Hybridized Particle Swarm Optimization for Aircraft Inspection Check and Continuous Airworthiness Maintenance Program

Asyraf Nur Adianto and Nurhadi Siswanto
Institut Teknologi Sepuluh Nopember

Keywords: Aircraft Maintenance, Maintenance Scheduling, Particle Swarm Optimization, Greedy Randomized Adaptive Search Procedures.

Abstract: This research compares the performances of two metaheuristic methods: Particle Swarm Optimization (PSO) and a hybridized PSO method with Greedy Randomized Adaptive Search Procedures (GRASP) for solving aircraft maintenance problems (AMP). In this problem, AMP consists of two different maintenance activity types: inspection and continuous airworthiness maintenance programs (CAMP). The purpose of this paper is to determine the number of periods that the aircraft needs to be maintained and which inspection and CAMP tasks need to be done in each period. The problem is NP-Hard in nature, so that metaheuristic methods are used to make sure the optimization process can be solved quickly. Computational experiments are performed by using 16 conditions, and four randomly generated dataset instances. The computational experiment result shows that PSO-GRASP outperforms PSO for a larger planning horizon.

1 INTRODUCTION

Aviation industries obtain their revenues based on the number of passengers they serve by using their aircraft (Gargiulo, Pascar, & Venticinque, 2013). However, sometimes, the aircraft need to be maintained. When an aircraft needs to be maintained, it cannot be operated to earn revenues for the owners. All airlines hope that their aircraft have high utilization to serve their customers by minimizing the number of maintenances for their aircraft without violating any regulation related to aircraft airworthiness. Research related to those problems is termed aircraft maintenance / aeronautical maintenance (Gargiulo et al., 2013; Han, Cao., & Yang, 2012).

To the best of our knowledge, the research related to aeronautical / aircraft maintenance only considered inspection check schedule (example A-Check, C-Check, D-Check) for the aircraft to be scheduled in the form of aircraft maintenance routing problem (AMRP). Several researches have been conducted in the area of AMRP, such as: Al-Thani, Ben Ahmed, and Haouari (2016), Eltoukhy, Chan, Chung, and Niu (2018), Ezzinbi, Sarhani, El Afia, and Benadada (2014) Gopalan and Talluri (1998) Liang, Chaovalitwongse, Huang, and Johnson (2011) and Safaei and Jardine (2018). In practice, there are other maintenance activities that need to be considered by the airlines, like the Continuous Airworthiness Maintenance Program (CAMP). The difference between inspection check and CAMP is that inspection check does visual checking on some components (Nickles, Him, Koenig, Gramopadhye, & Melloy, 1999) and determines whether the components need to be replaced or further maintained (this maintenance activity could be categorized as unscheduled maintenance), while CAMP does minor maintenance based on manufacturer direction and must be done regularly, known as scheduled maintenance (U.S. Department of Transportation & Federal Aviation Administration, 2016). Both maintenance categories use maintenance resources and could make the airlines bear the cost, but those tasks should be done by them in order to ensure their aircrafts' airworthiness. These problems encourage the airline to design a maintenance schedule that can either reduce its costs or maximize the aircraft's' utility.

Given the nature of aircraft maintenance, we consider this problem based on inspection and CAMP

Adianto, A. and Siswanto, N.
Hybridized Particle Swarm Optimization for Aircraft Inspection Check and Continuous Airworthiness Maintenance Program.
DOI: 10.5220/0009403900290035
In *Proceedings of the 1st International Conference on Industrial Technology (ICONIT 2019)*, pages 29-35
ISBN: 978-989-758-434-3

as NP-Hard. To tackle this problem, metaheuristic methods such as Particle Swarm Optimization (PSO) are used as an optimal solution search engine. PSO has a broad experience in resolving maintenance schedule. In this research, we would like to test hybridized PSO-GRASP to find whether it could solve the problem and outperform the PSO method.

2 PROBLEM DEFINITION

Consider a set of $\bar{I}$ inspection tasks $\{1, 2, \dots, I\}$ and a set of $\bar{R}$ CAMP tasks $\{1, 2, \dots, R\}$, which must be scheduled on a single aircraft. Each set has notable data, such as next do time, interval time, threshold time, and duration (only for the set I). We consider that in the set, I and R have two different time units, such as calendar days and flight hours. Both tasks must be scheduled on a set of $\bar{T}$ periods $\{1, 2, \dots, T\}$, which could be calculated first by using planning horizon h, Both next do nx_i for inspection task and ny_r for a CAMP task, and both intervals ix_i for inspection task and iy_r for a CAMP task, with the following equation (1(1).

$$\mathrm{T} = \left[\frac{h}{\min\left(\min\limits_{\mathrm{i}\in\bar{\mathrm{I}}}(\mathrm{ix_i}), \min\limits_{\mathrm{r}\in\bar{\mathrm{R}}}(\mathrm{iy_r})\right)}\right] + 1 \tag{1}$$

In order to use the model, available data must be converted from two different time units into one calendar day's unit. For next do, interval, threshold, and duration data of inspection tasks are converted into the following equations (2), (4), (6), and (8), consecutively. For next do, interval and threshold data of CAMP tasks are converted by the following equations (3), (5) and (7), respectively.

$$\mathrm{ncx_{i,1}} \begin{cases} \mathrm{nx_i} & \text{, calendar days} \\ \mathrm{nx_i} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{2}$$

$$\mathrm{ncy_{r,1}} \begin{cases} \mathrm{ny_r} & \text{, calendar days} \\ \mathrm{ny_r} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{3}$$

$$\mathrm{tcx_i} \begin{cases} \mathrm{tx_i} & \text{, calendar days} \\ \mathrm{tx_i} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{4}$$

$$\mathrm{tcy_r} \begin{cases} \mathrm{ty_r} & \text{, calendar days} \\ \mathrm{ty_r} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{5}$$

$$\mathrm{icx_i} \begin{cases} \mathrm{ix_i} & \text{, calendar days} \\ \mathrm{ix_i} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{6}$$

$$\mathrm{icy_r} \begin{cases} \mathrm{iy_r} & \text{, calendar days} \\ \mathrm{iy_r} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{7}$$

$$\mathrm{dcx_i} \begin{cases} \mathrm{dx_i} & \text{, calendar days} \\ \mathrm{dx_i} \;/\; \mathrm{u} & \text{, flight hours} \end{cases} \tag{8}$$

Those indexes and converted variables are used to determine the maximum utilization of an aircraft by following the equation below.

$$\max \mathrm{Z} = \frac{\sum_{t\,\in\,\bar{\mathrm{T}}}(c_{t+1} - c_t)}{c_{T+1} + m_T} \tag{9}$$

$$\sum\nolimits_{i\in\bar{I}} x_{i,t} \geq 1,\ t\ \forall\ \bar{T} \tag{10}$$

$$\sum\nolimits_{r\in\bar{R}} y_{r,t} \geq 1,\ t\ \forall\ \bar{T} \tag{11}$$

$$ncx_{i,(t+1)} - ncx_{i,t} \geq 0,\ i\ \forall\ \bar{I}\ ,\ t\ \forall\ \bar{T} \tag{12}$$

$$ncy_{r,(t+1)} - ncy_{r,t} \geq 0,\ r\ \forall\ \bar{R}\ ,\ t\ \forall\ \bar{T} \tag{13}$$

$$m_t - \sum\nolimits_{i\in\bar{I}} \left(x_{i,t}dcx_i\right) = 0,\ t\ \forall\ \bar{T} \tag{14}$$

$$ndx_{i,t} - x_{i,t}(c_{t+1} + icx_i + m_t) = 0,\ i\ \forall\ \bar{I}\ ,\ t\ \forall\ \bar{T} \tag{15}$$

$$ntx_{i,t} - \left(1 - x_{i,t}\right)\left(ncx_{i,t} + m_t\right) = 0,\ i\ \forall\ \bar{I}\ ,\ t\ \forall\ \bar{T} \tag{16}$$

$$ncx_{i,(t+1)} - \left(ndx_{i,t} + ntx_{i,t}\right) = 0,\ i\ \forall\ \bar{I},\ t\ \forall \bar{T} \tag{17}$$

$$ndy_{r,t} - y_{r,t}\left(c_{t+1} + icy_r + m_t\right) = 0,\ r\ \forall\ \bar{R},\ t\ \forall\ \bar{T} \tag{18}$$

$$nty_{r,t} - \left(1 - y_{r,t}\right)\left(ncy_{r,t} + m_t\right) = 0,\ r\ \forall\ \bar{R},\ t\ \forall\ \bar{T} \tag{19}$$

$$ncy_{\mathrm{r},(t+1)} - \left(ndx_{i,t} + ntx_{i,t}\right) = 0,\ r\ \forall\ \bar{R},\ t\ \forall\ \bar{T} \tag{20}$$

$$x_{i,t}ncx_{i,t} - c_{t+1} \geq 0,\ i\ \forall\ \bar{I},\ t\ \forall\ \bar{T} \tag{21}$$

$$y_{r,t}ncy_{r,t} - c_{t+1} \geq 0,\ r\ \forall\ \bar{R},\ t\ \forall\ \bar{T} \tag{22}$$

$$ncx_{i,(t+1)} - tcx_i \geq 0,\ i\ \forall\ \bar{I},\ t\ \forall\ \bar{T} \tag{23}$$

$$ncy_{r,(t+1)} - tcy_r \geq 0,\ r\ \forall\ \bar{R},\ t\ \forall\ \bar{T} \tag{24}$$

$$c_1 = 0 \tag{25}$$

$$x_{i,t}, y_{r,t} \in \{0,1\} \tag{26}$$

Equation (9) is the objective function of the model. Equation (10) and (11) ensure the decision has at least one task to be done in each period for inspection task and CAMP task, respectively. Equation (12) and (13) ensure generated next do data in $t + 1$ period always have greater or equal value with next do data in t period for inspection task and CAMP task, respectively. Equation (14) calculates total maintenance duration equal to all maintenance durations of inspection task that has to be done in t period. Equation (17) and (20) calculate next do data of $t + 1$ both of inspection task and CAMP task, respectively. Equation (17) and (20) could be calculated by equation (15) and (18) for calculating the difference value when the task is being done in t period, while equation (16) and (19) for calculating the difference value when the task is not being done in t period. Equation (21) and (22) ensure next do data of t period always have greater or equal value with current days of $t + 1$ period for inspection task and CAMP task, respectively. Equation (23) and (24) ensure next do data for $t + 1$ period always have greater or equal to threshold data for inspection task and CAMP task, respectively. Equation (25) ensures the current days of period 1 have a value equal to 0. Equation (26) ensures the decision variable for both the inspection task, and the CAMP task is filled with binary value.

3 MODEL DEVELOPMENT

In this research, we focused on solving the given problem using two metaheuristic methods, pure PSO and hybrid PSO-GRASP. In this following sub-section, the PSO and GRASP will be first presented, and then their combination to create the PSO-GRASP is discussed.

3.1 Particle Swarm Optimization

PSO is a swarm intelligent metaheuristic method based on how a flock of birds tracks down their prey (Santosa & Ai, 2017). Every bird will fly to the best location based on all information shared by other birds in that flock, including itself. To reach the best objective function for each iteration, each individual has to share their objective function value and conclude all information into one best. In order to make an individual move to its targeted value, the individual velocity needs to be updated as in the following equations (27) and (28).

$$v_{p,(a+1)}=\omega v_{p,a}+b_1r_1\left(pbest_{p,a}-x_{p,a}\right)+b_2r_2\left(gbest_a-x_{p,a}\right) \quad (27)$$

$$x_{p,(a+1)}=x_{p,a}+v_{p,(a+1)} \quad (28)$$

Both equations consist of the following index:

- Set $\bar{P}$ individual or population {1, 2, …, P}
- Set $\bar{A}$ iteration {1, 2, …, I}

Both equations use the following related variables:

- $v_{p,}$ a: velocity of p population of an iteration
- $x_{p,}$ a: generated solution in p population of an iteration
- $pbest_{p,}$ a: best-generated solution of p population in an iteration
- $gbest_a$: best-generated solution of all population in an iteration
- r_2 : randomized number with decimal value from 0 to 1

In equation (27), there are some parameters that can be set up manually, which are:

- b_1 : ratio to take the best-generated solution of each population that would affect the velocity variable
- b_2 : ratio to take the best-generated solution of all population that would affect velocity variable
- ω : ratio to take an earlier iteration of generated solution that would affect velocity variable

In equation (27), we will calculate velocity of $a+1$ iteration by determining a certain portion of velocity in an earlier iteration, a certain portion of the gap between the generated solution of the last iteration with best generated solution of p population, and a certain portion of the gap between the generated solution of the last iteration with the best generated solution of all population. Using those velocity values, we can update the new position of each individual using equation (28) by adding the generated solution of the last iteration with calculated velocity from equation (27).

3.2 Greedy Randomized Adaptive Search Procedures (GRASP)

GRASP is a metaheuristic method designed for helping other metaheuristic methods to find an optimal solution efficiently. This method was developed by Feo dan Resende (1995) and had two phases, the construction phase, and the local search phase. Both phases are developed by specifically identifying the decision variable of the problem and constructing the encoded form of those decision variables.

Table 1: Decision Variable Transformed Form

	Per 1	Per 2	...	Per T
$In-1$	1	0	...	1
$In-2$	0	1	...	1
...	...	...	...	...
$In-I$	0	0	...	0
$Ca-1$	0	0	...	1
$Ca-2$	1	1	...	1
...	...	...	...	...
$Ca-R$	1	0	...	1

In this research, decision variables of the problem have two-dimensional matrix; $x_{i,t}$, and $y_{r,t}$, which represents each inspection and CAMP task, respectively, at each period. Metaheuristic methods must generate those variables, as illustrated in Table 1. Using this form, the metaheuristic method sometimes may generate a solution that violates some constraints of the problem. To avoid generating an invalid solution, GRASP is implemented at both metaheuristic methods and creates a new decision variable form filled with inspection task ID, as illustrated in Table 2.

Table 2: Decision Variable with GRASP Transformed Form

	Per 1	Per 2	...	Per T
$In - ref$	3	4	...	2

3.3 Hybrid PSO-GRASP

PSO method must generate data by filling the form as described in the previous section with the decimal value from 0 to 1, as illustrated in Table 3. Those decimal values are converted into inspection task ID by finding the nearest multiplication of decimal value with I. For example, if $I = 6$, the generated form in Table 3 will be transformed, as illustrated in Table 2.

Table 3: Decision Variable with GRASP Form

	Per 1	Per 2	...	Per T
In-ref	0.439	0.72	...	0.272

To convert generated decision variables with GRASP, as in Table 2, into the original decision variable form as in Table 3, we use the local search method, which satisfies equation (29) and (30). Both equations ensure the selected inspection and CAMP tasks that have to be done next must-have lower or are equal to the referred inspection task's next values.

$$ncx_{ref_*,t} - x_{i,t} ncx_{i,t} \geq 0 \qquad (29)$$

$$ncx_{ref_*,t} - y_{r,t} ncy_{r,t} \geq 0 \qquad (30)$$

3.4 Proposed Model

The proposed algorithm of Hybrid PSO-GRASP is depicted in Figure 1. First, we generate the referred inspection task using a random number generator and convert them using the GRASP method for the first iteration, using PSO-GRASP later.

For the iteration, until the maximum iteration number is reached, we update the max time of each maintenance period in each generated solution, then construct all inspection and CAMP tasks that need to be ordered. Using the ordered task, we compute the maintenance duration needed. Using the ordered task and the maintenance duration in each period, we calculate when inspection and CAMP tasks should be done and then update the utilization of the aircraft. Using the ordered tasks, we validate the generated solution each period. Each valid solution will be marked and will be used to calculate the objective function value of the generated solution.

The objective function value is calculated from utilization value for a valid solution in each period and penalty value for an invalid solution in each period. Each period's penalty value has a different number. Increased period value means decreased penalty value, and the highest penalty value will be in the first period. Penalty value will be calculated

```
Generate referred inspection tasks.
For ( from p ← 1 to p ← $\bar{P}$ )
    For ( from t ← 1 to t ←$\bar{T}$ )
        Update the max period of time.
        For ( from i ← 1 to i ← $\bar{I}$ )
            Construct task order of all inspection tasks.
        End
        For ( from r ← 1 to r ← $\bar{R}$ )
            Construct task order of all CAMP tasks.
        End
        Update maintenance duration.
        Update next do value of the next period.
        Update utilization of the aircraft.
        Validate the generated solution.
    End
    Compute the objective function value for each
    solution population.
End
Save the maximum objective value of each generated
population
```

Figure 1: Pseudocode of PSO-GRASP Model

Following equation (31). When we have a maximum period $\bar{T}$ equal to 30 and the generated solution has an invalid solution at a few period $t = \{1, 3, 15\}$, then we have penalty$_1$ equal to 3000, penalty$_3$ equal to 2800, penalty$_{15}$ equal to 1600. From those values, we can conclude that the generated solution has a total penalty equal to 7400.

$$penalty_t - 100(\bar{T} - t + 1) = 0 \qquad (31)$$

The objective function value is calculated using calculated utilization and penalty value following equation (32). When the generated solution in p population calculates its objective function, the current value of the objective function will be added by summation of utilization time in each period that is marked as valid solution divided by next do time at last period $\bar{T}$ added by maintenance duration of last period $\bar{T}$ as described in equation (9). When the generated solution in p population and t period are marked as invalid solution, the current value of the objective function will be subtracted by penalty value, as described in equation (31).

Table 4: The Results by Varying Planning Horizon and Dataset Size Sensitivity Testing

ID	Plan Horizon	Group	PSO			PSO-GRASP		
			Last Maint. Finish Time	Obj Value	CPU Time	Last Maint. Finish Time	Obj Value	CPU Time
Test01	730	G1	453	-277	600.39	782	78.14	110.45
Test02		G2	267	-3650	600.64	781	78.11	181.19
Test03		G3	1008	-11680	600.25	1100	49	258.64
Test04		G4	807	-20440	600.15	923	48.43	390.59
Test05	1460	G1	639	-821	600.69	1776	77.03	197.65
Test06		G2	480	-2920	600.19	1673	77.35	337.11
Test07		G3	1839	-154760	600.85	2361	46.3	508.42
Test08		G4	1804	-143080	600.93	2362	43.4	744.06
Test09	2190	G1	856	-26280	601.53	2819	76.84	289.77
Test10		G2	759	-56940	600.53	2669	76.89	496.17
Test11		G3	2636	-575970	600.15	4171	41.99	600.32
Test12		G4	2528	-917610	600.93	4419	41.46	600.52

$$of\,value_p - \frac{\sum_{t\in\bar{T}} vld_{p,t}(c_{t+1} - c_t)}{c_{T+1}+m_T} + penalty_t\left(1 - vld_{p,t}\right) = 0 \qquad (32)$$

4 EXPERIMENTAL DESIGN

This research evaluates the performance of both the non-hybridized PSO method and PSO that have been hybridized with GRASP with four sets of randomized problems' data. These instances are generated on a different scale, from small scale to large scale, with five and 20 inspection tasks, combined with 500 and 1000 CAMP tasks. The combination of each dataset is described in Table 5. For example, Group G1 has a combination of five inspection tasks and 500 CAMP tasks. Moreover, PSO has three parameters that could be set up manually, consisting of c_1, c_2, and ω. In this experiment, we use the basic values of PSO parameters, which are $c_1 = c_2 = \omega = 1$.

Table 5: Dataset Instance

Group	Inspection Task	CAMP Task
G1	5	500
G2	5	1000
G3	20	500
G4	20	1000

5 COMPUTATIONAL RESULT

Both PSO and PSO-GRASP were executed until the objective value for the generated solution was labeled as a valid solution and reached the maximum iteration parameter. The other set condition was that the objective value is invalid, and the iteration process is stopped when CPU computational time is greater than or equal to 600 seconds. Using the parameters that have been set up before, we do two kinds of experiments. The first experiment is to do sensitivity testing on both varying planning horizon parameters and the size of the dataset. The second experiment is to do sensitivity testing on both iteration and population parameters.

The first experimental results are shown in Table 4. Increasing the planning horizon of the test has an impact on increasing CPU time in both PSO and PSO-GRASP methods. In the example for dataset G1 using PSO-GRASP, for planning horizon 730 days has 110.45 seconds of computational time, while for planning horizon 1460 days, the CPU time increases to 197.65 seconds, and for planning horizon, 2190 days CPU time usage increased to 289.77. For the other method, PSO used at least 600 seconds CPU time in all conditions. PSO always generates invalid solutions because, in every test, the objective values of PSO always have negative values, as we have explained in sub-section 3.4 so that the values are invalid solutions.

Experiments by using PSO-GRASP on the larger size as well as longer planning horizon datasets use more CPU time, either in inspection task number or CAMP task number. From all tests in this first experiment, 83.3% of the results show that PSO-GRASP uses lower CPU time compared with the PSO, while the other 16.6% have the same average CPU time as PSO.

Increasing the planning horizon of the tests can decrease objective function values. In the example for dataset G1 using PSO-GRASP, 730 days of planning horizon has an objective function value Equal to 78.14, 1460 days has 77.03, and 2190 days have

76.84. The same condition also applies for a larger size dataset, which mostly generates less objective function value, either in inspection task number or CAMP task number, both using PSO-GRASP. For 1460-day planning horizon, using PSO-GRASP, dataset G1 until G4 has objective function values equal to 77.03, 77.35, 46.3, and 43.4, consecutively. Based on these results, it can be concluded that increasing the number of CAMP tasks does not significantly change the objective function values

Table 6: The Results by Varying the Number of Iteration and Population Sensitivity Testing

ID	Iteration / Population Limitation	Group	PSO				PSO-GRASP			
			Last Maint. Finish Time	Obj Value	% Gap	CPU Time	Last Maint. Finish Time	Obj Value	% Gap	CPU Time
Test13	1000/100	G3	425	-305	-	600.44	1051	47.48	-	240.68
Test14	1000/200		467	-263	13.77%	600.32	861	53.08	11.79	475.16
Test15	2000/100		474	-256	16.07%	600.59	1146	49.92	5.14	481.61
Test16	2000/200		426	-304	0.33%	600.45	914	54.6	15.00	930.88

The second experiment explored only Dataset G3 with 730 days planning horizon, as shown in Table 6, by varying the number of iterations and population Test13 uses 1000 iterations and 100 populations as the baseline. Test 14 uses the same number of iterations as the one in Test 13; however, the number of populations of Test 14 is double the one of Test 13. Test 15 and Test 16 have the same number of iterations, 2000, but they differ in the number population.

Increasing the number of population and the number of iteration parameters escalates the CPU time and mostly increases the objective function values in both methods. In PSO-GRASP, doubling the number of populations significantly increased the objective function values with a 15% gap rather than doubling the number of iteration parameters with only a 5.14% gap.

Using PSO, increasing the number of iteration parameters significantly increases the objective function value with a 16.07% gap rather than that increasing the population parameter with a 13.77% gap and that increasing both iteration and population parameters with 0.33% gap. Even though the number of iterations and population parameters may affect PSO performance, we cannot make any accurate conclusion of CPU time usage because the objective function generated by the PSO method is always marked as an invalid solution and cannot be implemented to the real system.

6 CONCLUSIONS

In this paper, we presented a MILP model for optimizing aircraft maintenance scheduling problems, considering the inspection check task and CAMP task. Both tasks have to be considered by the airline because it could affect the airworthiness of their aircraft. Because of the NP-Hard nature of the problem, we developed PSO-GRASP metaheuristic methods to solve this problem in a reasonable time. We tested the model using four randomly generated datasets. We compared the metaheuristic models, PSO, and hybrid PSO-GRASP, based on objective function values and their CPU computational times.

The developed model could solve both small and large-scale datasets. Using a larger scale dataset, the result showed that the model could generate small objective function value, but it needs longer CPU time when tested in a similar parameter setting. Statistical analysis shows that the PSO-GRASP model is able to provide better performance than the PSO method without hybridization based on the objective function values. PSO cannot even provide a valid solution to this problem.

By using the larger values of the iterative and the population parameters, it makes PSO-GRASP work better, but, in the PSO, the changes in these parameters do not have any impacts on the solutions made and still provide an invalid solution. Moreover, by using these parameters or other parameters such as the planning horizon, the computational times are longer both for PSO and PSO-GRASP.

An interesting topic following this research is to develop the exact algorithm, whether using Mixed Integer Programming or Non-Linear Programming, using either small or larger similar datasets. Developing other metaheuristic methods, such as the Genetic Algorithm, Tabu Search, or Simulated Annealing, is also interesting for the next research agenda for solving this aircraft maintenance problem.

ACKNOWLEDGMENTS

The authors wish to acknowledge the funding of this research is from the Ministry of Research Technology and Higher Education, the Republic of Indonesia, through the Postgraduate Thesis Research Scheme Grant No. 781/PKS/ITS/2019.

REFERENCES

Al-Thani, N. A., Ben Ahmed, M., & Haouari, M. (2016). A model and optimization-based heuristic for the operational aircraft maintenance routing problem. Transportation Research Part C: Emerging Technologies, 72, 29–44. https://doi.org/10.1016/j.trc.2016.09.004

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers and Structures, 169, 1–12. https://doi.org/10.1016/j.compstruc.2016.03.001

Eltoukhy, A. E. E., Chan, F. T. S., Chung, S. H., & Niu, B. (2018). A model with a solution algorithm for the operational aircraft maintenance routing problem. Computers and Industrial Engineering, 120, 346–359. https://doi.org/10.1016 /j.cie.2018.05.002

Ezzinbi, O., Sarhani, M., El Afia, A., & Benadada, Y. (2014). A metaheuristic approach for solving the airline maintenance routing with aircraft on ground problem. Proceedings of 2nd IEEE International Conference on Logistics Operations Management, GOL 2014, 48–52. https://doi.org/10.1109/GOL.2014.6887446

Feo, T. A., & Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, 109–133.

Gargiulo, F., Pascar, D., & Venticinque, S. (2013). A Multi-agent and Dynamic Programming Algorithm for Aeronautical Maintenance Planning. International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, (6681263), 410–415. https://doi.org/10.1109 /3PGCIC.2013.165

Gopalan, R., & Talluri, K. T. (1998). The Aircraft Maintenance Routing Problem. Operations Research, 46(2), 260–271. https://doi.orgm /10.1287/opre.46.2.260

Han, Q., Cao., W., & Yang, L. (2012). Study on optimization of aeronautical maintenance process. Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012, 2, 203–206. https://doi.org/10.1109/ICCSEE.2012.398

Liang, Z., Chaovalitwongse, W. A., Huang, H. C., & Johnson, E. L. (2011). On a New Rotation Tour Network Model for Aircraft Maintenance Routing Problem. Transportation Science, 45(1), 109–120. https://doi.org/10.1287/trsc.1100.0338

Nickles, G., Him, H., Koenig, S., Gramopadhye, A., & Melloy, B. (1999). A Descriptive Model of Aircraft Inspection Activities, 1–6. https://doi.org/10.1109/ICCS.2012.6406136

Safaei, N., & Jardine, A. K. S. (2018). Aircraft routing with generalized maintenance constraints. Omega (United Kingdom), 80, 111–122. https://doi.org/10.1016/j.omega.2017.08.013

Santosa, B., & Ai, T. J. (2017). Pengantar Metaheuristik - Implementasi dengan Matlab. Surabaya: ITS Tekno Sains.

U.S. Department of Transportation, & Federal Aviation Administration. (2016). AC 120-16G - Air Carrier Maintenance Programs. Retrieved from https://www.faa.gov/documentLibrary /media/Advisory_Circular/AC_120-16G.pdf