Research Article

Mathematical Modeling and Discrete Firefly Algorithm to Optimize Scheduling Problem with Release Date, Sequence-Dependent Setup Time, and Periodic Maintenance

Mustafa A. Qamhan ¹, Ammar A. Qamhan ², Ibrahim M. Al-Harkan,²
and Yousef A. Alotaibi¹

¹College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

²College of Engineering, King Saud University, Riyadh, Saudi Arabia

Correspondence should be addressed to Mustafa A. Qamhan; 437107175@student.ksu.edu.sa

Received 9 April 2019; Revised 15 June 2019; Accepted 26 June 2019; Published 5 August 2019

Academic Editor: Francesco Zammori

Copyright © 2019 Mustafa A. Qamhan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An evolutionary discrete firefly algorithm (EDFA) is presented herein to solve a real-world manufacturing system problem of scheduling a set of jobs on a single machine subject to nonzero release date, sequence-dependent setup time, and periodic maintenance with the objective of minimizing the maximum completion time “makespan.” To evaluate the performance of the proposed EDFA, a new mixed-integer linear programming model is also proposed for small-sized instances. Furthermore, the parameters of the EDFA are regulated using full factorial analysis. Finally, numerical experiments are performed to demonstrate the efficiency and capability of the EDFA in solving the abovementioned problem.

1. Introduction

Production scheduling is a form of decision-making that significantly impacts any manufacturing or service [1]. Efficient production schedules are critical in improving organization productivity by exploiting the production system efficiently and conducting the best utilization. According to the production environment, the single-machine scheduling problem (SMSP) is one of the most common scheduling problems that have been studied in the past few decades. However, in some real-world manufacturing systems, studying the SMSP under the setup consideration between jobs is also required such as the manufacturing systems in painting and metalworking industries [2]. Comprehensive reviews regarding the setup consideration are provided [3–5] for all scheduling problem environments such as the single-machine scheduling, parallel machine scheduling, and flow shop scheduling problems. The authors clarified the sequence-dependent setup time when the setup time for any job depends on the immediately preceding job; i.e., the setup time for the job i after job j

should not equal the setup time for the same job i after any other job. Different algorithms were proposed in [6–15] to optimize the SMSP with considering of the sequence-dependent setup times constraints. An ant colony optimization was presented in [16] to minimize job tardiness for an SMSP with sequence-dependent setups. A single-machine scheduling problem with sequence-dependent setup times and release dates had been less studied [2, 17, 18]; one of these studies is [2] where a beam search and mixed-integer linear model were used to study the SMSP subject to the release date and nonanticipatory dependent setup time to minimize the makespan.

Additionally, in several studies on scheduling in a single-machine environment as well as the previously mentioned papers, the problems have been simplified by ignoring the maintenance intervention dates assuming the production systems are continuously available over the planning horizon, which would increase the probability of machine breakdowns. Many researchers have adopted some basic assumptions in their studies [19–21] by considering that

FIGURE 1: An illustration for the periodic maintenance.

maintenance activities are important to improve the manufacturing system efficiency and product quality. However, to maintain the production system in a good condition and to decrease the probability of system failures, various maintenance policies (periodic, preventive, among others) have been applied in many real-world manufacturing systems where the periodic maintenance policy has been widely used [22]. Maintenance activities were performed during known intervention dates and occupied part of the running time of the planning horizon of the machine. Hence, the scheduling of jobs on the available machine's running time is an important issue that attracted many researchers. Reference [23] addressed the problem when the maintenance duration is dependent on the workload time, and the start time for each maintenance is restricted to a given time window. A mixed-integer linear programming model was used by [24] to minimize the makespan of a set of jobs on the SMSP with flexible periodic preventive maintenances, where the jobs have their own release dates. Moreover, the authors provided two complementary algorithms, i.e., a heuristic consisting of earliest release date rule and longest processing time (ERD-LPT) and Branch and Bound algorithm. Reference [25] offered six types of heuristic algorithms to minimize the makespan in the SMSP by considering flexible periodic maintenance. An uncertain single-machine scheduling problem with periodic maintenance studies the SMSP subject to periodic maintenance with considering the uncertainty environment for the processing and maintenance time. The uncertain data caused where the data cannot be accurately obtained and could be estimated via work experience. Three uncertain programming models were reported in [26] to solve the SMPS with periodic maintenance when the time of processing and repair are nondeterministic. Reference [27] provided a mathematical formulation and a knowledge-based variable neighborhood search algorithm to minimize the sum of maximum earliness and tardiness on a single machine with multiple unavailability constraints.

The works that address the SMSP with sequence-dependent setup times and periodic maintenance constraints are few. References [28, 29] considered the scheduling problem by minimizing the total earliness and tardiness penalties on the single machine with sequence-dependent setup times and planned maintenance constraints. Paper [30] addressed the scheduling problem on a single machine with availability constraints and sequence-dependent setup to minimize the makespan; they also presented a mixed-integer linear model. In another work [31], the same authors offered a

metaheuristic that combined the greedy randomized adaptive search procedure and a local search for the SMSP subject to preventive maintenance and sequence-dependent setup times. Finally, [32] addressed the same above-mentioned problem and presented another metaheuristic that depended on the tabu search embedded in a multistart procedure.

The primary interest in this work is to provide a mixed-integer linear programming model and an effective algorithm called evolutionary discrete firefly algorithm for the production schedules for the single-machine problem subject to release date, dependent setup time, and periodic maintenance, to minimize the makespan. To our knowledge, the addressed problem herein has been not studied hitherto. The remainder of this paper is organized as follows: The next section will be dedicated to describe the problem in detail and formulate it with a mixed integer linear programming model. Section 3 presents a brief introduction to firefly algorithm, and Section 4 will be to demonstrate the proposed algorithm (EDFA) in details. Section 5, discusses the parameter tuning for the proposed algorithm. Computational experiments are given in Section 6. Finally, Section 7 makes a conclusion and recommendation discuss for the possible future work.

2. Problem Description and Mathematical Modeling

2.1. Problem Definition and Assumptions. The considered problem is defined as a set of jobs $J = \{J_1, J_2, \dots, J_n\}$ on a single machine where each job has a certain processing time $p = \{p_1, p_2, \dots, p_n\}$ and release date $r = \{r_1, r_2, \dots, r_n\}$. The machine can only process one job J_i at a time. Job preemptions are not allowed and no precedence relations exist between them. Each job J_j has a setup time s_{ij} that depends on the immediate successor job J_i . The machine is not continuously available over the planning horizon of the machine and a periodic maintenances activity should be performed, as shown in Figure 1. The time between two maintenances T is presumptively fixed, and the time required to perform maintenances is M . The aim is to obtain the best sequence of the n jobs on the machine that minimizes the maximum completion time C_{max} .

2.2. Proposed Model. The notations in Table 1 are used to formulate the problem.

The mixed-integer linear programming (MILP) model formulated based on the assumptions above and the mathematical notations are as follows:

TABLE 1: Model notations.

<i>Indices</i>	
i	the index of jobs, where $i = 1, 2, \dots, n$;
j	the extended index of jobs, where $j = 0, 1, \dots, n$;
k	the index of jobs position in the sequence, where $k = 1, 2, \dots, n$;
<i>Parameters</i>	
p_i	Processing time of job i
s_{ij}	Setup time for job j after job i
r_i	Release date of job i
M	Time of performing maintenances activates
T	Fixed time between two maintenance
A	A large enough number
<i>Decision variables</i>	
x_{ik}	A binary decision variable equal to 1 if job i processed in position k , otherwise 0.
y_{ji}	A binary decision variable equal to 1 if job i processed immediately after job j , otherwise 0.
z_k	A binary decision variable equal to 1 if a maintenance is performed immediately before job in position k , otherwise 0.
C_{max}	Maximum completion time
t_k	Starting time of the job in position k .
f_k	Completion time of the job in position k .
c_k	The machine's age immediately after performed job in position k
ST_{jik}	A decision variable equal to s_{ij} if the job j is performed in position k , otherwise 0.

$$\min C_{max} \quad (1)$$

$$\text{Subject to} \quad \sum_{(i \neq j, j=0)}^n x_{ji} = 1, \quad i = 1, 2, \dots, n \quad (2)$$

$$\sum_{(i \neq j, i=1)}^n x_{ji} \leq 1, \quad j = 1, 2, \dots, n \quad (3)$$

$$\sum_{(i=1)}^n x_{0i} = 1 \quad (4)$$

$$\sum_{(k=1)}^n y_{ik} = 1, \quad i = 1, 2, \dots, n \quad (5)$$

$$\sum_{(i=1)}^n y_{ik} = 1, \quad k = 1, 2, \dots, n \quad (6)$$

$$y_{ik} + y_{j,k+1} \leq 2x_{ij} - A(y_{ik} + y_{j,k+1} - 2), \quad (7)$$

$$i, j = 1, 2, \dots, n \text{ \& } k = 1, 2, \dots, n-1$$

$$y_{i1} = x_{0i}, \quad i = 1, 2, \dots, n \quad (8)$$

$$ST_{j,i,k} = s_{ji}x_{ji} + A(y_{j,k-1} + y_{i,k} - 2), \quad (9)$$

$$i, j, k = 1, 2, \dots, n$$

$$t_j \geq Mz_j + T \sum_{(i=1)}^j z_i, \quad j = 1, 2, \dots, n \quad (10)$$

$$t_j \geq f_{j-1}, \quad j = 1, 2, \dots, n \quad (11)$$

$$t_k + \sum_{(i=1)}^n \left(p_i y_{ik} + \sum_{(j=1)}^n ST_{j,i,k} \right) = f_k, \quad k = 1, 2, \dots, n \quad (12)$$

$$t_1 + \sum_{(i=1)}^n (p_i + s_{0i}) y_{i1} = f_1 \quad (13)$$

$$\sum_{(i=1)}^n (r_i + p_i + s_{0i}) y_{i1} = c_1 \quad (14)$$

$$f_k - T \sum_{(i=1)}^k z_i = c_k, \quad k = 1, 2, \dots, n \quad (15)$$

$$c_k \leq T, \quad k = 1, 2, \dots, n \quad (16)$$

$$\sum_{(i=1)}^n r_i y_{ik} - t_k \leq 0, \quad k = 1, 2, \dots, n \quad (17)$$

$$C_{max} \leq f_i, \quad i = 1, 2, \dots, n \quad (18)$$

$$x_{ji}, y_{ik}, z_k \text{ binary variables } \in \{0, 1\}, \quad i, j, k = 1, 2, \dots, n \quad (19)$$

$$f_i, t_i, ST_{jik}, c_i \geq 0, \quad i, j, k = 1, 2, \dots, n \quad (20)$$

Equation (1) minimizes the makespan. Constraints (2) until (4) were used to obtain the expected sequence that determines the required setup time, whereas constraint (2) guarantees that every job has exactly one predecessor. The first job in the schedule has a predecessor which is the dummy job 0, for that ($j = 0$) as shown in Constraint (2). Constraint (3) ensures the feasibility of the sequence in which the maximum number of successors for any job is no greater than one. Because the last job in the scheduled has no successor, the equality condition for Constraint (3) cannot be considered. Constraint (4) imposes that the dummy job 0 be placed as the first in the sequence. Constraint (5) ensures that each real job occupies only one position of the scheduling sequence. Constraint (6) ensures that each position can only be occupied by one job. Constraints (7) and (8) guarantee the sequencing similarity of the jobs in setup decision variable ' x_{ji} ' and job scheduling decision variable ' y_{ik} '; i.e., if the job i processed in position k ($y_{ik} = 1$) and the job i also processed immediately after job j ($x_{ji} = 1$), Constraints (7) and (8) guarantee that job j occupies position $k-1$ ($y_{i(k-1)} = 1$). Constraint (9) calculates the setup time for each job by considering its position and the immediately preceding job. Constraint (10) guarantees that the job cannot start when performing maintenance activities. Constraint (11) ensures each job cannot start until the predecessor job is completed. Constraints (12) and (13) determine the starting time and completion time for all jobs. Constraint (14) calculates the machine's age after running the first job, and constraint (15) calculates the machine's age after running each job, where the

machine's age is the total running time for the machine after the last maintenance activity with adding the idle time that caused the release date constraint. Constraint (16) guarantees that the machine's age is no larger than the time required between two maintenances. Constraint (17) indicates that the job cannot start until the job is released. Constraint (18) calculates the makespan. Constraints (19) and (20) are used to indicate the type of decision variables.

By using the CPLEX solver, an instance with six jobs ($n = 6$) was used to test the proposed MILP model. The input data for the given example are presented in Table 2. The Gantt chart for the optimal schedule is shown in Figure 2.

2.3. Preliminary Analysis. To evaluate the performance of the EDFD for large-scale problems, some properties of the problem are introduced in the following.

Property 1. For any feasible schedule s , $\max_i(r_i + p_i + \min_j s_{ji}) \leq C_{max}(s)$

Proof. As it is known, the maximum completion time for any set of jobs must be larger than the latest release date $\max_i(r_i)$, by adding the processing time and the smallest setup time for the job with the latest release date.

Therefore, we can obtain $\max_i(r_i + p_i + \min_j s_{ji}) \leq C_{max}(s)$. $\square$

Property 2. For any scheduling problem, $\min_i r_i + \sum_{(i=1)}^n (p_i + \min_j s_{ji}) \leq C_{max}$.

FIGURE 2: Optimal schedule for the example.

TABLE 2: Job data for the given example.

Job	1	2	3	4	5	6
p_i	67	93	54	32	14	95
r_i	51	104	184	61	280	206
s_{0i}	8	3	6	2	1	8
s_{1i}	-	1	5	7	6	7
s_{2i}	6	-	4	1	1	7
s_{3i}	5	4	-	3	3	3
s_{4i}	6	1	1	-	5	3
s_{5i}	2	7	8	8	-	7
s_{6i}	1	7	5	2	8	-
$T = 114$	$M = 7$					

Proof. Suppose that the total machine idle time is only when the machine is waiting for the first job in any feasible schedule s . First, the machine cannot start before the smallest release date ($\min_i r_i$) for the set of job $\{j_1, j_2, \dots, j_n\}$; subsequently, the actual running time for the machine equals to the smallest possible amount of time by using the smallest setup time for each job regardless of the sequence consideration except for the processing time ($\sum_{i=1}^n (p_i + \min_j s_{ji})$). Because the maintenance constraint is also not considered, the C_{max} is larger. $\square$

Note. The lower bound of the problem is estimated by the maximum value between these two properties.

3. Firefly Algorithm (FA)

The firefly algorithm (FA) is one of the swarm intelligence methods first developed by [33]. The algorithm is inspired by the social behaviour of fireflies in which they produce flashlights to attract their mates. Three idealized characteristics of the fireflies' flashing light are used to develop the FA to obtain an optimal solution [34]. First, all fireflies are assumed to be unisex, and any firefly can be attracted to another regardless of its gender. Next, the attractiveness of the fireflies is directly proportional to their brightness, and it decreases when the distance between the two fireflies increases. Thus, for any two flashing fireflies, the less luminous one will be attracted

and moves to the other. Finally, the brightness of the flashing light of a firefly is considered as an objective function that requires optimization. The FA procedure can be described as in Algorithm 1 [35].

4. Evolutionary Discrete Firefly Algorithm

The FA was primarily developed for continuous optimization [36], which attracted many researchers to adopt the classic FA to a discrete version to solve different discrete problems such as the scheduling problem [37, 38] and vehicle routing [35, 39]. The primary steps of our EDFA are presented as follows.

4.1. Generation of the Initial Population. The first step of the EDFA is to generate an initial population that represents a set of possible and feasible solutions for the addressed problem. This initial population of fireflies contains m random permutations of $(1, 2, \dots, n)$, with the consideration that each firefly has a different permutation from the others. The population size determines the number of solutions or the size of the search space whose purpose is to direct the search to the best location.

4.2. Evaluation of the Light Intensity. The light intensity for each firefly is determined by its objective function value in the EDFA, which is the maximum completion time in our study (Algorithm 2). Because the problem is a minimization

```

Parameter Initialization for the algorithm & define the objective function  $F(x_i)$ 
Generation of the initial population  $x_i = [x_1, x_2, \dots, x_m]$ .
while  $i \leq$  number of fireflies do
    Evaluation of the light intensity  $I_{xi}$ 
     $i++$ ;
End
while  $x \leq$  Maximum number of iterations do
     $i \leftarrow 1$ ;
    while  $i \leq$  number of fireflies do
         $j \leftarrow 1$ ;
        while  $j \leq$  number of fireflies do
            if  $I_{xi} > I_{xj}$ 
                Calculate the distance between two fireflies  $r_{ij} = \|X_i - X_j\|$ 
                Firefly with less luminous moves to the brighter once
                Attractiveness varies with distance  $r$  via  $e^{-\gamma r}$ ;
                Evaluation of the light intensities of new solutions and update the light
                intensity;
                 $j++$ ;
            End
            Rank the fireflies and obtain the current best;
             $i++$ ;
        End
         $x++$ ;
    end

```

ALGORITHM 1: Classic Firefly Algorithm (FA).

```

 $x \leftarrow$  the index of jobs position, where  $x = 1, 2, \dots, n$ ;
 $f_x \leftarrow$  Finishing time of the job in position  $x$ .
 $r_x \leftarrow$  Release date of job in position  $x$ 
 $p_x \leftarrow$  Processing time of job in position  $x$ 
 $s_{(Job_{x-1})(Job_x)} \leftarrow$  Setup time for job in position  $x$  after job in position  $x - 1$ 
 $a \leftarrow$  First job on the schedule
 $f_a = r_a + p_a + s_{0a}$ 
while  $x = x + 1 \leq n$  do
    If ( $f_{x-1} > r_x$ )
         $f_x = f_{x-1} + p_x + s_{(Job_{x-1})(Job_x)}$ 
        If ( $Job\ x$  interrupted any maintenance intervention dates)
             $M =$  the end of that maintenance
             $f_x = M + p_x + s_{(Job_{x-1})(Job_x)}$ 
    Else
         $f_x = r_x + p_x + s_{(Job_{x-1})(Job_x)}$ 
        If ( $Job\ x$  interrupted any maintenance intervention dates)
             $M =$  the end of that maintenance
             $f_x = M + p_x + s_{(Job_{x-1})(Job_x)}$ 
     $C_{max} =$  maximum completion time of  $t_x$ 
end

```

ALGORITHM 2: Objective function.

problem, the firefly with a lower objective function value is more attractive than the others. Generally, this evaluation is used to obtain the best individual or bad individual.

4.3. Distance Function. In the continuous optimization problem, the Euclidian distance is widely used to determine the distance between two fireflies. For our discrete problem, the

distance between any two fireflies is measured by the number of different arcs between them [40]. For example, given we have two fireflies, each consists of eight nodes:

First firefly, $\pi_1 = \{2, 5, 1, 3, 6, 8, 7, 4\}$.

Second firefly, $\pi_2 = \{2, 5, 3, 8, 6, 1, 7, 4\}$.

The distance between the two fireflies is 5 = {5-1, 5-3}, {1-3, 1-7}, {3-6, 3-8}, {6-8, 6-1}, {8-7, 1-7}.

4.4. Attractiveness. In the classic FA, the attractiveness function is defined using the light intensity, the distance between fireflies, and an absorption coefficient.

$$\beta(r) = \beta_0 e^{-\gamma r^2} \quad (21)$$

This scheme is completely adopted in the EDFA.

4.5. Firefly Movement. The movement of a firefly i that is attracted to another brighter firefly j is expressed as

$$n = \text{random}(2, r_{ij} \cdot \gamma^g) \quad (22)$$

where r_{ij} is the distance between the two fireflies i and j , and g is the iteration number. In this case, the length of the firefly movement will be a random number between 2 and $r_{ij} \cdot \gamma^g$. This value is used to generate n successors from the solution corresponding to the firefly to be moved.

4.6. New Population Creation. When the fireflies are moving toward to the brighter ones, the translocation mutation strategy for each firefly couple is applied to create the “ n ” number of new fireflies. The length of the translocation mutation is determined by

$$L = \text{random}(2, r_{ij}) \quad (23)$$

Figure 3 illustrates an example of the proposed mutation strategy. In this method, at first two fireflies are selected through their attractiveness evaluation (Figure 3(a)). And then, the first mutation point is selected randomly from a discrete uniform distribution in $(1, n - L)$ and the second mutation point equals to $(\text{the first mutation point} + L)$, (Figure 3(b)). Then all digits between the two mutation points will be exchanged between the two fireflies (Figure 3(c)). After that, the remaining empty positions in the new fireflies 1 and 2 are consecutively filled by the unassigned jobs in the same order that appears in the original fireflies (Figure 3(d)). This process will be repeated for n times according to the values of fireflies' movement. After all successor new fireflies are created, their light intensities will be evaluated. During the pair comparison loop, the best solution currently is updating iteratively.

The new fireflies are generating from the best solution on each iteration (Figure 4(a)). In continuance, creating an empty template has the same number of digits of the solution filling it with a random binary number to each digit (Figure 4(b)). The digits with ‘1’s in the binary templet will keep the same job in its positions from the best solution in the new solution (Figure 4(c)). Then, fill the remaining empty digits with the remaining unselected jobs randomly (Figure 4(d)). For all new iterative stages, the brightness and attraction of each firefly are calculated.

4.7. Local Search. The procedure consists of swapping two jobs selected randomly (IT), or the random insertion of a job in another random position (IS) for the possibility of improvement (Figure 5). Both variants IT/IS and IS/IT could be used in a random way depending on which is implemented first.

TABLE 3: EDFA parameter setting.

Parameters	
Number of iterations	{80, 120}
Number of fireflies	{5, 10, 15}
Fireflies' attractiveness coefficient (β)	{0.05, 1}
Light absorption coefficient (γ)	{0.5, 0.75, 1}

4.8. EDFA Decoding Scheme. The structure of the proposed EDFA is shown in Figure 6.

The EDFA has two useful elements that render it competitive with other algorithms:

- (1) creating a new generation of fireflies by the translocation mutation method and
- (2) creating a local search combining IT and IS.

5. EDFA Parameters Setting

Different types of parameter settings were used to test the EDFA. The list of parameters and their expected levels are presented in Table 3. About 540 investigational runs have been done on ten instances with a number of jobs = 15. The average relative percentage deviation (ARPD) from the lower bound is the response variable of the factorial analysis. To conduct the best model, the nonsignificant terms were eliminated according to the P value and the final result showed Table 4.

Figure 7 shows the interaction between the parameters, in which we found that the number of fireflies has a significant effect on the algorithm performance.

The best values were obtained by a full factorial design analysis, where the number of fireflies = 15, the fireflies' attractiveness coefficient (β) = 0.5, the light absorption coefficient (γ) = 0.75, and the number of iterations = 120.

6. Computational Analysis

In this section, the EDFA performance evaluation is discussed. To examine the EDFA behaviour, the algorithm was coded in C and tested on 480 randomly generated instances using a PC with an Intel Core i7 2.80-GHz CPU and 8-GB RAM.

6.1. Data Generation. As a wide set of benchmark instances is not available for the addressed problem, two sets of instances are randomly generated based on a uniform distribution for processing time, release dates, and setup time as shown in Table 5. The maintenance time and the fixed time between two maintenances are generated dependent on the total expected time ($\sum \sum (p_j + s_{ij}/n)$) and a control variable for each parameter, where ‘ t ’ and ‘ m ’ are control variables for the fixed time between two maintenances and the maintenance time respectively. As Table 5 clearly indicates, the parameters are randomly generated from different parameters and include different kinds of circumstances for a more convincing validation.

TABLE 4: Analysis of the significance of critical parameters.

Source	DF	Adj SS	Adj MS	F-value	P-value
Model	20	2.62292	0.131146	389.56	0.001
Blocks	9	2.60643	0.289604	860.26	0.001
Linear	7	0.01218	0.001740	5.17	0.001
Number of iterations	1	0.00048	0.000482	1.43	0.232
Number of fireflies	2	0.00973	0.004864	14.45	0.001
β	2	0.00174	0.000872	2.59	0.076
γ	2	0.00023	0.000114	0.34	0.713
2-way Interactions	4	0.00431	0.001078	3.20	0.013
Number of fireflies* γ	4	0.00431	0.001078	3.20	0.013
Error	519	0.17472	0.000337		
Total	539	2.79765			

TABLE 5: EDFA data generation conditions.

Group	Processing times (p)	Release dates (r)	Setup time (s)
First set	$U(10, 100)$	$U\left(0, R * \sum \sum \left(p_j + \frac{s_{ij}}{n}\right)\right)$	$U(1, 9)$
Second set			$U(1, 49)$
where $R = \{0.6, 1, 1.4\}$			
Maintenance time (M) = $m * \sum \sum (p_j + s_{ij}/n)$, where $m = \{0.02, 0.05\}$			
Fixed time between two maintenance(T) = $t * \sum \sum (p_j + s_{ij}/n)$, where $t = \{0.3, 0.4\}$			

TABLE 6: Comparisons of EDFA with CPLEX.

No.	Parameters				CPLEX CPU running time				
	n	T	m	R	ARPD	Avg	Max	Min	Std
1	10	0.3	0.02	0.6	0	39.15	258.35	7.40	73.23
2	10	0.3	0.02	1	0	12.65	22.60	7.65	4.54
3	10	0.3	0.02	1.4	0	18.31	88.97	6.27	24.18
4	10	0.3	0.05	0.6	0	17.55	40.57	10.43	9.86
5	10	0.3	0.05	1	0	33.20	89.06	9.47	32.62
6	10	0.3	0.05	1.4	0	14.54	23.51	7.18	5.81
7	10	0.4	0.02	0.6	0	14.58	38.45	7.15	9.78
8	10	0.4	0.02	1	0	11.72	23.33	6.64	5.41
9	10	0.4	0.02	1.4	0	13.02	24.15	6.50	6.74
10	10	0.4	0.05	0.6	0	18.27	35.42	7.60	10.40
11	10	0.4	0.05	1	0	14.11	36.99	7.36	8.14
12	10	0.4	0.05	1.4	0	20.45	32.46	10.29	8.16
13	12	0.3	0.02	0.6	0	91.89	216.73	10.44	62.45
14	12	0.3	0.02	1	0	192.83	638.50	15.00	250.81
15	12	0.3	0.02	1.4	0	432.52	1856.24	20.69	643.90
16	12	0.3	0.05	0.6	0	123.57	250.33	21.67	84.96
17	12	0.3	0.05	1	0	111.06	472.00	15.77	139.42
18	12	0.3	0.05	1.4	0	117.48	301.11	13.58	100.32
19	12	0.4	0.02	0.6	0	155.81	529.74	28.43	141.83
20	12	0.4	0.02	1	0	195.74	481.58	11.45	206.51
21	12	0.4	0.02	1.4	0	159.02	544.37	12.26	172.64
22	12	0.4	0.05	0.6	0	93.34	249.68	18.68	71.96
23	12	0.4	0.05	1	0	127.08	353.24	15.33	115.65
24	12	0.4	0.05	1.4	0	148.48	331.85	37.86	102.55

FIGURE 3: An illustration of the proposed mutation.

FIGURE 4: An example of the new fireflies' generation.

FIGURE 5: Local search scheme.

6.2. Experimental Results

6.2.1. Comparisons of EDFA with CPLEX. The efficiency and effectiveness of the proposed algorithm will be evaluated in this section by comparing the computation result of the EDFA with the optimal result obtained by solving the same small-scaled instances by the proposed MILP in CPLEX with a 1-h run time limitation. Table 6 shows the summary results of 240 instances by five replications for each parameter combination, which indicates that the proposed EDFA can obtain the exact solutions for all instances. Because the computation times of the EDFA are less than 2.5 s in all instances, which outperformed the CPLEX solver CPU times, we did not include EDFA time in the table. The relative percentage deviation for small-scaled instances is calculated by the following formula:

$$RPD = \frac{|Cmax_{EDFA} - Cmax_{CPLEX}|}{Cmax_{CPLEX}} \times 100 \quad (24)$$

Figure 8 shows that the EDFA is superior to well-known dispatching rules, which are first in first out (FIFO), shortest processing time (SPT), and longest processing time (LPT) for all small-scaled instances with respect to the RPD.

6.2.2. Comparisons of EDFA with Other Algorithms. To evaluate the proposed EDFA performance, different methods that have been proposed in the literature for solving a similar problem were investigated in this section. The first method is called ERD-LPT to minimize the makespan for a set of jobs on the SMSP with flexible periodic maintenances [24]. The second method is EDFA for solving the transport salesman problem (TSP) [40], which is related to the addressed problem in this work if the constraints of the maintenance and the release dates are ignored; the problem collapses to TSP where nodes correspond to jobs and edges correspond to setup times. For the comparison clarity, the second method was renamed to (O-EDFA). The third method is particle swarm optimization (PSO), which is similar to the proposed

FIGURE 6: Flowchart of the EDFA.

FIGURE 7: Interaction plot of the EDFA parameters.

FIGURE 8: Box plot compression of the RPD for small-scaled instances.

FIGURE 9: Tukey differences of means for RPD. If an interval does not contain zero, the corresponding means are significantly different.

EDFA with corresponding the Light absorption coefficient (γ) = 0 [36].

Tables 7 and 8 show the comparisons between the proposed EDFA (P-EDFA), PSO, ERD-LPT, and O-EDFA for large-scaled instances, where “max”, “min”, and “ave” are the maximum, minimum, and the average of the relative percentage deviation RPD, respectively, from the lower bound calculated from Properties 1 and 2 (Section 2.3). Tables 7 and 8 show that P-EDFA have a better performance than other methods.

The relative percentage deviation is calculated using the following formula:

$$RPD = \frac{|C_{\max_{Method}} - C_{\max_{Lower\ Bound}}|}{C_{\max_{Lower\ Bound}}} \times 100 \quad (25)$$

Once the experimental results were presented, a statistical test has been done to check if there was a significant difference between the performance of the four methods (ERD-LPT, PSO, O-EDFA, P-EDFA) or not. Table 9 demonstrates the result of the statistical test which is the one-way ANOVA analysis test. From Table 9, the P value is smaller than α value, which is 0.05 (significance level), referring that there is

a significant difference between the performance of the four methods.

Figure 9 demonstrates the Tukey simultaneous 95% confidence interval test. The aim of applying this test is to compare the means of the performance of the four methods and check if there are significant differences from each other. If an interval does not contain zero, the corresponding means will significantly differ from each other. In Figure 9 all intervals do not contain zero, which is strong evidence that the performances of the four methods are significantly different.

Finally, some sensitive analyses are performed for assessing the behaviour of algorithms versus different conditions. The interaction between the RPD values versus the number of jobs, setup time factor, maintenance interval factors, release date factor, and maintenance time factor is shown in Figures 10(a), 10(b), 10(c), 10(d), and 10(e), respectively.

7. Conclusion and Results Discussion

Herein, an EDFA was developed to minimize the makespan for a single machine with periodic maintenance,

TABLE 7: Comparisons of EDEFA with other algorithms for first set of instances.

No.	Generation conditions					Max				RPD AVR				Min			
	n	t	m	R	R	ERD-LPT	PSO	O-EDFA	P-EDFA	ERD-LPT	PSO	O-EDFA	P-EDFA	ERD-LPT	PSO	O-EDFA	P-EDFA
1	12	0.3	0.02	0.6	0.6	0.26	0.24	0.16	0.13	0.19	0.18	0.14	0.11	0.15	0.16	0.11	0.09
2	12	0.3	0.02	1	1	0.34	0.39	0.33	0.17	0.22	0.34	0.27	0.12	0.12	0.29	0.23	0.05
3	12	0.3	0.02	1.4	1.4	0.07	0.27	0.20	0.07	0.04	0.16	0.11	0.03	0.00	0.06	0.06	0.00
4	12	0.3	0.05	0.6	0.6	0.37	0.35	0.36	0.21	0.35	0.34	0.31	0.20	0.33	0.31	0.25	0.18
5	12	0.3	0.05	1	1	0.51	0.51	0.51	0.34	0.37	0.45	0.41	0.29	0.28	0.42	0.34	0.18
6	12	0.3	0.05	1.4	1.4	0.23	0.39	0.35	0.21	0.17	0.27	0.21	0.10	0.12	0.20	0.14	0.03
7	12	0.4	0.02	0.6	0.6	0.17	0.23	0.16	0.10	0.14	0.16	0.13	0.08	0.11	0.11	0.09	0.06
8	12	0.4	0.02	1	1	0.43	0.45	0.40	0.32	0.27	0.37	0.31	0.17	0.16	0.31	0.25	0.09
9	12	0.4	0.02	1.4	1.4	0.18	0.24	0.15	0.07	0.07	0.16	0.11	0.03	0.00	0.04	0.02	0.00
10	12	0.4	0.05	0.6	0.6	0.29	0.37	0.34	0.15	0.24	0.27	0.23	0.14	0.20	0.20	0.19	0.13
11	12	0.4	0.05	1	1	0.45	0.54	0.51	0.38	0.34	0.45	0.38	0.24	0.18	0.35	0.26	0.09
12	12	0.4	0.05	1.4	1.4	0.23	0.26	0.18	0.10	0.09	0.20	0.12	0.04	0.01	0.09	0.04	0.01
13	15	0.3	0.02	0.6	0.6	0.29	0.35	0.30	0.13	0.19	0.28	0.23	0.11	0.14	0.21	0.17	0.10
14	15	0.3	0.02	1	1	0.51	0.52	0.47	0.22	0.23	0.49	0.43	0.16	0.10	0.45	0.38	0.10
15	15	0.3	0.02	1.4	1.4	0.27	0.35	0.31	0.10	0.08	0.24	0.21	0.03	0.00	0.12	0.05	0.00
16	15	0.3	0.05	0.6	0.6	0.40	0.46	0.44	0.24	0.31	0.44	0.41	0.22	0.23	0.40	0.38	0.20
17	15	0.3	0.05	1	1	0.52	0.64	0.57	0.19	0.28	0.52	0.45	0.16	0.17	0.44	0.38	0.14
18	15	0.3	0.05	1.4	1.4	0.38	0.38	0.28	0.05	0.09	0.29	0.24	0.03	0.00	0.23	0.20	0.00
19	15	0.4	0.02	0.6	0.6	0.36	0.36	0.27	0.11	0.17	0.30	0.24	0.09	0.11	0.24	0.19	0.08
20	15	0.4	0.02	1	1	0.40	0.45	0.43	0.23	0.21	0.42	0.40	0.11	0.12	0.40	0.37	0.05
21	15	0.4	0.02	1.4	1.4	0.19	0.45	0.40	0.18	0.09	0.27	0.22	0.05	0.01	0.18	0.10	0.00
22	15	0.4	0.05	0.6	0.6	0.37	0.37	0.33	0.14	0.23	0.33	0.24	0.14	0.19	0.23	0.20	0.13
23	15	0.4	0.05	1	1	0.45	0.53	0.49	0.17	0.21	0.47	0.42	0.14	0.13	0.45	0.35	0.12
24	15	0.4	0.05	1.4	1.4	0.27	0.27	0.26	0.05	0.07	0.22	0.23	0.02	0.00	0.15	0.20	0.00

TABLE 8: Comparisons of EDFA with other algorithms for second set of instances.

No.	Generation conditions				Max				RPD				Min			
	n	t	m	R	ERD-LPT	PSO	O-EDFA	P-EDFA	ERD-LPT	PSO	O-EDFA	P-EDFA	ERD-LPT	PSO	O-EDFA	P-EDFA
1	12	0.3	0.02	0.6	0.69	0.62	0.55	0.32	0.55	0.54	0.47	0.27	0.38	0.47	0.38	0.15
2	12	0.3	0.02	1	0.33	0.41	0.32	0.19	0.23	0.32	0.25	0.09	0.13	0.26	0.16	0.01
3	12	0.3	0.02	1.4	0.30	0.30	0.26	0.10	0.07	0.16	0.10	0.03	0.00	0.04	0.00	0.00
4	12	0.3	0.05	0.6	0.88	0.83	0.70	0.48	0.77	0.69	0.57	0.42	0.56	0.56	0.49	0.34
5	12	0.3	0.05	1	0.50	0.54	0.41	0.25	0.36	0.44	0.36	0.18	0.29	0.34	0.30	0.10
6	12	0.3	0.05	1.4	0.15	0.22	0.20	0.05	0.07	0.19	0.12	0.02	0.01	0.15	0.07	0.00
7	12	0.4	0.02	0.6	0.55	0.51	0.41	0.28	0.46	0.42	0.35	0.23	0.37	0.35	0.30	0.20
8	12	0.4	0.02	1	0.30	0.34	0.28	0.08	0.17	0.27	0.20	0.05	0.07	0.20	0.13	0.01
9	12	0.4	0.02	1.4	0.17	0.22	0.13	0.04	0.06	0.15	0.10	0.02	0.01	0.03	0.04	0.01
10	12	0.4	0.05	0.6	0.62	0.59	0.52	0.35	0.56	0.53	0.48	0.33	0.50	0.49	0.44	0.32
11	12	0.4	0.05	1	0.36	0.41	0.38	0.17	0.29	0.35	0.28	0.08	0.16	0.28	0.17	0.01
12	12	0.4	0.05	1.4	0.22	0.24	0.12	0.04	0.06	0.18	0.09	0.01	0.01	0.08	0.05	0.00
13	15	0.3	0.02	0.6	0.69	0.69	0.65	0.38	0.60	0.64	0.58	0.34	0.53	0.62	0.53	0.30
14	15	0.3	0.02	1	0.44	0.51	0.46	0.16	0.22	0.43	0.40	0.10	0.13	0.33	0.36	0.06
15	15	0.3	0.02	1.4	0.19	0.23	0.23	0.02	0.04	0.21	0.19	0.01	0.00	0.19	0.15	0.00
16	15	0.3	0.05	0.6	0.70	0.89	0.84	0.52	0.63	0.78	0.75	0.46	0.56	0.70	0.72	0.43
17	15	0.3	0.05	1	0.49	0.59	0.58	0.29	0.34	0.53	0.50	0.19	0.19	0.49	0.46	0.16
18	15	0.3	0.05	1.4	0.25	0.36	0.28	0.06	0.10	0.32	0.26	0.03	0.02	0.25	0.24	0.01
19	15	0.4	0.02	0.6	0.62	0.63	0.58	0.35	0.51	0.60	0.53	0.31	0.39	0.54	0.48	0.28
20	15	0.4	0.02	1	0.38	0.42	0.42	0.13	0.18	0.38	0.37	0.07	0.06	0.32	0.34	0.02
21	15	0.4	0.02	1.4	0.22	0.25	0.28	0.02	0.06	0.22	0.23	0.01	0.00	0.20	0.18	0.00
22	15	0.4	0.05	0.6	0.75	0.77	0.67	0.43	0.63	0.68	0.61	0.39	0.51	0.55	0.50	0.36
23	15	0.4	0.05	1	0.50	0.59	0.55	0.21	0.26	0.49	0.47	0.13	0.16	0.40	0.43	0.08
24	15	0.4	0.05	1.4	0.39	0.39	0.36	0.11	0.10	0.30	0.26	0.04	0.01	0.24	0.20	0.00

TABLE 9: Analysis of the significance of different methods.

Source	DF	Adj SS	Adj MS	F-value	P-value
Method	3	6.813	2.27088	82.25	0.001
Error	956	26.395	0.02761		
Total	959	33.207

FIGURE 10: Box plot of the RPD for large-scaled instances with different parameters.

sequence-dependent setup times, and release date constraint. The proposed algorithm has two distinctive features, including the inversion mutation method for creating the new generation of fireflies and a local search. In total, 420 random instances were generated to test the efficiency and effectiveness of the EDFA. The EDFA performances were compared with the CPLEX and PSO. The results revealed

that the proposed EDFA showed the optimal solution as generated in the CPLEX solver for most of the small-scaled instances. Regarding the CPU time, the results indicate that the EDFA outperforms the CLPEX in execution time for all problem instances. For large-scaled instances, the EDFA shows excellent performance and it outperformed the PSO algorithm.

Data Availability

The data used to support the findings of this study are available from the coauthor Ammar Qamhan (435108378@student.ksu.edu.sa) upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group No. RG-1439- 033.

References

- [1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer Science & Business Media, 2012.
- [2] M. C. Vélez-Gallego, J. Maya, and J. R. Montoya-Torres, "A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan," *Computers & Operations Research*, vol. 73, pp. 132–140, 2016.
- [3] A. Allahverdi, J. N. D. Gupta, and T. Aldowaisan, "A review of scheduling research involving setup considerations," *Omega*, vol. 27, no. 2, pp. 219–239, 1999.
- [4] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *European Journal of Operational Research*, vol. 246, no. 2, pp. 345–378, 2015.
- [5] A. Allahverdi, C. T. Ng, T. C. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 985–1032, 2008.
- [6] H. Xu, Z. Lü, A. Yin, L. Shen, and U. Buscher, "A study of hybrid evolutionary algorithms for single machine scheduling problem with sequence-dependent setup times," *Computers & Operations Research*, vol. 50, pp. 47–60, 2014.
- [7] H. Xu, Z. Lü, and T. C. Cheng, "Iterated local search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness," *Journal of Scheduling*, vol. 17, no. 3, pp. 271–287, 2014.
- [8] Q. X. Guo and L. X. Tang, "An improved scatter search algorithm for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times," *Applied Soft Computing*, vol. 29, pp. 184–195, 2015.
- [9] G. Kirlik and C. Oguz, "A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine," *Computers & Operations Research*, vol. 39, no. 7, pp. 1506–1520, 2012.
- [10] C. Chao and C. Liao, "A discrete electromagnetism-like mechanism for single machine total weighted tardiness problem with sequence-dependent setup times," *Applied Soft Computing*, vol. 12, no. 9, pp. 3079–3087, 2012.
- [11] K.-C. Ying, S.-W. Lin, and C.-Y. Huang, "Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic," *Expert Systems with Applications*, vol. 36, no. 3, pp. 7087–7092, 2009.
- [12] M. F. Tasgetiren, Q.-K. Pan, and Y.-C. Liang, "A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times," *Computers & Operations Research*, vol. 36, no. 6, pp. 1900–1915, 2009.
- [13] W. Bożejko, "Parallel path relinking method for the single machine total weighted tardiness problem with sequence-dependent setups," *Journal of Intelligent Manufacturing*, vol. 21, no. 6, pp. 777–785, 2010.
- [14] D. Anghinolfi and M. Paolucci, "A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times," *European Journal of Operational Research*, vol. 193, no. 1, pp. 73–85, 2009.
- [15] M. A. González, J. J. Palacios, C. R. Vela, and A. Hernández-Arauzo, "Scatter search for minimizing weighted tardiness in a single machine scheduling with setups," *Journal of Heuristics*, vol. 23, no. 2-3, pp. 81–110, 2017.
- [16] C.-J. Liao and H.-C. Juan, "An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups," *Computers & Operations Research*, vol. 34, no. 7, pp. 1899–1909, 2007.
- [17] L. Bianco, S. Ricciardelli, G. Rinaldi, and A. Sassano, "Scheduling tasks with sequence-dependent processing times," *Naval Research Logistics*, vol. 35, no. 2, pp. 177–184, 1988.
- [18] J. R. Montoya-torres, F. González-Solano, and M. Soto-ferrari, "Deterministic machine scheduling with release times and sequence-dependent setups using random-insertion heuristics," *Journal of Advanced Operations Management*, vol. 4, pp. 4–26, 2012.
- [19] R. Ramezani, M. Saidi-Mehrabad, and P. Fattahi, "MIP formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints," *Journal of Manufacturing Systems*, vol. 32, no. 2, pp. 392–401, 2013.
- [20] Y.-C. Chen, "An optimal production and inspection strategy with preventive maintenance error and rework," *Journal of Manufacturing Systems*, vol. 32, no. 1, pp. 99–106, 2013.
- [21] A. Garg and S. G. Deshmukh, "Maintenance management: literature review and directions," *Journal of Quality in Maintenance Engineering*, vol. 12, no. 3, pp. 205–238, 2006.
- [22] W.-J. Chen, "Minimizing number of tardy jobs on a single machine subject to periodic maintenance," *Omega*, vol. 37, no. 3, pp. 591–599, 2009.
- [23] T. Wang and D. Xu, "Single-machine scheduling with workload-dependent maintenance duration to minimize maximum lateness," *Mathematical Problems in Engineering*, vol. 2015, Article ID 732437, 5 pages, 2015.
- [24] W. W. Cui and Z. Lu, "Minimizing the makespan on a single machine with flexible maintenances and jobs' release dates," *Computers & Operations Research*, vol. 80, pp. 11–22, 2017.
- [25] C. Low, M. Ji, C.-J. Hsu, and C.-T. Su, "Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance," *Applied Mathematical Modelling*, vol. 34, no. 2, pp. 334–342, 2010.
- [26] J. Shen and K. Zhu, "An uncertain single machine scheduling problem with periodic maintenance," *Knowledge-Based Systems*, vol. 144, pp. 32–41, 2018.
- [27] M. Yazdani, S. M. Khalili, M. Babagolzadeh, and F. Jolai, "A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach," *Journal of Computational Design and Engineering*, vol. 4, no. 1, pp. 46–59, 2017.

- [28] F. Zammori, M. Braglia, and D. Castellano, "Harmony search algorithm for single-machine scheduling problem with planned maintenance," *Computers & Industrial Engineering*, vol. 76, pp. 333–346, 2014.
- [29] M. Bertolini, D. Mezzogori, and F. Zammori, "Comparison of new metaheuristics, for the solution of an integrated jobs-maintenance scheduling problem," *Expert Systems with Applications*, vol. 122, pp. 118–136, 2019.
- [30] F. Ángel-Bello, A. Álvarez, J. Pacheco, and I. Martínez, "A single machine scheduling problem with availability constraints and sequence-dependent setup costs," *Applied Mathematical Modelling*, vol. 35, no. 4, pp. 2041–2050, 2011.
- [31] F. Ángel-Bello, A. Álvarez, J. Pacheco, and I. Martínez, "A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times," *Computers & Mathematics with Applications*, vol. 61, no. 4, pp. 797–808, 2011.
- [32] J. Pacheco, F. Ángel-Bello, and A. Álvarez, "A multi-start tabu search method for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times," *Journal of Scheduling*, vol. 16, no. 6, pp. 661–673, 2013.
- [33] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.
- [34] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36–50, 2013.
- [35] E. Osaba, X. Yang, F. Diaz, E. Onieva, A. D. Masegosa, and A. Perallos, "A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy," *Soft Computing*, vol. 21, no. 18, pp. 5295–5308, 2017.
- [36] I. Fister, I. Fister Jr., X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, no. 1, pp. 34–46, 2013.
- [37] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 301–305, 2014.
- [38] L. Hou, C. Zhao, C. Wu, S. Moon, and X. Wang, "Discrete firefly algorithm for scaffolding construction scheduling," *Journal of Computing in Civil Engineering*, vol. 31, no. 3, Article ID 04016064, 2016.
- [39] E. Osaba, R. Carballedo, X. Yang, and F. Diaz, "An evolutionary discrete firefly algorithm with novel operators for solving the vehicle routing problem with time windows," *Nature-Inspired Computation in Engineering*, vol. 637, pp. 21–41, 2016.
- [40] G. K. Jati and Suyanto, "Evolutionary discrete firefly algorithm for travelling salesman problem," *International Conference on Adaptive and Intelligent Systems*, vol. 6943, pp. 393–403, 2011.

Submit your manuscripts at
www.hindawi.com