Thesis Defense Scheduling Optimization Using Harris Hawk Optimization

Kevin Setiono[1], Mikhael Setiawan[1], Grace L. Dewi[1], and Erwin Dhaniswara[2]
[1]Informatics Department, Faculty of Science and Technology, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia
[2]Informatics Department, Universitas Widya Kartika Surabaya, Surabaya, Indonesia

Corresponding author: Kevin Setiono (e-mail: kevinsetiono@stts.edu).

ABSTRACT This research discusses how the Harris Hawk Optimization (HHO) algorithm handles scheduling problems. The scheduling of thesis defenses at the Institut Sains dan Teknologi Terpadu Surabaya (ISTTS) is a complex issue because it involves the availability of lecturers, teaching/exam schedules, lecturer preferences, and limited room and time availability. The scheduling constraints in this research are divided into two categories: Hard Constraints and Soft Constraints. Hard constraints must not be violated, including each lecturer's unique availability, conflicts, and existing exam or teaching schedules. Soft constraints, on the other hand, include preferences for specific days or rooms for the defense. The complexity of scheduling due to these two types of constraints leads to longer scheduling times and an increased likelihood of human error. To automate and optimize this process, the author employs the HHO algorithm. HHO is inspired by the behavior of the Harris Hawk, known for its intelligence and ability to coordinate while hunting. The results of the HHO algorithm are translated into a slot meter, which helps to map the solution to available time slots. The HHO algorithm can generate schedules that comply with 90% of the hard constraints at ISTTS. Evolutionary algorithms typically have high complexity and computational time; in this case, the researcher experimented with multiprocessing. Multiprocessing improved the computational time by up to 39%.

KEYWORDS Evolutionary Computation, Harris Hawk Optimization, Scheduling, Time Table

I. INTRODUCTION

Scheduling is a common challenge encountered in daily life, particularly within higher education institutions. At Institut Sains dan Teknologi Terpadu Surabaya (ISTTS), scheduling is typically handled manually by the staff in the Academic Bureau (BAA). Recurring tasks include class scheduling, thesis proposal scheduling, and thesis defense scheduling.

Thesis defense scheduling at ISTTS is particularly complex due to the numerous constraints that must be considered, such as lecturer availability, teaching schedules, exam schedules, lecturer preferences, and room availability. Thesis defenses at ISTTS involve four to five lecturers, all of whom must meet these constraints. The constraints are categorized into two types: hard constraints and soft constraints. Hard constraints must not be violated under any circumstances; these include ensuring that lecturers are available during the scheduled time, that they are not supervising an exam, and that they are not examining or supervising another defense at the same time. Soft constraints, such as room availability and available day slots, can be adjusted if necessary. However, the complexity of the scheduling process often results in hard constraint violations.

Given these challenges, this research aims to develop an optimal scheduling system. This type of optimization problem is often not solvable using standard mathematical calculations. As a result, many researchers have explored the use of metaheuristic algorithms. Metaheuristic algorithms have proven to be capable of solving complex problems, including those unsolvable by exact methods. In this study, the researcher seeks to optimize the scheduling problem using a derivative of a metaheuristic algorithm, the Harris Hawk Optimization (HHO) method [1].

Previous studies have applied evolutionary algorithms to scheduling tasks. Much of this research has focused on job scheduling [2], [3], [4]. While university scheduling has been attempted using evolutionary algorithms, those studies primarily focused on ensuring that lecturers' availability did not conflict with teaching commitments and that students' schedules were not too congested. This study addresses a different gap in the research by focusing on thesis defense scheduling, where each defense involves coordinating four to five lecturers in the same time slot.

II. EVOLUTIONARY ALGORITHM

The Evolutionary Algorithm (EA) is a nature-inspired algorithm that solves problems by mimicking the survival strategies of living organisms [5]. It operates similarly to Darwin's theory of evolution. The exact mechanism of an EA can vary based on the organisms being studied, but most algorithms typically include the phases of selection, crossover, and mutation, as shown in Figure 1.

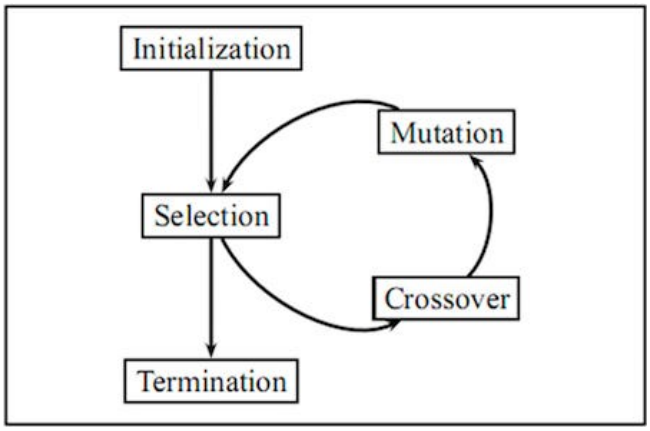

Figure 1. **Evolutionary Algorithm Workflow**

An advancement of the standard evolutionary algorithm is Particle Swarm Optimization (PSO) [6], which eventually led to the development of the Harris Hawk Optimization (HHO) algorithm. PSO is a global optimization technique that solves problems by representing solutions as points or surfaces in an n-dimensional space. Unlike standard evolutionary algorithms, PSO focuses on the collective behavior of organisms, usually simulating how animals move and work in groups, such as bird flocks or fish schools.

Evolutionary algorithms and their variations fall under the category of metaheuristics. EA has been widely applied to diverse problems, including scheduling [2], [7], optimization [8], solver searches [9], [10], and many others. In particular, EA has been frequently used to address scheduling problems, such as travel scheduling [11], job scheduling, and university scheduling [12].

III. HARRIS HAWK OPTIMIZATION (HHO)

Harris Hawk Optimization (HHO) is a relatively new optimization algorithm inspired by the hunting behavior of the Harris hawk [1]. HHO is an extension of the evolutionary algorithm, mimicking the hunting tactics of Harris hawks as they pursue rabbits. Unlike other hawks, Harris hawks employ a unique group hunting method. When hunting, they demonstrate intelligent strategies for tracking, surrounding, deceiving their prey, and eventually launching a coordinated attack.

Figure 2. **Surprise Pounce Strategy of Harris Hawks**

Harris hawks hunt in groups of 2 to 6 individuals, utilizing a tactic known as the Surprise Pounce, as depicted in Figure 2. The distinctiveness of this strategy lies in the way the hawks attack simultaneously from multiple directions, startling the rabbit and making it difficult for the prey to escape. Over time, this approach becomes more effective as the rabbit's energy is gradually depleted from repeatedly evading the coordinated group attacks. The phases of the HHO algorithm are illustrated in Figure 3.

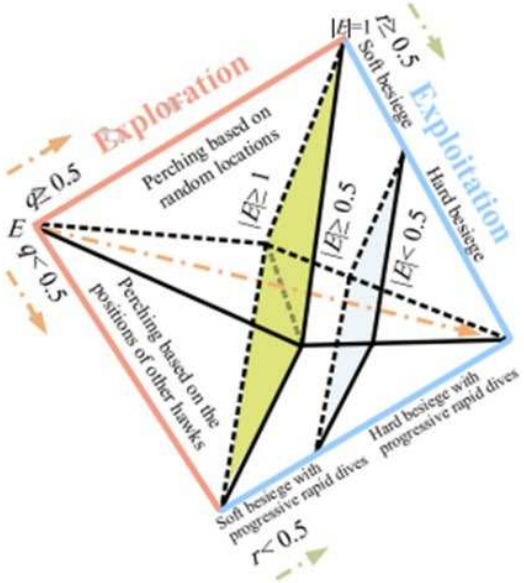

Figure 3. **Fase Harris Hawk Optimization**[1]

A. EXPLORATION PHASE

During the exploration phase, Harris hawks are randomly positioned in various locations, awaiting the detection of prey (rabbits) based on two distinct strategies. These strategies are applied randomly, with an equal probability of being selected,

determined by the value of q, a random number ranging between zero and one.

q≥0.5

$$x(t+1) = x_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| \quad (1)$$

q<0.5

$$x(t+1) = (x_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) \quad (2)$$

The variable $x(t+1)$ represents the position vector of the Harris hawk in the next iteration (1), while $x_{rand}(t)$ denotes a random position vector of the Harris hawk within the population. r_1 dan r_2 are random numbers ranging from zero to one, whereas $x(t)$ indicates the position vector of the Harris hawk in the current iteration. In the second strategy, the Harris hawk perches based on the position of the rabbit. This strategy is applied only if q ≥ 0,5 (2). The second strategy in the exploration phase is described by (2), where $x_{rabbit}(t)$ is the position vector of the rabbit in the current iteration, $X_m(t)$ represents the average position vector of all Harris hawks in the population, r_3 dan r_4 are random numbers ranging from zero to one, LB is the lower bound or minimum value of the Harris hawk's position vector, and UB is the upper bound or maximum value of the Harris hawk's position vector.

B. EXPLORATION TO EXPLOITATION TRANSITION PHASE

The HHO algorithm transitions from the exploration phase to the exploitation phase based on the remaining energy of the escaping rabbit. The energy of the rabbit significantly decreases during the escape process. E represents the energy level of the rabbit (3), while E_0 indicates the initial energy condition of the rabbit (4). This can be calculated using the formula, where rand is a random value ranging from zero to one, t denotes the current iteration, and T represents the predetermined maximum number of iterations.

$$E = 2E_0(1 - \frac{t}{T}) \quad (3)$$

$$E_0 = 2rand - 1 \quad (4)$$

C. EXPLOITATION PHASE

In this phase, the Harris hawk employs a surprise pounce to attack the detected rabbit. However, since the rabbit is also attempting to escape from the threatening situation, two distinct scenarios are utilized based on the conditions during the pursuit by the Harris hawk. The four strategies implemented are Soft Besiege and Hard Besiege.

- Soft Besiege

In this strategy, the rabbit has sufficient energy and is still trying to flee. This condition is applied when r ≥ 0,5 and $|E| <$ 0,5 (5).

$$(t+1) = \Delta X(t) - E\,|JX_{rabbit}(t) - X(t)| \quad (5)$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \quad (6)$$

$$J = 2(1 - rand)) \quad (7)$$

The difference between the position vectors of the rabbit is calculated using $\Delta X(t)$ (6). J is a random number computed using equation (7), which simulates the movement of the rabbit in its natural habitat. Meanwhile, rand is a random number ranging from zero to one.

- Hard Besiege

In this strategy, the rabbit is fatigued and possesses low energy to escape. The Harris hawk encircles the rabbit aggressively and executes a surprise pounce. This condition is applied when r ≥ 0,5 and $|E| < 0{,}5$ (8).

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \quad (8)$$

- Soft Besiege with Progressive Rapid Dives

In this strategy, the rabbit possesses sufficient energy to escape. Consequently, the Harris hawks perform a soft besiege before executing a surprise pounce. This strategy differs slightly from the previous case, where the concept of Levy flight (LF) is incorporated into the HHO algorithm to mimic the rabbit's deceptive zigzag movements. As a result, the Harris hawks carry out a series of rapid dives around the rabbit, progressively adjusting their positions to account for the rabbit's deceptive maneuvers.

$$Y = X_{rabbit}(t) - E\,|JX_{rabbit}(t) - X(t)| \quad (9)$$

$$Z = Y + S \times LF(D) \quad (10)$$

In this strategy, the Harris hawk has the freedom to choose the movement that it considers most advantageous. Therefore, the hawk moves based on a calculation, where Y represents the new position vector being considered. If the Harris hawk assesses that position Y is worse than its current position, it recalculates its position using calculation X. Here, Z represents the new position vector, LF is the result of the "Levy flight" movement calculation, which can be computed using equation (11), S is a random vector with dimensions 1 x D, and D is the dimensionality of the problem being solved. The value of theta is obtained from (12).

$$LF(x) = 0{,}01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \quad (11)$$

$$\sigma = \left(\frac{\gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}}\right)^{\frac{1}{\beta}} \quad (12)$$

Where u and v represent random numbers in the range from zero to one, and β is a constant with a value of 1.5. Based on this, we can determine the new position vector for the Harris hawk using (13). In this case, the value of Y is calculated using (9), while the value of Z is derived through (10).

$$X(t+1) = \begin{cases} Y \; if \; F(Y) < F(X(t)) \\ Z \; if \; F(Z) < F(X(t)) \end{cases} \quad (13)$$

- Hard Besiege with Progressive Rapid Dives

In this situation, the rabbit lacks the energy to attempt escape. As a result, the Harris hawk uses a hard besiege strategy before launching a surprise pounce to catch and kill the rabbit. This strategy applies when |E|<0,5 and r<0,5 is met. The concept is similar to the previous strategy, with the main difference being that the hawk aggressively approaches the rabbit's average position. In this context, (14) is used to find the new position vector for the Harris hawk. Y dan Z can be obtained using (15) and (16). Here $X_m(t)$ represents the average position of all hawks in the population. LF can be found using (11).

$$X(t+1) = \begin{cases} Y \; if \; F(Y) < F(X(t)) \\ Z \; if \; F(Z) < F(X(t)) \end{cases} \quad (14)$$

$$Y = X_{rabbit}(t) - E|J\, X_{rabbit}(t) - X_m(t)| \quad (15)$$

$$Z = Y + S \times LF(D) \quad (16)$$

IV. HHO CONFIGURATION

A. REPRESENTATION

In this research, an individual is represented as a one-dimensional array, with the size corresponding to the number of candidate students available. Each cell within the array contains a float value, which indicates the position of a candidate student within the available schedule. A higher float value signifies that the student will be scheduled later in the available time range. The float value, also referred to as the representational value, is used in the HHO algorithm calculations in this research and for fitness evaluation.

The Slotmeter is a tool that functions to convert the representational number in each cell into the actual slot position. The Slotmeter is an object that contains two arrays: the Representational Value and the Actual Value. The Representational Value is the decimal number used in the HHO algorithm's calculations [13], while the Actual Value holds the real values, which include the date, timeslot, and room associated with the slot.

To perform the conversion, the number in the cell is compared against all Representational Values in the Slotmeter. The comparison starts from the first Representational Value and proceeds sequentially. If a Representational Value is greater than the number in the cell, then that cell's number is considered to correspond to the slot indexed by that Representational Value. The complete information regarding the slot, including the date, timeslot, and room, can be accessed through the Actual Value array, which holds these details for the slot at the respective index.

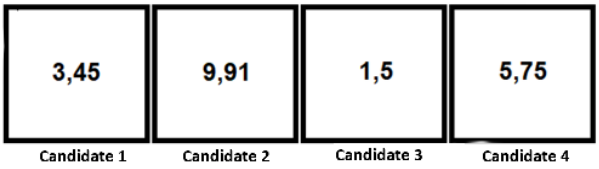

Figure 4. Individual Representation

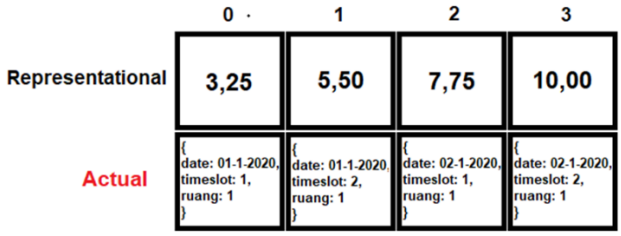

Figure 5. Slotmeter

A simple example of a Slotmeter can be seen in Figure 4. The Slotmeter shown in Figure 5 is configured with two days, two timeslots, and one room. When converting Candidate 1 from Figure 4, the first step is to compare the value in the cell, which is 3.45, against the Representational Value at index 0, which is 3.25. Since the Representational Value at index 0 is smaller, the next step is to compare the value in Candidate 1's cell with the Representational Value at the next index, which is 5.50. The Representational Value at index 1 is larger than the number in Candidate 1's cell, so the appropriate slot for the value in Candidate 1's cell is the slot at index 1. The complete information about that slot includes the date 01-01-2020, timeslot number one, and room number one.

$$\boldsymbol{gap} = \frac{(maxIndividual - minIndividual)}{maxTimeslot \times jumlahRuang \times jumlahTanggal} \quad (17)$$

The gap refers to the distance between the representational values of the Slotmeter. The calculation of the gap in the Slotmeter can be found in Equation (17). The parameters minIndividual and maxIndividual are configuration values from the HHO, while maxTimeSlot, numberOfRooms, and numberOfDates are derived from the scheduling configuration.

The Slotmeter must be created before the HHO algorithm can begin. The process starts by generating the Actual Value, which includes creating all possible slots with complete details, such as dates, timeslots, and rooms. After generating the Actual Value, the next step is to create the Representational Value. This involves calculating the gap (distance) as explained in (12). For each slot, the Representational Value is incremented, and the distance between adjacent slots is determined by the calculated gap. In this way, the Slotmeter becomes a tool that represents the conversion between the abstract representational values used in the HHO algorithm and the actual values, which contain detailed information such as dates, timeslots, and rooms. This

Slotmeter is crucial for understanding how the abstract representation of values can be linked to actual slots within the context of specific scheduling problems.

B. FITNESS

The fitness system within this scheduling program adopts a cost-based approach, where each violation of a rule or constraint results in an increase in the cost or performance value of the individual. Consequently, individuals with lower fitness or cost values are deemed superior in this context. Constraints, often referred to as "constraints," serve as guidelines or rules that must not be violated, or at the very least, should be adhered to as closely as possible. These constraints can be categorized into two types: Hard Constraints and Soft Constraints. The weights representing the significance of each rule in the scheduling program can be found in Table 1.

TABLE I
FITNESS WEIGHT

Constraint Name	Constraint Type	Weight
SELF_CONFLICT	Hard	500
DOSEN_SELF_CONFLICT	Hard	500
UJIAN_CONFLICT	Hard	100
AJAR_CONFLICT	Hard	100
HADIR_CONFLICT	Hard	250
DAY_USED	Soft	20
REPEATED_CONFLICT	Soft	50

In this context, fit(X) represents the fitness value of hawk X. Constraint[n] denotes the total violations for constraint n, where n ranges from 1 to 7 (18). Weight[n] signifies the weight assigned to constraint n. The methodology for calculating the total weight of violations is explained as follows.

$$fit(X) = constraint[0] * weight[0] + constraint[1] * weight[1] + \cdots + constraint[6] * weight[6] \quad (18)$$

C. HARD CONSTRAINT

Hard Constraints refer to the limitations or rules that must be strictly adhered to and cannot be violated. The implementation of Hard Constraints within the Harris Hawks Optimization algorithm involves assigning a high fitness cost to individuals that breach specific rules. This approach encourages the HHO algorithm to avoid violations of these rules. By imposing a significant cost, the HHO algorithm is incentivized to produce solutions that comply with the established guidelines. In the context of this scheduling program, there are five rules identified as Hard Constraints:

1. SELF_CONFLICT: Candidates must not occupy the same slot. This constraint prevents a candidate's defense schedule from appearing more than once in the same schedule.
2. DOSEN_SELF_CONFLICT: Participants in the defense must not have the same supervising professor, co-supervisor, or examiners as other participants in the same schedule.
3. HADIR_CONFLICT: The defense schedule for professors must fall within their working hours.
4. AJAR_CONFLICT: A professor must not be scheduled to teach during the defense schedule.
5. UJIAN_CONFLICT: There must be no overlap between the professor's schedule and the exam schedule (e.g., mid-term and final exams).

D. SOFT CONSTRAINT

Soft Constraints are limitations or rules that may be violated if conditions do not allow adherence, yet they should ideally be followed or minimized in violation. The fitness value assigned when such violations occur is lower than that for Hard Constraints. This design encourages the HHO algorithm to strive to minimize breaches of these rules while still permitting a degree of violation at a lower level. By assigning a lower cost to Soft Constraints, the HHO algorithm prioritizes compliance with Hard Constraints over Soft Constraints. In the context of this scheduling program, there are two rules classified as Soft Constraints:

1. DAY_USED: The number of days in the schedule or individual should be minimized. This rule implies that a violation will occur for each day utilized in a hawk's schedule. The constraint suggests that the generated defense schedule should have a shorter duration. Emphasis on this constraint aims to encourage the HHO algorithm to prefer schedules that utilize fewer days rather than those that extend over more days.
2. REPEATED_CONFLICT: Professors should not supervise or examine more than three consecutive times, either in examination or guidance roles. This constraint is implemented to ensure that professors have adequate breaks between their examination and supervision duties.

E. MIN MAX NORMALIZATION

MinMax normalization operates by transforming cell values into a specified range while preserving the differences among those values. Maintaining these differences is crucial because the optimal solution in this context is an individual with diverse cell values. By applying MinMax normalization (19), the values of a hawk do not tend to converge toward either the minimum or maximum bounds of the individual. An example of Individual Representation can be seen in Figure 6.

$$X_{scaled} = \frac{X - old\_\min(hawk)}{\text{old\_max}(hawk) - old\_\min(hawk)} \times (new\_max - new\_\min(hawk)) + new\_\min(hawk) \quad (19)$$

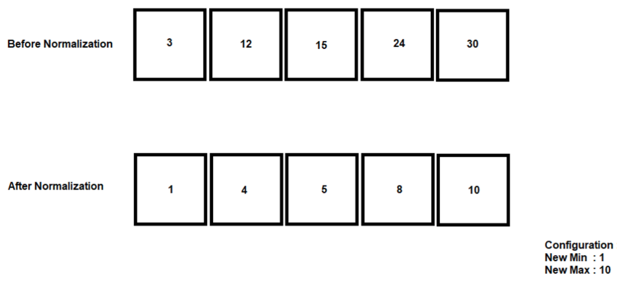

Figure 6. Individual Representation

MinMax normalization is applied within the Harris Hawk Optimization (HHO) algorithm during the translation or conversion from float representation to schedule representation. Consequently, when fitness calculations are performed, the information contained in the individual remains relatively consistent with its pre-normalization state. The MinMax normalization process for a single individual uses (19).

V. THESIS DEFENSE SCHEDULING

In this scheduling system, specific inputs are required for effective operation. All inputs will be entered in the form of an Excel file. The content of the Excel file will be presented in this study in tabular form.

TABLE II
CANDIDATE LIST

NRP	SUPERVISING LECTURER	CO-SUPERVISOR	EXAMINER 1	EXAMINER 2	EXAMINER 3
219210615	Hendrawan Armanto, S.Kom., M.Kom.	[EMPTY]	Yosi Kristian, Dr., S.Kom., M.Kom.	Lukman Zaman PCSW, Dr., S.Kom., M.Kom.	Arya Tandy Hermawan, Ir., M.T.
219210615	Hendrawan Armanto, S.Kom., M.Kom.	[EMPTY]	Yosi Kristian, Dr., S.Kom., M.Kom.	Lukman Zaman PCSW, Dr., S.Kom., M.Kom.	Arya Tandy Hermawan, Ir., M.T.
219210615	Hendrawan Armanto, S.Kom., M.Kom.	[EMPTY]	Yosi Kristian, Dr., S.Kom., M.Kom.	Lukman Zaman PCSW, Dr., S.Kom., M.Kom.	Arya Tandy Hermawan, Ir., M.T.

The XLSX input file consists of four sheets: the candidate list, class schedule, examination schedule, and attendance schedule. The "Candidate List" sheet in Table 2 contains information about the students who have registered for their thesis defenses at the Institut Sains dan Teknologi Terpadu Surabaya. In addition to details about the candidates, this worksheet also includes data regarding the supervising lecturers, co-supervisors (if applicable), and three examiners. If there is no co-supervisor, the corresponding column will be filled with the text "[EMPTY]."

The Class Schedule sheet includes information such as the name of the lecturer, time, course title, number of credit hours (with one credit hour assumed to be 50 minutes according to the ISTTS academic guidelines), and the day of the week. Similarly, the Examination Schedule sheet contains fields such as the lecturer's name, course title, date, time, and whether it pertains to the mid-term (UTS) or final exam (UAS).

TABLE III
ATTENDANCE SCHEDULE

Constraint Name	Attendance Schedule
Kevin Setiono, S.Kom.	09:00-13:00,08:00-17:00,08:00-17:00,08:00-17:00,09:00-13:00
Mikhael Setiawan, S.Kom.	08:00-17:00,08:00-17:00,08:00-17:00,08:00-17:00,08:00-17:00
Evan Kusuma Susanto, S.Kom.	08:00-17:00,08:00-17:00,08:00-17:00,08:00-17:00,08:00-17:00

The five attendance times indicate the arrival and departure times of lecturers in sequential order from Monday to Friday. Each entry in the attendance schedule is separated by the character "-", representing the check-in and check-out times. If the attendance schedule includes more than five entries, only the first five entries will be utilized. An example of the attendance schedule can be found in Table 3.

VI. EXPERIMENT RESULTS

Experiments were conducted to test the optimal HHO configuration and fitness weights, showed in Table 4, in scheduling thesis defenses at the Institut Sains dan Teknologi Terpadu Surabaya. In addition to configuration trials, testing was performed on all available schedule data. The implementation of multiprocessing was tested on the

performance of the HHO algorithm. Each weight would be tested a minimum of five times.

TABLE IV
TEST FITNESS WEIGHTS

Test Constraint Name	1	2	3
SELF_CONFLICT	200	1000	500
DOSEN_SELF_CONFLICT	200	1000	500
HADIR	200	500	250
UJIAN	200	500	100
AJAR	200	500	100
DAY	100	50	20
REPEATED	100	50	50

TABLE V
TOTAL CONFLICT AND CONFLICT WEIGHT

Test Fitness Number	Total Conflict	Total Conflict Weight
F1R1	33	5600
F1R2	47	8200
F1R3	39	6600
F1R4	47	8200
F1R5	36	6200
F2R1	25	7100
F2R2	42	16550
F2R3	53	20650
F2R4	48	18600
F2R5	60	25150
F3R1	29	2080
F3R2	29	2390
F3R3	36	3790
F3R4	29	2410
F3R5	33	4400

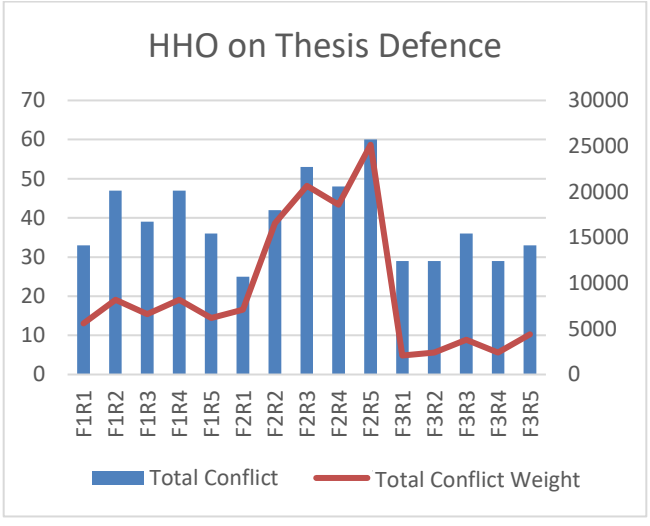

Figure 7. HHO Trial Graph for Thesis Scheduling with Three Types of Fitness Weights and Five Runs for Each Fitness Type

In the experiment for testing fitness weights, each configuration was executed five times as written in Table 5, and the best and worst outcomes from all trials were recorded as in Figure 7. The success of the weight trials was measured based on the non-violation of hard constraints. Thus, the evaluation of the weight configuration was based on the number of hard constraint violations that occurred. A smaller number of hard constraint violations indicates a better weight configuration.

In the first fitness trial, the first run yielded the best result with a total of 23 hard constraint violations. These violations included 10 violations of attendance rules, 0 violations of exam rules, and 13 violations of teaching or lecture rules. The scheduling outcome from the first trial resulted in a schedule lasting 10 days.

In the second fitness trial, the first run also achieved the best outcome with 13 hard constraint violations: 1 attendance rule violation and 12 teaching rule violations. The schedule in this trial lasted 12 days.

The third fitness trial, specifically the third run, produced the best result. The first trial of this configuration generated a total of 15 hard constraint violations, consisting of 2 violations of attendance rules and 13 violations of teaching rules. The schedule from this trial lasted 14 days.

In the third configuration experiment, it was observed that the best result from five trials did not yield a more optimal schedule compared to the best outcome from the second configuration trial. Despite the differences resulting in less favorable outcomes, this can be considered as a matter of luck. Additionally, the BAA administrator responsible for creating the thesis scheduling at ISTTS during this research indicated that the constraints related to attendance scheduling are of higher importance than those concerning lecture or exam scheduling. Therefore, the third configuration will be adopted as the final configuration in this experiment..

To enable the HHO algorithm to produce optimal results, appropriate configurations are essential. Therefore, trials were conducted to determine the correct HHO configuration. The tested configurations included maximum iterations, the number of hawks, minimum individuals, and maximum individuals. The evaluation of the configurations was performed using (20).

$$score = 1 - \frac{final\_fitness}{maximum\_fitness} \tag{20}$$

The configuration trial scores were calculated using Formula 14. The scores range from 0 to 1, where 0 indicates the worst outcome and 1 represents the best possible result. However, in the context of the HHO algorithm, achieving a score of 1 is not feasible due to the day constraint. Since the schedule must always account for the use of days, this constraint is inevitably violated. Seven configurations were tested to identify the optimal setup, refer to Table 6. Below is a description of the trials conducted.

TABLE VI
TOTAL CONFLICT DAN CONFLICT WEIGHT

No	Min Individual	Max Individual	Max Iteration	Starting Individual
1	1	10	100	20
2	1	500	100	20
3	1	1000	100	20
4	1	1000	100	50
5	1	1000	500	100
6	1	1000	500	200
7	1	1000	500	500

TABLE VII
CONFIGURATION TESTS AVERAGE RESULTS

Config Number	Ending Iteration	Final Fitness	Time Elapsed	Score
1	100	3093	1,073,666	0,983633
2	100	3223	1.364	0,983281
3	99	3470	1324	0,981542
4	100	3214	3.214,333	0,980957
5	410,666	2306	19.302,333	0,987730
6	428,666	1450	27.688,333	0,992287
7	100	1670	55.614,333	0,991117

To review the final fitness values from all trials, refer to Table 7. A comparison of configurations one, two, and three indicates that the parameters min_individual and max_individual had minimal influence on the final fitness value. In contrast, testing the number of individuals and iteration parameters (configurations four through seven) led to a significant reduction in hard constraint violations and an improvement in the final fitness value between configurations three and four. Configuration seven achieved the best result after increasing both parameters. The fitness results from the configuration trials are shown in Figure 8.

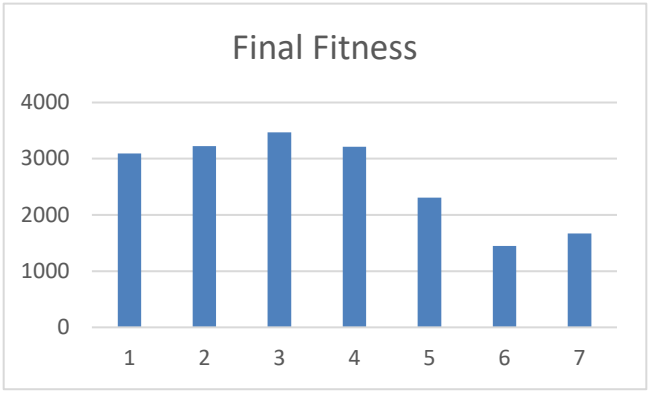

Figure 8. **Fitness Results of Configuration Trials**

In configuration six, the HHO algorithm converged between iterations 400 and 450. Consequently, in configuration seven, only the number of individuals was increased. Despite this modification, configuration seven did not yield a higher-quality schedule. Therefore, configuration six was determined to be the optimal setup.

TABLE VIII
TOTAL CONFLICT DAN CONFLICT WEIGHT

No	Process Type	Time Elapsed (Seconds)
1	1	1874
2	1	1128

For the multiprocessing experiment, data from the second period of the odd semester in the 2020/2021 academic year were utilized, involving 35 candidates. The experiment was conducted once for each type of process. The schedule spanned two weeks, with four timeslots and four rooms. The HHO setup included a maximum of 100 iterations and 20 hawks. The multiprocessing configuration employed four workers. The limited number of iterations ensured that the HHO algorithm would not terminate prematurely due to convergence, thus minimizing its impact on runtime. The trial results (see Table 8) indicate that, with a linear process, the program required 1874 seconds (approximately 31 minutes). In contrast, with multiprocessing, the task was completed in 1128 seconds (approximately 18.8 minutes), demonstrating a performance improvement of about 39%.

VII. CONCLUSION

The scheduling of thesis defenses at Institut Sains dan Teknologi Terpadu Surabaya, using the Harris Hawk Optimization (HHO) method, successfully produced a schedule with a hard constraint violation rate of less than 10%. HHO has proven to be an effective approach for solving time scheduling problems, especially due to its integrated exploration and exploitation phases, as well as its phase transition mechanism. Trials demonstrated that HHO could schedule approximately 90% of the candidates without significant hard constraint violations. Additionally, implementing multiprocessing in the HHO algorithm, particularly in the fitness calculation process, improved the algorithm's speed by reducing execution time by 39%.

AUTHORS CONTRIBUTION

Kevin Setiono: Investigation, Original Draft Writing Preparation, Project Administration, Original Drafting Writing;
Mikhael Setiawan: Investigation, Data Collection & Processing, Review & Editing Writing;
Grace Levina Dewi: Investigation, Data Analysis, Results Interpretation, Review & Editing Writing;
Erwin Dhaniswara: Investigation, Resources;

COPYRIGHT

REFERENCES

[1] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97, 2019, doi: 10.1016/j.future.2019.02.028.

[2] K. Mesghouni and S. Hammadi, "Evolutionary algorithms for job-shop scheduling," *Int. J. Appl. Math. Comput. Sci*, vol. 14, pp. 91–103, Nov. 2004.

[3] C. Liu, "An improved Harris hawks optimizer for job-shop scheduling problem," *Journal of Supercomputing*, vol. 77, no. 12, 2021, doi: 10.1007/s11227-021-03834-0.

[4] B. Tagtekin, M. U. Öztürk, and M. K. Sezer, "A Case Study: Using Genetic Algorithm for Job Scheduling Problem," *CoRR*, vol. abs/2106.04854, 2021, [Online]. Available: https://arxiv.org/abs/2106.04854

[5] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. in Decision Engineering. Springer London, 2010. [Online]. Available: https://books.google.co.id/books?id=rHQf_2Dx2ucC

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.

[7] G. B. Satrya and S. Y. Shin, "Evolutionary computing approach to optimize superframe scheduling on industrial wireless sensor networks," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, 2022, doi: 10.1016/j.jksuci.2020.01.014.

[8] M. B. Wall, "A Genetic Algorithm for Resource-Constrained Scheduling by," *Design*, 1996.

[9] E. Dinata, H. Budianto, and H. Armanto, "Hyper Sudoku Solver dengan Menggunakan Harris Hawks Optimization Algorithm," *INSYST: Journal of Intelligent System and Computation*, vol. 2, no. 1, pp. 20–27, Apr. 2020, doi: 10.52985/insyst.v2i1.153.

[10] B. Milenković, "Implementation of Harris Hawks Optimization (HHO) algorithm to solve engineering problems," *Tehnika*, vol. 76, no. 4, 2021, doi: 10.5937/tehnika2104439m.

[11] H. Armanto, R. Kevin, and P. Pickerling, "Perencanaan Perjalanan Wisata Multi Kota dan Negara dengan Algoritma Cuttlefish," *INSYST: Journal of Intelligent System and Computation*, vol. 1, no. 2, pp. 99–109, Dec. 2019, doi: 10.52985/insyst.v1i2.91.

[12] M. Aldasht, M. Alsaheb, S. Adi, and M. A. Qopita, "University course scheduling using evolutionary algorithms," in *4th International Multi-Conference on Computing in the Global Information Technology, ICCGI 2009*, 2009. doi: 10.1109/ICCGI.2009.15.

[13] F. Rothlauf, "Representations for Evolutionary Algorithms," in *GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion*, Nov. 2011, pp. 1191–1212. doi: 10.1145/2330784.2330921.