odili & Mohmad Kahar, 2016). For such a task, one can tap into the so-called swarm intelligence which considered as a specific kind of collective intelligence (Saka et al., 2013). Collective intelligence emerging from the collaboration and competition of many individuals from different sources (Pieterink & Nguyen, 2014; Sliwko & Nguyen, 2007). In Dhiman and Kumar (2017) and Dhiman and Kumar (2019), Dhiman et al. have introduced a novel swarm-based algorithm named Spotted Hyena Optimizer (SHO), which is inspired by the social relationship between spotted hyenas and their collaborative behaviour. The authors have mathematically modelled and implemented three main steps of the algorithm based on the hunting behaviour of spotted hyenas including searching for prey, encircling, and attacking prey. Referring to Dhiman and Kumar (2018), the authors have introduced a multi-objective version of SHO (called MOSHO). Accordingly, a roulette wheel mechanism is used to select optimal solutions, which is then used to simulate spotted hyenas’ social and hunting behaviours. Later, SHO algorithm has been applied to solve economic dispatch problems (for both convex and non-convex problems) (Dhiman et al., 2018); its improved version is proposed in Dhiman and Kaur (2019) by using PSO algorithm to enhance the hunting prey of spotted hyenas; Kumar et al. have proposed a binary version of SHO for dealing with discrete optimization problems (Kumar & Kaur, 2020). In Nguyen et al. (2020), a discrete version of SHO algorithm has been introduced to solve TSP. Accordingly, hyenas update their positions by applying swap sequences. A swap sequence includes a set of swap operators representing two cities can be swapped each other to generate a better solution. Experiments conducted with selected benchmark datasets from TSPLib have indicated that the proposed method can be applied to find feasible solutions to TSP.

In general, timetable scheduling is the task of allocating the right timeslots and classrooms for various courses by taking into account predefined constraints. Referring to Vriealink et al. (2019), it is a quite challenging and difficult task because of taking into consideration a number of constraints in the timetable scheduling process. Roughly speaking, this kind of task can be done manually or automatically using scheduling programs. However, a manual process takes a lot of effort and requires numbers of verification by lecturers, supervisors, etc. before it can be approved, especially with a large number of students as timetable scheduling at universities. With the later one, many approaches have been investigated to find a feasible timetable such as GA-based (Abdelhalim & El Khayat, 2016; Arogundade et al., 2010; Jat & Yang, 2009; Nguyen, 2007; Nguyen et al., 2011; Sigl et al., 2003), PSO-based (Ahmad & Shaari, 2016; Foong & Rahim, 2013; Ho et al., 2009; Nguyen et al., 2019; Tassopoulos & Beligiannis, 2012) and SA-based methods (Basir et al., 2013; Goh et al., 2019). With timetable scheduling problem, the constraints are often classified into two broad categories: hard and soft constraints. The first kind of
constraints, as its name implies, is used for representing the constraints in which a feasible timetable must be satisfied (Henry Obit, 2010). Meanwhile, the second kind of constraints stands for the constraints that a feasible timetable could be violated (Katsaragakis et al., 2015). This kind of constraints is utilized to measure the quality of a timetable. The lower the violation of soft constraints, the better the feasible timetable.

Although many research results have been investigated to solve timetable scheduling problem, it is difficult to propose an approach that can be applicable for all universities because universities have their own constraints in the scheduling process. Therefore, in this paper we first investigate proposing an SHO-based approach to timetable scheduling problem. Then, a combination of SA and SHO algorithms also investigated to improve the overall performance of the proposed method. Simulated Annealing (SA) is a probabilistic single-solution-based search method, which is inspired by the process of annealing in metallurgy (Ralston et al., 2000), in which the temperature is carefully controlled to decrease slowly so that the material reaches a state with minimum free energy. One of the advantages of SA over other methods is the ability to avoid being trapped in local minima by accepting a worse solution with a probability dependent on the temperature and energies of current and neighbour states. Experiments are conducted with the real-world data at the faculty of Information Technology, Nong Lam University, Vietnam. In addition, we compare the proposed method to other competitive metaheuristic algorithms such as traditional SHO and PSO in finding feasible timetables. To the best of our knowledge, no work on the use of SHO algorithm to solve timetable scheduling problem has been investigated so far.

The rest of the paper is organized as follows. Section 2 presents a brief review of some related works. Next, we introduce the problem statement of timetable scheduling, objective function, SHO-based approach and the combination of SA and SHO algorithms to solve the timetable scheduling problem. Section 4 provides the experimental results as well as their analysis. In addition, we compare the obtained results to PSO algorithm. The paper is ended with some conclusions and future research.

2. Related works

In recent years, metaheuristic algorithms have attracted much research attention. Indeed, much research focused on the use of these algorithms to solve common optimization problems (Bui et al., 2018; Nguyen et al., 2019). According to Dhiman and Kumar (2017), they are classified into three categories such as evolutionary-based, physical-based, swarm-based algorithms. This section presents a brief summary on the use of metaheuristic algorithms to solve timetable scheduling problems.

One of the most popular evolutionary-based algorithm has been applied widely to find a feasible timetable is GA. Referring to Sigl et al. (2003), the authors have presented a modification of basic genetic operators of GA for solving timetable scheduling problem. Accordingly, the achieved results converge much faster than the basic algorithm and represent a good starting point for complete solving of the full-scale problem. However, to completely solve the full-scale problem, further improvement of algorithms will have to be made. In Nguyen et al. (2007), the authors proposed a hybrid method by combining GA and Greedy algorithms to solve the timetable scheduling problem at Nong Lam University. Referring to Jat and Yang (2009), the authors presented a lot of
techniques such as a Memetic Algorithm (MA), a novel Guided Search (GS) strategy to combine genetic algorithm for course timetabling problems in a university. The experimental results indicated that the combination of GS and LS (Local Search) strategies significantly improves the performance of GAs. In Arogundade et al. (2010), the author developed a GA-based approach to a real-world university examination timetabling problem. The achieved results demonstrated that GA is useful in expressing constraint and provides an intuitive solution to the constraints. In Nguyen et al. (2011), the authors have presented a new hybrid GA-Bees algorithm for solving a real-world university timetabling problem in Vietnam. The obtained results confirm that the proposed method is able to produce better high-quality solutions to the university course timetabling problem. In Abdelhalim and El Khayat (2016), authors have introduced a modified version of standard GA by applying some heuristics to initialize population having feasible good quality individuals. Later, in order to reduce waste, a crossover type focusing on the utilization rates of learning spaces is used for crossover and a local search heuristic is employed for mutation. Meanwhile, space utilization, gaps between events and a maximum number of lectures per day are taken into account for measuring fitness values of timetables. Experimental results with a large dataset from the Faculty of Commerce, Alexandria University in Egypt have indicated that the proposed method is an effective tool for managing timetables and resources in universities.

SA (physical-based algorithm) is a probabilistic single-solution-based search method, which is inspired by the process of annealing in metallurgy (Ralston et al., 2000) in which the temperature is carefully controlled to decrease slowly so that the material reaches a state with minimum free energy. One of the advantages of SA over other methods is the ability to avoid being trapped in local minima by accepting a worse solution with a probability dependent on the temperature and energies of current and neighbour states. The efficiency of SA algorithm has been documented in many publications (Abdolhosseinzadeh & Alipour, 2020; Basir et al., 2013; Grabusts et al., 2019). Referring to Goh et al. (2019), the authors have proposed an improved version of SA algorithm with reheating mechanism (named SAR) to solve timetable scheduling problems. Concretely, a reinforcement learning-based method has been applied to enhance the search operation. Experimental results have showed that the proposed method is better than other metaheuristic algorithms in finding feasible timetables. In Basir et al. (2013), the authors investigate to solve university timetable scheduling using SA algorithm by taking into account various soft and hard constraints such as a number of subjects, time-slots, classrooms, students and workloads. Tested results with the real-world data have indicated the efficiency of the proposed method in solving timetable scheduling problems.

The most popular swarm-based algorithm is PSO, which is inspired by the social behaviour of bird flocking or fish school (Kennedy & Eberhart, 1995). In the current literature, PSO has been widely applied to find optimal solutions to various optimization problems because of its fast convergence and fewer parameter settings. The standard PSO algorithm has been applied to solve many continuous optimization problems. Referring to Chu et al. (2006), an discrete version of PSO has been proposed to solve timetable scheduling problem. Experimental results have indicated that PSO can also be applied to solve discrete optimization problem with promising result. Ho et al. (2009) have introduced a hybrid approach using PSO and local search to solve timetable scheduling problems.
Concretely, courses are scheduled by using PSO algorithm. Meanwhile, local search is used to search the nearest neighbour timeslot and classroom when clashes occur. By means of experiments, the proposed method is better than other competitive algorithms in terms of the quality of obtained timetables as well as computational time. Foong and Rahim (2013) have presented an application of PSO algorithm to timetable scheduling problem. Experimental results have revealed that the obtained timetable satisfies all pre-defined hard constraints and the violation of soft constraints is minimal. In Tassopoulos and Beligiannis (2012), the authors have proposed a new PSO-based hybrid adaptive algorithm to find feasible and efficient timetables for high schools. By means of experiments with datasets from different high schools, the proposed method is better than other competitive algorithms in finding optimal timetables. Ahmad and Shaari (2016) have proposed a PSO-based approach to generate an exam timetable. Experiments with real-world data have indicated that the proposed method can be applied to solve exam timetable scheduling problem.

As aforementioned, SHO is a recent novel swarm-based algorithm which developed to solve engineering optimization problem (Dhiman & Kumar, 2017, 2019). The algorithm is inspired by the collaborative behaviour of spotted hyenas in hunting preys. Accordingly, three main steps of SHO are searching for prey, encircling, and attacking prey are mathematically modelled and implemented. In these works, the algorithms have applied to find optimal solutions to numerical or function optimization problems. Accordingly, experiments carried on 29 well-known benchmark test functions have revealed the efficiency of the algorithm in comparison to other metaheuristic algorithms (i.e. PSO, GA, GWO, MVO, SCA, GSA and HS). In Dhiman and Kumar (2018), the authors have proposed a multi-objective version of SHO algorithm (named MOSHO). With this approach, the roulette wheel mechanism is applied to simulate the social and hunting behaviours of spotted hyenas. By means of experiments carried on 24 benchmark test functions, MOSHO is better than other metaheuristic algorithms in term of obtained solutions as well as convergence rate.

An improved version of SHO algorithm has been formulated in Dhiman and Kaur (2019) by using PSO algorithm to enhance the hunting behaviour of spotted hyenas. In Kumar and Kaur (2020), a binary version of SHO has been introduced to solve discrete optimization problems. Accordingly, the continuous positions are discretized using the hyperbolic tangent function. Experimental results have indicated that the modified version is a suitable technique not only for test functions but also for feature selection domain. In addition, the obtained results are better than or equal to other competitive metaheuristic techniques. Apart from these, the research results in Dhiman et al. (2018) have indicated that SHO algorithm is also suitable for solving economic dispatch problems (for both convex and non-convex). Referring to Nguyen et al. (2020), a modified version of SHO algorithm has been introduced to solve TSP. Accordingly, we have modified three main steps of SHO algorithm which were mathematically modelled in Dhiman and Kumar (2017) and Dhiman and Kumar (2019). For such a task, the concepts of swap sequence and swap operator used in Sopto et al. (2018) are employed for updating positions of spotted hyenas toward the position of the best individual of the group. Experiments carried on four benchmark datasets in TSPLib (Burma14, Bays29, Att48, and Berlin52) have indicated the efficiency and effectiveness of the modified version in finding feasible solutions to TSP.
From the above analysis, it can be seen that the standard SHO algorithm is developed for solving numerical optimization problems. Although many research results have been investigated to solve timetable scheduling problem, it is difficult to propose an approach that can be applicable for all universities because universities have their own constraints in the scheduling process. Therefore, in this paper we investigate proposing an approach to solve the timetable scheduling problem using SHO algorithm. This is a discrete version of SHO algorithm. Besides, a combination of SA and SHO algorithms is also introduced to improve the overall performance of the standard SHO algorithm. Furthermore, the proposed methods are compared to other metaheuristic algorithm such as PSO using a real-world dataset. To the best of our knowledge, no work on the use of SHO algorithm to solve timetable scheduling problem has been investigated so far.

3. The proposed method

3.1. Problem description

As aforementioned, university timetable scheduling is the task of allocating the right timeslots and classrooms for various courses by taking into account predefined constraints. In fact, it is difficult to propose an approach that can be used for all universities because universities have their own constraints in the scheduling process. In this paper, we take into consideration the timetable scheduling at Nong Lam University (NLU) as a case study. The general process of timetable scheduling is described in Figure 1.

According to Figure 1, a course is a group of lectures (each lecture places on one timeslot per week) that have the same lecturer, the same attending classes and subject. Each course is opened for a one or some attending classes, taught by a lecturer. Courses include additional information concerning the type of the course, the subject which
the course belongs to. In the scheduling process, a course has one lecture in a week and each lecture takes 3 periods (1 timeslot). In addition, the input of the timetable scheduling program is a list of courses including lecturers and classes. Therefore, we do not take into consideration the assignment of lecturers, classes into courses.

The number of timeslots of each day is four and each timeslot lasts 3 consecutive periods. For each day, the first period begins at 07:00 AM and the last period ends at 05:00 PM. In particular, the first timeslot begins at the first period meanwhile the second timeslot begins at the fourth period. The detail of timeslots is described in Table 1.

In addition, we consider two kinds of courses: theoretical and practical courses. Courses must be scheduled to appropriate classrooms. In other words, assigned classrooms for practical courses must be into practical classrooms (similar for theoretical courses) (Table 2).

The process of timetable scheduling in this work is based on a 3D array. These dimensions represent the days of the week, classrooms and timeslots. Therefore, each course is assigned in an appropriate day, classroom and timeslot. The detail timetable of these course is described in Figure 2.

According to Figure 2, Course 1 (theoretical course) is placed on Monday at the first timeslot and the classroom is CT102. Meanwhile, two practical courses (Course 2 and Course 3) are placed on Tuesday at the second and the third timeslots with the same classroom (PM1).

3.1.1. Hard constraints
As aforementioned, hard constraints are the ones that a feasible timetable must be fulfilled. In the proposed method, we assumed that the initialized individuals in a population are satisfied all hard constraints. Then, the aim of optimization process is to minimize the soft constraints violation.

In this paper, we consider the following hard constraints:

- $HC_1$: Each course should be absolutely placed in a timeslot.
- $HC_2$: Lecturers, classes, and classrooms could not be assigned to periods at which they’re not available.

| Table 1. Timeslots. |
|---------------------|
| Timeslot | Period | Begin | End |
| 1          | 1 → 3  | 07:00 | 09:15 |
| 2          | 4 → 6  | 09:30 | 11:45 |
| 3          | 7 → 9  | 12:15 | 14:30 |
| 4          | 10 → 12| 14:45 | 17:00 |

| Table 2. A sample of courses. |
|-------------------------------|
| Course 1 | Course 2 | Course 3 |
| Lecturer: LPHung | Lecturer: LPHung | Lecturer: LPHung |
| Subject: Basic Programming | Subject: Basic Programming | Subject: Basic Programming |
| Type: Theoretical | Type: Practical | Type: Practical |
| Class: DH18DTA | Class: DH18DTA | Class: DH18DTA |
3.1.2. Soft constraints

In contrary to hard constraints, a feasible timetable could violate soft constraints. However, the less the soft constraints violation, the better the feasible timetable. In practice, this kind of constraints is often utilized to measure the quality of a feasible timetable. In addition, each soft constraint is assigned a weighted value representing its importance value in comparison to other soft constraints. Our objective is to try minimizing the soft constraints violation.

In this paper, the following soft constraints are taken into consideration:

- *SC₁*: The number of days that a lecturer was assigned to should be as small as possible.
- *SC₂*: Courses should not be assigned in busy timeslots of lecturers who teach these courses.
- *SC₃*: Courses should not be assigned in busy timeslots of rooms.
- *SC₄*: The distance a lecturer walks in a day should be minimized.
- *SC₅*: Lecturers should not teach two consecutive theoretical courses a day.
- *SC₆*: Assigned timeslots on weekend should be avoided.
- *SC₇*: The number of timeslots that a teacher is assigned in a day should be less than 4.
- *SC₈*: The idle time between consecutive courses in the same day for a lecturer or a class should be avoided.
- *SC₉*: The number of students of each course is higher 80% the capacity of the assigned classroom.

![Figure 2. Detail timetable of Basic Programming courses.](image-url)
### 3.1.3. Objective function

In this paper, soft constraints are used for designing the objective function of timetable scheduling problem. In addition, each soft constraint can be associated with a weighted value representing its importance in comparison to other constraints. Concretely, for a given set of soft constraints \( SC = \{ SC_1, SC_2, SC_3, \ldots, SC_n \} \), the objective function is described as follows:

\[
\mu = \sum_{i=1}^{N} w_i f(s_i) \tag{1}
\]

where \( w_i \) represents the importance level of soft constraint \( s_i \), \( f(s_i) \) represents the violation number of constraint \( s_i \), \( N \) presents the total number of soft constraints.

The main objective of timetable scheduling is to find a timetable so that the value of \( \mu \) to be minimal. In other words, the smaller the value of \( \mu \), the better the quality of feasible timetable.

**Example 3.1** For a given set of soft constraints and their importance values are described in the following table. We compute fitness values of timetables based on the soft constraints violation.

According to Table 3, the fitness values of timetables T1 and T2 are \( \mu_{T1} = 0.5 \times 3 + 0.3 \times 5 + 0.6 \times 4 + 0.1 \times 6 = 4.0 \) and \( \mu_{T2} = 0.5 \times 2 + 0.3 \times 3 + 0.6 \times 1 + 0.1 \times 1 = 2.6 \), respectively. Therefore, timetable T2 is better than timetable T1 in terms of fitness value.

### 3.2. SHO-based algorithm to timetable scheduling

In this section, we introduce a modified version of SHO algorithm to timetable scheduling. Concretely, we present the main components of SHO algorithm: encircling prey, hunting prey and attacking prey. Accordingly, we have modified three main steps of SHO algorithm which were mathematically modelled in Dhiman and Kumar (2017) and Dhiman and Kumar (2019). For such a task, the concepts of swap sequence and swap operator used in Sooto et al. (2018) is employed.

Let \( SS = \{ SO_1, SO_2, SO_3, \ldots, SO_n \} \) be a swap sequence including \( n \) swap operators, where \( SO_i \) represents two positions in timetables that might be swapped to get a new timetable.

**Example 3.2** For given two timetables T1 and T2, SS is determined based on the number of \( SO_i \) in which timetable T2 should be swapped to get T1. Each swap operator is characterized by the positions of two courses in two timetables that can be swapped to each other. Each position including three indices \( i, j, k \) (\( j \) represents a day of

| Soft constraints \((s_i)\) | \( SC_1 \) | \( SC_2 \) | \( SC_3 \) | \( SC_4 \) |
|--------------------------|--------|--------|--------|--------|
| Weighted values \((w_i)\)| 0.5    | 0.3    | 0.6    | 0.1    |
| \( f(s_i) \)_{T1}           | 3      | 5      | 4      | 6      |
| \( f(s_i) \)_{T2}           | 2      | 3      | 1      | 1      |
the week, \(j\) the classroom, \(k\) the timeslot of an assigned course). In addition, the determination of SS is described as follows.

\[
SS = T_1 - T_2 = (SO_1, SO_2, \ldots, SO_n)
\]

where the notion ‘–’ represents the generation of SS with one or more SOs to get \(T_2\) from \(T_1\).

Suppose with \(SO_1\), the first position is a course \((c_1)\) placed at day 1, room 2 and timeslot 2, called \(p_1 = (1, 2, 2)\) and the second position is a course \((c_2)\) placed at day 2, room 4 and timeslot 1, called \(p_2 = (2, 4, 1)\). Then the new timetable is generated by applying \(SO_1\) to the first timetable to change the position of the course \((c_1)\) to position \(p_2\) day 2, room 4 and timeslot 1. The main steps of SHO algorithms are modified as follows:

- **Encircling prey:** In this step, since the search space is not known in advance, we suppose the current best solution is the target prey. When the target prey is identified, other hyenas in the group of hyenas will try to update their positions toward the position of the prey.

\[
D_h = B.(P_p(i) - P(i))
\]

\[
P(i + 1) = P_p(i) + E.D_h
\]

where \(D_h\) presents a swap sequence including a number of swap operators that an hyenas might be applied to get the prey, \(i\) represent the current iteration, \(B\) and \(E\) are coefficient values. \(P_p\) represents the position of the prey, \(P\) represents the position of hyenas. \(B\) and \(E\) are computed as follows:

\[
B = 2.\text{rd}_1
\]

\[
E = 2.\text{h} \cdot \text{rd}_2 - \text{h}
\]

\[
\text{h} = 5 - (i \times (5/\text{Maxiteration}))
\]

where \(\text{rd}_1\) and \(\text{rd}_2\) are random values belonging to \([0, 1]\). The value of \(\text{h}\) belongs to \([0, 5]\), which is linearly decreased from 5 to 0 according to \(\text{Maxiteration}\).

In Equation (3) the notion ‘+’ indicates the application of SS found in Equation (2). However, not all SOs in \((P_p(i)-P(i))\) are applied (only \(E.(P_p(i) - P(i))\)).

- **Hunting prey:** To mathematically modelling this step, we suppose that the best individual who has knowledge about the location of the prey and others try to update their positions based on the best solutions obtained so far. Concretely, consider the following formulas:

\[
D_h = B.(P_h - P(k))
\]

\[
P(k) = P_h + E.D_h
\]

\[
C_h = \{P_k, P_{k+1}, \ldots, P_{k+N}\}
\]

where \(P_h\) represents the position of the first best spotted hyenas, \(P_k\) represents the positions of other hyenas, \(N\) represents the number of spotted hyenas in the group and is calculated based on the following equation:

\[
N = \text{countnos}(P_h, P_{h+1}, \ldots, P_{k+N})
\]
where $M \in \mathbb{N}$, $nos$ represents the number of solutions, $C_h$ represents the group of optimal solutions.

- **Attacking prey:** Attacking the prey is mathematically modelled as follows: the value of $h$ will be decreased from 5 to 0 over the iterations. The variation in values of $E$ is also decreased to change the value of $h$. The best solution $P(i+1)$ is selected from $C_h$ and the positions of other spotted hyenas will be updated based on the position of the best search agent (the closest spotted hyenas to the target prey).

The flowchart of the algorithm is described in **Figure 3**.

![Flowchart of SHO algorithm](image-url)
According to Figure 3, each individual in the population is considered as a solution (a feasible timetable). On the basis of these individuals, the next step is to optimize their soft constraints violation by means of updating their positions with respect to the best solution. For such a task, we determine a swap sequence including swap operators as presented in the previous section.

The pseudo code of the proposed method is described in Algorithm 1.

**Algorithm 1** SHO algorithm for timetable scheduling

1: **Input:** Input of timetable requirements
2: **Output:** optimal timetable.
3: **BEGIN**
4: **Step 1:** Initialize population $P = \{P_1, P_2, P_3, \ldots, P_n\}$.
5: Generate a prey – a feasible timetable having fitness value less than fitness values of all individuals in the population.
6: **Step 2:** Compute the fitness values of individuals in the population
7: **Step 3:** Determine the best individual from the population ($P_b$).
8: **Step 4:** Determine the group of best solutions ($C$) based on the current best individual ($P_b$).
9: **Step 5:** Update the position of each individual (by changing day, classroom, and timeslot) with respect to the best individual ($P_b$). Update solutions violating hard constraints (if any).
10: **Step 6:** Compute the fitness values of individuals and update $P_b$ if there exits another better solution.
11: **Step 7:** Update $C_b$ based on the best individual ($P_b$).
12: **Step 8:** If stop condition is satisfied, return the best solution. Else, go to Step 4
13: **END**

The stop criterion of the proposed method is based on the number of iterations.

### 3.3. SHO-SA-based algorithm to timetable scheduling

As aforementioned, SA is a probabilistic single-solution-based search method, which is inspired by the process of annealing in metallurgy (Ralston et al., 2000). In particular, the temperature is carefully controlled to decrease slowly so that the material reaches a state with minimum free energy. The common problem of Local search algorithms is trapped into local minima. However, SA can overcome the obstacle by accepting worse solution with a probability dependent on the temperature and energies of current and neighbour states. In this paper, SA is used to optimize timetables after their positions are updated. The main aim of this step is to reduce the running time of SHO-based algorithm.

Concretely, the pseudo code of the proposed method is described in Algorithm 2.

**Algorithm 2** The combination of SHO and SA algorithms

1: **Input:** list of timetables
2: **Output:** the best optimal timetable.
3: **BEGIN**
4: Initialize the parameters
5: Calculate the fitness of each timetable
6: $P_n$ – a timetable having smallest fitness value
7: $C_n$ - the group or cluster of all optimal solutions
8: **while** $x < $ MaxIterations **do**
9: **for** each timetable $i$ **do**
10: Update the position of current timetable
11: Check if timetable violates the hard constraints then adjust it
12: Optimize timetable using SA algorithm
13: **end for**
14: Update parameters
15: Calculate the fitness of each timetable
16: Update $P_n$ if there is a better solution than previous optimal solution
17: **end while**

The stop criterion of the proposed method is based on the number of iterations.
The flowchart of the algorithm is described in Figure 4.

**Figure 4.** Flowchart of SHO-SA algorithm.
4. Experimental results and their evaluation

4.1. Settings

The experiments are conducted with the real dataset at Faculty of Information Technology, NLU, Vietnam. The number of courses is 108 (45 theoretical and 63 practical courses). The number of laboratories, which is used for scheduling practical courses, is four (PM1, PM2, PM3, PM4). Theoretical classrooms are shared with all faculties in NLU. Algorithms are implemented on a computer with the following configurations: Intel Core (TM) i7-5500U CPU 2.40 GHz/8GB RAM Laptop. The parameters of selected metaheuristic algorithms are described in Table 4.

In addition, the weighted values of soft constraints are assigned in Table 5.

4.2. Experimental results

Experimental results are described in Table 6 with settings described in Tables 4 and 5. Accordingly, SHO-SA approach generates feasible timetables having smaller fitness values than other approaches for both best and average cases. In addition, SHO-based approach is better than PSO-based approach in generating feasible timetables. Notice that, fitness values are measured using soft constraints and their weighted values. We performed 50 runs for each setting and recorded the best and average (AVG) values are selected from these 50 corresponding fitness values (Figure 5).

Apart from fitness values, we also record the running time of each algorithm. The average running times are described in Table 7 as follows:

According to Figure 6, PSO algorithm outperforms both SHO and SHO-SA algorithms in terms of running time. The combination of SHO and SA algorithms is helpful in reducing the running time of the traditional SHO algorithm.

5. Conclusion and future works

Recently, metaheuristic algorithms have attracted much research attention. Indeed, much research focused on the use of these algorithms to solve common optimization problems (Bui et al., 2018; Nguyen et al., 2019). To date, many publications have documented their positive roles in finding optimal solutions to a wide range of optimization problems such as extracting multiple-choice tests (Bui et al., 2018; Nguyen et al., 2019), timetable scheduling (Abayomi-Alli et al., 2019; Chu et al., 2006; Matijaš et al., 2010; Montero et al., 2011), vehicle routing and scheduling (Mirjalili et al., 2020), travelling salesman problem (TSP) (Chaudhari & Thakkar, 2019; Panda, 2018). Timetable scheduling problem is an NP-Hard problem and generating feasible timetables is a crucial task that many universities

| Table 4. Experimental parameters. |
|-----------------------------------|
| Parameter                        | SA-SHO | PSO  | SHO |
| Population                       | 40      | 40   | 40  |
| Iterations                       | 1000    | 1000 | 1000|
| Ω                                | --      | 0.5  | --  |
| Total number of runs             | 50      | 50   | 50  |
|                                  | – Not Applicable |
have to face every semester. This kind of task requires time and effort of the involved personnel. In this paper, we have proposed methods for solving timetable scheduling based on SA and SHO algorithms. Experimental results with real dataset at the Faculty of Information Technology, Nong Lam University, Ho Chi Minh City have indicated the efficiency of the proposed method in comparison to other competitive metaheuristic algorithm PSO algorithm. Concretely, the hybrid approach SHO-SA outperforms other approaches in finding feasible timetables in terms of the quality of obtained results. In addition, it is helpful in reducing the running time of the traditional SHO algorithm in solving timetable scheduling problem.

### Table 5. Weights of soft constraints.

| SC₁ | SC₂ | SC₃ | SC₄ | SC₅ | SC₆ | SC₇ | SC₈ | SC₉ |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Weights | 10   | 20   | 20   | 20   | 10   | 20   | 100  | 30   | 10   |

### Table 6. Experimental results.

| Algorithm | Best | AVG |
|-----------|------|-----|
| SHO       | 120  | 237.4 |
| SHO-SA    | 100  | 180  |
| PSO       | 250  | 320  |

### Table 7. Experimental results.

| Algorithms | Running time (minutes) |
|------------|------------------------|
| SHO        | 12.60                  |
| SHO-SA     | 11.40                  |
| PSO        | 10.33                  |

![Figure 5. Performance of algorithms.](image-url)
For future work, we intend to propose a hybrid method by combining SHO algorithm and other metaheuristic algorithms to improve the overall performance of the algorithm. Experiments will be conducted with larger datasets to evaluate the efficiency of the proposed method in terms of running time, convergence rate. Apart from these, we also take into consideration proposing parallel approach to deal with larger datasets.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

This research is supported by project CS-CB19-CNTT-01.

**Notes on contributor**

*Van Du Nguyen* received the MSc. degree in Computer Science from Wroclaw University of Science and Technology, Poland and received the Ph.D. degree in Computer Science from Wroclaw University of Science and Technology, Poland. His scientific interests consist of collective intelligence, swarm intelligence, consensus theory, and inconsistent knowledge processing. He had more than 30 publications.

*Tram Nguyen* received her M.Sc. degree in Computer Science from University of Information Technology–Vietnam National University HCMC in 2013. She is currently a Ph.D. student at VŠB-Technical University of Ostrava, Czech Republic. Her research interests include metaheuristic algorithms, particle swarm optimization, genetic algorithm, single-objective/multi-objective optimization problems, grid computing, parallel computing.
References
Abayomi-Alli, O., Abayomi-Alli, A., Misra, S., Damasevicius, R., & Maskeliunas, R. (2019). Automatic examination timetable scheduling using particle swarm optimization and local search algorithm. In Data, engineering and applications (pp. 119–130). Springer.
Abdelhalim, E. A., & El Khayat, G. A. (2016). A utilization-based genetic algorithm for solving the university timetabling problem (uga). Alexandria Engineering Journal, 55(02), 1395–1409. https://doi.org/10.1016/j.aej.2016.02.017
Abdolhosseinzadeh, M., & Alipour, M. M. (2020). A simulated annealing algorithm for the restricted stochastic traveling salesman problem with exponentially distributed arc lengths. Journal of Mathematical Modeling, 08(3), 279–290. https://doi.org/10.22124/jmmm.2020.15535.1378
Ahmad, A., & Shaari, F. (2016). Solving university/polytechnics exam timetable problem using particle swarm optimization. In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication (pp. 1–4). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/2857546.2857593.
Arogundade, O. T., Akinwale, A. T., & Aweda, O. M. (2010). A genetic algorithm approach for a real-world university examination timetabling problem. International Journal of Computer Applications, 12(5), 0975–8887. https://doi.org/10.5120/ijca
Basir, N., Ismail, W., & Norwawi, N. M. (2013). A simulated annealing for Tahmidi course timetabling. Procedia Technology, 11(2), 437–445. https://doi.org/10.1016/j.protcy.2013.12.213
Bui, T., Nguyen, T., Vo, B., Nguyen, T., Pedrycz, W., & Snasel, V. (2018). Application of particle swarm optimization to create multiple-choice tests. Journal of Information Science and Engineering, 34(6), 1405–1423. https://doi.org/10.6688/JISE.201811_34(6).0004
Chaudhari, K., & Thakkar, A. (2019). Travelling salesman problem: An empirical comparison between ACO, PSO, ABC, FA and GA. In Proceedings of emerging research in computing 2019, information, communication and applications (pp. 397–405). Springer.
Chu, S. C., Chen, Y. T., & Ho, J. H. (2006). Timetable scheduling using particle swarm optimization. In Proceedings of the first international conference on innovative computing, information and control (Vol. 3, pp. 324–327). IEEE. https://doi.org/10.1109/ICICIC.2006.541
Dhiman, G., Guo, S., & Kaur, S. (2018). ED-SHO: A framework for solving nonlinear economic load power dispatch problem using spotted hyena optimizer. Modern Physics Letters A, 33(40), 1850239. https://doi.org/10.1142/S0217732318502395
Dhiman, G., & Kaur, A. (2019). A hybrid algorithm based on particle swarm and spotted hyena optimizer for global optimization. In Proceedings of soft computing for problem solving (pp. 599–615). Springer.
Dhiman, G., & Kumar, V. (2017). Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Advances in Engineering Software, 114, 48–70. https://doi.org/10.1016/j.advengsoft.2017.05.014
Dhiman, G., & Kumar, V. (2018). Multi-objective spotted hyena optimizer: a multi-objective optimization algorithm for engineering problems. Knowledge-Based Systems, 150, 175–197. https://doi.org/10.1016/j.knosys.2018.03.011
Dhiman, G., & Kumar, V. (2019). Spotted hyena optimizer for solving complex and non-linear constrained engineering problems. In Proceedings of harmony search and nature inspired optimization algorithms (pp. 857–867). Springer.
Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39. https://doi.org/10.1109/CI-M.2006.248054
Ebadi, F., & Babamir, S. M. (2018). A PSO–based task scheduling algorithm improved using a load–balancing technique for the cloud computing environment. Concurrency and Computation: Practice and Experience, 30(12), e4368. https://doi.org/10.1002/cpe.v30.12
Foong, O. M., & Rahim, S. B. (2013). Particle swarm inspired timetabling for ICT courses. In Applied mechanics and materials (Vol. 263, pp. 2138–2145). Trans Tech Publications Ltd. https://doi.org/10.4028/www.scientific.net/AMM.263-266.2138.
Fu, C., Zhang, L., Wang, X., & Qiao, L. (2018). Solving TSP problem with improved genetic algorithm. In Proceedings of AIP conference (Vol. 1967, No. 01, p. 040057). AIP Publishing LLC. https://doi.org/10.1063/1.5039131.

Goh, S. L., Kendall, G., & Sabar, N. R. (2019). Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem. Journal of the Operational Research Society, 70(6), 873–888. https://doi.org/10.1080/01605682.2018.1468862

Grabusts, P., Musatovs, J., & Golenkov, V. (2019). The application of simulated annealing method for optimal route detection between objects. Procedia Computer Science, 149(6), 95–101. https://doi.org/10.1016/j.procs.2019.01.112

Henry Obit, J. (2010). Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems [Doctoral dissertation]. University of Nottingham.

Ho, I. S. F., Safaai, D., & Zaiton, M. H. S. (2009). A combination of PSO and local search in university course timetabling problem. In Proceedings of international conference on computer engineering and technology 2009 (Vol. 02, pp. 492–495). IEEE. https://doi.org/10.1109/ICCET.2009.188.

Jat, S. N., & Yang, S. (2009). A guided search genetic algorithm for the university course timetabling problem. MISTA 2009.

Katsaragakis, I. V., Tassopoulos, I. X., & Beligiannis, G. N. (2015). A comparative study of modern heuristics on the school timetabling problem. Algorithms, 8(3), 723–742. https://doi.org/10.3390/a8030723

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN’95-international conference on neural networks (Vol. 4, pp. 1942–1948). https://doi.org/10.1109/ICNN.1995.488968.

Korayem, L., Khorsid, M., & Kassem, S. S. (2015). Using grey wolf algorithm to solve the capacitated vehicle routing problem. In Proceedings of IOP conference series: materials science and engineering (Vol. 83, No. 01, p. 012014). IOP Publishing. https://doi.org/10.1088/1757-899x/83/1/012014.

Kumar, V., & Kaur, A. (2020). Binary spotted hyena optimizer and its application to feature selection. Journal of Ambient Intelligence and Humanized Computing, 11(7), 2625–2645. https://doi.org/10.1007/s12652-019-01324-z

Matijaš, V. D., Molnar, G., Ćupić, M., Jakobović, D., & Bašić, B. D. (2010). University course timetabling using ACO: A case study on laboratory exercises. In Proceedings of international conference on BT-knowledge-based and intelligent information and engineering systems (pp. 100–110). Berlin: Springer.

Mirjalili, S., Dong, J. S., & Lewis, A. (2020). Ant colony optimizer: theory, literature review, and application in AUV path planning. Nature-Inspired Optimizers, 811, 7–21. https://doi.org/10.1007/978-3-030-12127-3

Montero, E., Riff, M. C., & Altamirano, L. (2011). A psO algorithm to solve a real course+ exam timetabling problem. In Proceedings of international conference on swarm intelligence (Vol. 24, pp. 1–8). Springer-Nature. https://doi.org/10.1109/ICICIC.2006.541

Nguyen, D. T. (2007). Solving timetabling problem using genetic and heuristic algorithms. In Proceedings of eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing (SNPD 2007) (Vol. 3, pp. 472–477). IEEE. https://doi.org/10.1109/SNPD.2007.464.

Nguyen, T., Bui, T., & Vo, B. (2019). Multi-swarm single-objective particle swarm optimization to extract multiple-choice tests. Vietnam Journal of Computer Science, 6(02), 147–161. https://doi.org/10.1142/S21968881950009X

Nguyen, D. T., Nguyen, C. T., Luong, T. L., Le, T. N. T., & Le, K. T. T. (2007). An application of genetic algorithm for weekly course timetabling problem. In Proceedings of international conference on artificial intelligence (Vol. II, pp. 455–462). ICAI.

Nguyen, V. D., Nguyen, T., Nguyen, T. L., Tran, V. C., & Truong, H. B. (2020). Spotted hyena optimizer: An approach to travelling salesman problems. In Proceedings of international conference on computational collective intelligence (pp. 217–228). Springer. https://doi.org/10.1007/978-3-030-63007-2_17.

Nguyen, B. P., Nguyen, T. T. M. K., & Tran, T. H. N. (2011). A new hybrid GA-Bees algorithm for a real-world university timetabling problem. In Proceedings of international conference on intelligent
Odili, J. B., & Mohmad Kahar, M. N. (2016). Solving the traveling salesman’s problem using the african buffalo optimization. *Computational Intelligence and Neuroscience, 2016*, 1510256:1–1510256:12. https://doi.org/10.1109/ICBMI.2016.1510256

Panda, M. (2018). Performance comparison of genetic algorithm, particle swarm optimization and simulated annealing applied to TSP. *International Journal of Applied Engineering Research, 13* (9), 6808–6816.

Pietranik, M., & Nguyen, N. T. (2014). A multi-attribute based framework for ontology aligning. *Neurocomputing, 146*(2), 276–290. https://doi.org/10.1016/j.neucom.2014.03.067

Ralston, A., Reilly, E., & Hemmendinger, D. (2000). *User interfaces. encyclopedia of computer science* (4th ed.). Nature Publishing Group.

Rani, S., Suri, B., & Goyal, R. (2019). On the effectiveness of using elitist genetic algorithm in mutation testing. *Symmetry, 11*(9), 1145. https://doi.org/10.3390/sym11091145

Saka, M. P., Doğan, E., & Aydogdu, İ. (2013). Analysis of swarm intelligence-based algorithms for constrained optimization. In *Swarm intelligence and bio-inspired computation* (pp. 25–48). Elsevier.

Sigl, B., Golub, M., & Mornar, V. (2003). Solving timetable scheduling problem using genetic algorithms. In *Proceedings of the 25th international conference on information technology interfaces, 2003. ITI 2003* (pp. 519–524). IEEE. https://doi.org/10.1109/ITI.2003.1225396.

Sliwko, L., & Nguyen, N. T. (2007). Using multi-agent systems and consensus methods for information retrieval in internet. *Proceedings of International Journal of Intelligent Information and Database Systems 2007, 1*(2), 181–198. https://doi.org/10.1504/IJIIDS.2007.014949

Sopto, D. S., Ayon, S. I., Akhand, M. A. H., & Siddique, N. (2018). Modified grey wolf optimization to solve traveling salesman problem. In *Proceedings of international conference on innovation in engineering and technology (ICIET) 2018* (pp. 1–4). IEEE. https://doi.org/10.1109/CIET.2018.8660872.

Tassopoulos, I. X., & Beligiannis, G. N. (2012). Using particle swarm optimization to solve effectively the school timetabling problem. *Soft Computing, 16*(7), 1229–1252. https://doi.org/10.1007/s00500-012-0809-5

Vrielink, R. O., Jansen, E. A., Hans, E. W., & van Hillegersberg, J. (2019). Practices in timetabling in higher education institutions: a systematic review. *Annals of Operations Research, 275*(1), 145–160. https://doi.org/10.1007/s10479-017-2688-8

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82. https://doi.org/10.1109/4235.585893

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In *Proceedings of international symposium on stochastic algorithms* (pp. 169–178). Springer.

Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In *Proceedings of nature inspired cooperative strategies for optimization* (NICSO 2010) (pp. 65–74). Springer.

Yang, X. S. (2020). Nature-inspired optimization algorithms: Challenges and open problems. *Journal of Computational Science, 46*, 101104. https://doi.org/10.1016/j.jocs.2020.101104

Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *2009 World Congress on nature & biologically inspired computing (NaBiC)* (pp. 210–214). IEEE. https://doi.org/10.1109/NABIC.2009.5393690.