This is the Pre-Published Version.

Makespan Minimization in Aircraft Landing Problem under Congested Traffic Situation using Modified Artificial Bee Colony Algorithm

K.K.H. NG, C.K.M. LEE
Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China
kkh.ng@connect.polyu.hk , ckm.lee@polyu.edu.hk

Abstract – **Due to the increase in air transportation demand, runway capacity is reaching a bottleneck at the international airports, especially during peak hours. Managing on aircraft sequencing or sequencing problems in airport perspective have become a crucial operation nowadays in order to maintain safety landing and utilize the runway facility to handle the schedules for all incoming aircraft under congestion. The traditional approach allows aircraft to remain an economic speed during approaching to the airport. However, such approach may not be applicable in congested air traffic situation. Therefore, the makespan minimization is more practical for the rescheduling efforts afterwards. This article presents a modified artificial bee colony algorithm to obtain nearly optimal solution to support the air tower controller in order to obtain last-minute decisions of landing sequence. The modified artificial bee colony algorithm for aircraft landing problem provides a promising optimal search within 6.1 seconds to handle last-minute disruption.**

Keywords – **Aircraft landing problem, Air traffic control, Meta-heuristics, Artificial bee colony algorithm**

I. INTRODUCTION

Aircraft Landing Problem (ALP) is one of the major attributes in Air Traffic Flow Management to maintain smooth air traffic in from airspace to the corresponding airport. The runway capacity is the key bottleneck for the aircraft landing system. The cost of delays may rise due to late departure, reassignment and customer dissatisfaction. The current approach in ALP follows the rule of First-Come-First-Serve (FCFS) to arrange the landing sequence. The landing time is not exactly same as the earliest arrival time of a flight, as the air tower control must follow the standard landing procedure in order to sustain a safe landing sequence and schedule. Since aircraft landing generates vortices that create adverse effect to the trailing aircraft, separation time between two aircraft must be considered in the landing schedule. The separation time is mainly determined by the flight classes matrix in terms of time [1]. In this regard, the total makespan of the runway may not necessarily be the optimal solution using the FCFS approach.

In normal traffic situation, the landing decision is based on the target landing time of incoming flights. Flights remain an economic speed in reaching to the airport. The objective in such situation is to minimize the total tardiness of all flights from the target landing time, which means that a penalty cost will exist with earlier or late arrival. In contrast, the total makespan is expected to be lengthy, as the airline aims to maintain a low cost in air transport [2-4]. Under congested traffic scenario, certain runway capacity is omitted if the model has taken tardiness as an objective function. In order to reduce the deterioration of following flight schedules, makespan minimization in airport aspect should be respected [1].

Meta-heuristics research in ALP model is much more preferable in handling last-minute changes under congested air traffic, since the computational time using exact method is lengthy that could not meet with the near time decision in air tower control. Typical examples include Genetic algorithm [5], Simulated Annealing [3] and Iterated Local Search [4]. This research is the first attempt to apply Artificial Bee Colony algorithm in ALP model, and the proposed algorithm yields a better optimal solution within several seconds.

TABLE I
Separation time between two consecutive flights for safe landing [1]

		Following aircraft		
		SSF	MSF	LSF
Leading aircraft	SSF	82	69	60
	MSF	131	69	60
	LSF	196	157	96

SSF = Small size flight
MSF = Medium size flight
LSF = Large size flight

II. PROBLEM FORMULATION

The ALP model under congested traffic scenario is to minimize the overall completion or makespan of a runway with hard constraints in separation requirement. Table II shows the notation and decision variables in this ALP model. The proposed algorithm is able to provide a promising solution to support the landing decision.

TABLE II
Notation and decision variable

Notations	Explanation
i	Aircraft ID i (i = 1,2, … , n)
n	The maximum number of aircraft
r	Runway ID r (r = 1,2, … m)
m	The maximum number of runway
S_{ij}	The separation time between aircraft i and j scheduled on the same runway, $S_{ij} \geq 0$
e_i	The possible earliest landing time of aircraft i

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

l_i	The possible latest landing time of aircraft i
T_{ir}	The assigned landing time for aircraft i on runway r
M	Large number associated with the artificial variable

Decision variables	Explanation
x_{ir}	1, if aircraft i is assigned to runway r; 0, otherwise
y_{jir}	1, if aircraft j is assigned to before aircraft i in a consecutive sequence on the same runway r 0, otherwise
T_{ir}	The assigned landing time for aircraft I on the runway r in schedule s, $T_{ir} \geq 0$
C_r	The total makespan of runway r in schedule s, $C_r \geq 0$

$$\min C_r \tag{1}$$

s. t.

$$x_{jr} + x_{ir} \leq 1 - y_{ijr} - y_{jir}, \forall i, j, i \neq j, r \tag{2}$$

$$y_{jir} + y_{ijr} \leq 1, \forall i, j, i \neq j, r \tag{3}$$

$$\sum_{r=1}^{m} x_{ir} = 1, \forall i \tag{4}$$

$$e_i \leq T_{ir}(\mathrm{X}) \leq l_i, \forall i \tag{5}$$

$$T_{ir}(\mathrm{X}) - T_{jr}(X) \geq S_{ji} \times Y_{jir} - M\left(1 - Y_{jir}\right), \forall i, j, r \tag{6}$$

$$C_r \geq T_{ir}(X) - M(1 - x_{irk}), \forall i, r \tag{7}$$

$$x_{irk} \in \{0,1\}, \forall i, r, k \tag{8}$$

$$y_{jir} \in \{0,1\}, \forall i, j, r \tag{9}$$

Objective function (1) is to minimize the competition of each runway C_r by modifying the landing sequence, named makespan minimization. In congested airport, the runway loading is reaching to the maximum tolerance of runway capacity. Makespan minimization is the approach to schedule the incoming aircraft according to the earliest landing time of each flight and separation time of two consecutive aircraft. The separation time herein follows the requirements from the literature [1]. Constraints (2) and (3) calculate the consecutive sequence in Boolean value. If aircraft i and j are assigned to runway r, y_{jir} equals to 1, otherwise, y_{jir} equals to 0. Constraint (4) ensures that each aircraft can only land on one runway. The landing time T_i of each flight must lie on the time window $[e_i, l_i]$ in constraint (5). The separation time between aircraft classes is to measure the minimum time requirement of flight i and j in terms of their flight classes. The landing time of the trailing aircraft must be larger or equal to the summation of previous landing time and separation requirement using inequality equation (6). The makespan of each runway must be larger than the completion landing time of all flights T_{ir} on runway r by constraint (7). The decision variable x_{ir} and y_{jir} could be either 0 or 1 by the constraints (8) and (9).

III. MODIFIED ARTIFICIAL BEE COLONY ALGORITHM

ABC algorithm is classified as a Swarm Intelligence (SI) algorithm in optimization, which is proposed by Karaboga [6]. The algorithm architecture includes three main phases, which are initialization, exploitation phase and exploration phase. Employed bees and onlooker bees carry the random neighbourhood searching to the previous solution so as to obtain better solution quality in the process of exploitation, while the scout bee will terminate the searching on a solution and replace by an initialized solution when a solution is tripped in local optimal. The termination criterion is measured by the number of unsuccessful update. The detail descriptions are shown as follows:

Various probabilistic selection processes in the ABC algorithm contribute to the ability in exploitation. First, arbitrary neighbourhood searching provides randomness during searching locally among each food source, which enhances the ability to exploit unknown space from a known solution by different types of operators. Second, the greedy exploitation in the neighbourhood search operation is applied in local selection achieved by employed bee and onlooker bees when the neighbourhood solution $\overline{c_i}$ is more desirable than the previous solution c_i. Third, onlooker bees will conduct a global probabilistic selection among the discovered regions to encourage exploitation of better solution by fitness proportionate selection. The fitness proportionate selection follows the rule in roulette wheel selection with a random number p. The method of ABC algorithm in exploration employs scout bee phase to terminate searching in worse candidate solutions, and has the ability to escape from local optimum traps. Local optima may exist and fail to converge to the global optimum when a neighbourhood search is trapped.

TABLE III
Notation of artificial bee colony algorithm

Notations	Explanation
CS	The size of bee colony
SN	The number of colony solution
MaxIter	The maximum number of iteration
D	The dimension of an independent solution
$c_i, i = 1,2, \ldots, SN$	The position of each solution in bee colony
$\mathrm{fun}(c_i)$	The objective value of solution c_i
$\mathrm{fit}(c_i)$	The fitness value of solution c_i
$IndiProb_i$	The probability of an individual solution c_i among the entire colony in term of fitness value
$CumuProb_i$	The cumulative probability of an individual solution c_i in ascending order among the entire colony in term of fitness value
$\overline{c_i}$	The neighbour solution of an individual solution c_i
$\mathrm{trial}(c_i)$	The accumulated trial value of an individual solution c_i, which cannot be enhanced the quality of solution in terms of its objective value
$limit$	The maximum tolerance of $\mathrm{trial}(c_i)$
p	Random number, $0 \leq p \leq 1$

Table III demonstrates the notation of ABC algorithm. The size of bee colony is equal to CS. In a bee colony, half of the bees are functioned as employed bees, and the remaining are onlooker bees. Therefore, the number of colony solutions SN equals half of the CS. The pseudo code of ABC algorithm is illustrated in Table IV.

TABLE IV
Pseudo code of artificial bee colony algorithm

Algorithm Architecture

Initialization

Generate the initial solution randomly for each individual solution $c_i, i = 1,2, \dots, SN$

Compute the objective value $\text{fun}(c_i)$ and fitness value $\text{fit}(c_i)$ of each solution c_i

$$fit(c_i) = \begin{cases} \dfrac{1}{1 + fun(c_i)}, if\ fun(c_i) \geq 0 \\ 1 + abs(fun(c_i)), if\ fun(x_i) \leq 0 \end{cases}, \forall i$$

set $\text{Loop} = 0$

set $\text{MaxIter} = \text{Number of Aircraft} \times 1000$

Exploitation Phase

do

Employed Bee Phase

Adopt neighbourhood operation for each solution $c_i, \forall i$ to generate a neighborhood solution $\overline{c_i}$

Calculate the objective value $\text{fun}(\overline{c_i})$ and fitness value $\text{fit}(\overline{c_i})$

IF the neighbourhood solution $\overline{c_i}$ is better than original solution c_i

THEN

$c_i \leftarrow \overline{c_i}$

ELSE

$\text{trial}(c_i) = \text{trial}(c_i) + 1$

Onlooker Bee Phase

Calculate the selective probability of individual solution $c_i, \forall i$

$$IndiProb_i = \frac{fit(c_i)}{\sum_{i=1}^{SN} fit(c_i)}, \forall i$$

Compute the cumulative probability of each solution $CumuProb_i$ in ascending order

Rand generate a number p in roulette wheel selection, and select one solution c_i. The selected c_i pass to employed bee phase again.

Exploration Phase

Scout Bee Phase

IF the trial of a solution c_i, $\text{trial}(c_i)$ over the maximum tolerance of neighbourhood searching $limit$

THEN

Generate a new solution to replace solution c_i

$\text{trial}(c_i) = 0$

Find the best solution from the bee colony, $c_i, i = 1,2, \dots, SN$

Record the current best solution as global solution

Stopping Criterion

Loop = Loop + 1

while loop < MaxIter

Return the global solution

The convergence from a random initial solution in ABC algorithm to a promising region may take several thousand iterations. The solution space may be too large to coverage to global optimal using random assignment. Certain promising solution may be abandoned as it overs the maximum tolerance in searching optimal. One of the easiest ways to solve this problem is to increase the tolerance of searching, but this will further deteriorate the ability of exploration, and limit the search coverage in a solution space. Therefore, a constructive heuristics is applied to provide a solution quality. The ABC algorithm can further improve the solution without spending extra effort to converge to a promising solution region. The aircraft sequence is sorted based on their earliest landing time. Each aircraft is randomly assigned to a runway, until all flights are assigned. The random number in runway assignment is to ensure the diversity between solutions so as to avoid convergence problem in population-based meta-heuristics. The pseudo code of constructive heuristic is shown in Table V.

TABLE V
Constructive heuristics for initialization in aircraft landing problem

Modified Initialization Phase

Store the earliest landing time in permutation array $permu$ and create an ascending flight number in sequential array $sequence$

$permu = (e_1, e_2, \dots, e_{n-1}, e_n)$
$sequence = (i_1, i_2, \dots, i_{n-1}, i_n)$

Sort the aircraft landing sequencing in ascending order based on the target landing time e_i in permutation array $permu$ and revise the corresponding sequential order in sequential array $sequence$

Create a runway assignment using random number $\text{r}, \text{r} = (1,2, \dots, \text{m})$, and take the first flight ID in $sequence$ and remove it afterwards.

Assign the flight ID to corresponding runway array

Until all the flight ID are assigned

The employed bee phase adopts several neighbourhood operators NO to generate a neighbourhood solution iteratively in discrete ABC algorithm. This operator randomly selects one or two elements to measure the feasible revision in solution quality by objective value.

NO_{Swap}: This neighbourhood operator randomly selects two flights and swap their sequence and position. The swap method can be applied on the same or different runways.

NO_{insert}: This neighbourhood operator randomly selects one flight and insert to another position. The insert method can be applied on the same or different runways.

$NO_{reverse}$: This neighbourhood operator selects a sequential range using random number, which is smaller than the dimension of the solution size. Reverse order to the selected region will be applied. Same position will be filled up by a reversed sequence.

IV. EXPERIMENTAL RESULT AND DISCUSSION

The proposed ABC algorithm is evaluated with a randomly generated 18 ALP instances under high traffic situation. The number of runway is ranged from 1 to 4. The landing time is randomly generated within the landing time window $[\text{LandingLB}, \text{LandingUB}]$. The instances with equal or less than 15 aircraft are considered as small size instances (ALP_1 to ALP_9), while the others are considered as large size instances (ALP_10 to ALP_18). The distribution of aircraft classes is either dominated by

large size, medium size or small size aircrafts. The proposed ABC algorithm was coded in *C# language* with *visual studio 2015* on a computer with *Intel Core i7 3.60 GHz CPU* and *16.0 GB ram* under *Window 7 Enterprise 64-bit* operating environment. The proposed algorithm is compared with *IBM CPLEX Optimizer* and *ABC algorithm* using the same operating system. The processing time of Mixed Integer Programming (MIP) is limited by 3600 seconds in *CPLEX Optimizer*. The description of ALP instances are shown in Table VI.

TABLE VI
The description of aircraft landing problem instances

No. flight	ID	Landing LB	Landing UB	LSF	MSF	SSF
5	ALP_1	60	600	3	1	1
5	ALP_2	60	600	1	3	1
5	ALP_3	60	600	1	1	3
10	ALP_4	60	1200	6	2	2
10	ALP_5	60	1200	2	6	2
10	ALP_6	60	1200	2	2	6
15	ALP_7	60	1800	9	3	3
15	ALP_8	60	1800	3	9	3
15	ALP_9	60	1800	3	3	9
20	ALP_10	60	2400	12	4	4
20	ALP_11	60	2400	4	12	4
20	ALP_12	60	2400	4	4	12
25	ALP_13	60	3000	15	5	5
25	ALP_14	60	3000	5	15	5
25	ALP_15	60	3000	5	5	15
30	ALP_16	60	3600	18	6	6
30	ALP_17	60	3600	6	18	6
30	ALP_18	60	3600	6	6	18

The parameter $limit$ is equal to $\mathrm{SN} * \mathrm{n} * \mathrm{m}$. The number of iterations $MaxIter$ in our preliminary study is equal to $1000\mathrm{n}$. Table VIII describes the experimental result using modified ABC algorithm, and compared the performances of MIP and ABC algorithm in terms of objective value on average. Each instance was run 10 times to summarize an average performance in objective value and CPU. For the purpose of comparison, the average gap of the objective values between solution from proposed algorithm and optimal solution are indicated in the Table VII to measure the ability of exploitation and exploration to global optimal. The experimental results for the solution obtained by modified ABC algorithm within no deviation from the true optimal are excluded in the context for the ease of presentation ($\% Gap_{opt} = 0.00$). Table VIII illustrates that the modified ABC algorithm is able to obtain a better solution in terms of objective value than MIP model with the processing time requirement of 3600 seconds. The modified ABC algorithm outperforms MIP in large size instance. The computational time for large size instance with MIP is normally over than 3600 seconds. Modified ABC algorithm is able to obtain a better solution quality within 7 seconds, except the ALP_18 instances with 2 runways.

The original ABC algorithm is not stable to converge to the optimal. The average deviation from optimal in percentage for ABC and modified ABC algorithms are 4.44% and -1.58% correspondingly. In this regard, the results indicate the importance and contribution of constructive heuristic for ABC algorithm. In addition, the average computational time in modified ABC algorithm is less than the original one (2.08sec < 2.32sec), as shown in Table VII. The result herein can be interpreted as the number of abandoned solution, which is significantly reduced and imply a lower level of unsuccessful update from neighbourhood solution by constructive heuristic.

TABLE VII
The average performance of experimental results

ABC		Modified ABC	
Avg. CPU	**Avg. % Gap (Opt.)**	**Avg. CPU**	**Avg. % Gap (Opt.)**
2.32	4.44%	**2.08**	**-1.58%**
Min % Gap	**Max % Gap**	**Min % Gap**	**Max % Gap**
-21.08%	23.35%	**-24.45%**	**4.18%**

TABLE VIII
The experimental result of aircraft landing problem instances

ID	No. Runway	Ins. Node	MIP w/ CPLEX			ABC			Modified ABC		
			Makespan	CPU (sec)	GAP (LB)	Makespan	CPU (sec)	Gap (Opt.)	Makespan	CPU (sec)	Gap (Opt.)
ALP_1	1	1	514	<1	0.00%	541.0	<1	5.25%	514.9	<1	0.18%
ALP_3	1	9	507	<1	0.00%	526.0	<1	3.75%	528.2	<1	4.18%
	2	10	420	<1	0.00%	423.0	<1	0.71%	421.8	<1	0.43%
ALP_5	1	17	1271	45.38	0.00%	1448.0	<1	13.93%	1314.2	<1	3.40%
ALP_6	1	21	1088	>3600	6.25%	1342.0	<1	23.35%	1116.2	<1	2.59%
ALP_7	1	25	1963	>3600	13.70%	2314.0	<1	17.88%	1982.1	<1	0.97%
	3	27	1699	>3600	0.29%	1783.0	1.17	4.94%	1699.3	1.19	0.02%
ALP_9	1	33	2055	>3600	14.89%	2311.0	<1	12.46%	**1936.7**	**<1**	**-5.76%**
ALP_10	1	37	2581	>3600	12.55%	3080.0	1.44	19.33%	**2470.8**	**1.24**	**-4.27%**
ALP_11	1	41	2641	>3600	11.85%	3095.0	1.54	17.19%	**2594.3**	**1.412**	**-1.77%**

	2	42	2388	>3600	2.51%	2581.0	2.11	8.08%	**2381**	**1.87**	**-0.29%**
	3	43	2388	>3600	2.51%	2418.0	2.76	1.26%	**2328**	**2.28**	**-2.51%**
ALP_12	1	45	2410	>3600	2.66%	2915.0	1.53	20.95%	**2348.6**	**1.27**	**-2.55%**
ALP_13	1	49	3229	>3600	7.31%	3829.0	2.22	18.58%	**2993**	**1.75**	**-7.31%**
	4	52	3341	>3600	10.42%	**2993.0**	**4.87**	**-10.42%**	**2993**	**4.29**	**-10.42%**
ALP_14	1	53	3133	>3600	10.61%	3764.5	2.21	20.16%	**2974.1**	**1.91**	**-5.07%**
	2	54	2861	>3600	2.10%	3094.5	3.00	8.16%	**2801**	**2.78**	**-2.10%**
	3	55	2903	>3600	3.51%	2906.2	3.96	0.11%	**2801**	**3.55**	**-3.51%**
	4	56	2937	>3600	4.63%	**2831.9**	**4.71**	**-3.58%**	**2801**	**4.26**	**-4.63%**
ALP_15	1	57	2868	>3600	3.24%	3534.9	2.03	23.25%	**2775**	**1.83**	**-3.24%**
ALP_16	1	61	4232	>3600	18.71%	4892.9	2.56	**15.62%**	**3680.9**	**2.62**	**-13.02%**
	2	62	3882	>3600	11.39%	**3881.8**	**4.21**	**-0.01%**	**3440**	**3.78**	**-11.39%**
	3	63	3536	>3600	2.71%	3607.4	5.71	2.02%	**3440**	**4.96**	**-2.71%**
ALP_17	1	65	3576	>3600	1.68%	4285.1	3.02	19.83%	**3516**	**2.68**	**-1.68%**
	3	67	3678	>3600	4.40%	**3516.0**	**5.84**	**-4.40%**	**3516**	**4.92**	**-4.40%**
	4	68	3516	200.75	0.00%	**3516.0**	**6.61**	**-4.40%**	**3516**	**5.86**	**-4.40%**
ALP_18	1	69	4714	>3600	26.28%	4958.7	2.78	5.19%	**3868.1**	**2.51**	**-17.94%**
	2	70	3416	>3600	19.49%	3942.3	4.04	15.41%	3534.1	3.56	3.46%
	3	71	4650	>3600	25.27%	**3669.7**	**5.00**	**-21.08%**	**3512.9**	**4.38**	**-24.45%**

V. CONCLUSION

In this article, the constructive heuristic for ABC algorithm is proposed to meet the near time decision making in aircraft landing problem under congested traffic situation. The research focus has been revised in airport aspect rather than in airline aspect under high level of traffic in surface operation. The re-schedule can be done within several seconds to reduce the impact of transient queueing congestion with limited runway capacity. The experimental results indicate the modified ABC algorithm outperforms the MIP and original ABC to achieve better results with limited computational requirement. The further work in aircraft landing problem with congested situation may refer to the development of robust landing schedule. The consideration of unknown variable under congested air traffic is a more realistic representation of landing operation. The main concern is shifted to construct a robust schedule that is able to examine all possible worst scenarios with an analysis of flight time delay.

ACKNOWLEDGMENT

The research is supported by The Hong Kong Polytechnic University. The authors would like to thank the research committee and the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University for support of this project (RU8H).

REFERENCES

[1] H. Balakrishnan and B. G. Chandran, "Algorithms for scheduling runway operations under constrained position shifting," *Operations Research*, vol. 58, pp. 1650-1665, 2010.

[2] H. Pinol and J. E. Beasley, "Scatter search and bionomic algorithms for the aircraft landing problem," *European Journal of Operational Research*, vol. 171, pp. 439-462, 2006.

[3] A. Salehipour, M. Modarres, and L. M. Naeni, "An efficient hybrid meta-heuristic for aircraft landing problem," *Computers & Operations Research*, vol. 40, pp. 207-213, 2013.

[4] N. R. Sabar and G. Kendall, "An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem," *Omega*, vol. 56, pp. 88-98, 2015.

[5] J. Beasley, J. Sonander, and P. Havelock, "Scheduling aircraft landings at London Heathrow using a population heuristic," *Journal of the operational Research Society*, pp. 483-493, 2001.

[6] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department2005.