Solving Aircraft Scheduling Problem using Artificial Bee Colony Optimisation and Receding Horizon Control

Abhijith T.R.
2020CSB1062
2020csb1062@iitrpr.ac.in

Pratham Kundan
2020CSB1114
2020csb1114@iitrpr.ac.in

***Abstract*—The Aircraft Scheduling Problem is an optimization problem from the field of Air Traffic Control involving the scheduling of incoming and outgoing aircraft onto allotted runways. The problem involves satisfying the scheduling requirements of various aircraft while considering separations and runway availability. It is a problem with inherent difficulty as it is an NP-Hard problem. This paper solves this problem for the multiple aircraft and runway case using an improvised Bee Colony Optimisation technique which is a local search algorithm that attempts to achieve a solution in real time for dynamic inputs. The solution first factors in the timing requirements of various aircraft and then attempts to find a solution that minimizes deviations from preferred arrival and departure times, by mimicking the behavior of foraging bees. The use of a metaheuristic based approach is capable of providing acceptable solutions while also enabling use in dynamic environments due to its fast running nature. The paper also compares the performance of the algorithm to other prevalent algorithms in this domain.**

Nomenclature

ABC	Artificial Bee Colony
ALP	Aircraft Landing Problem
ASP	Aircraft Scheduling Problem
ASS	Arrival Sequencing and Scheduling
ATP	Aircraft Takeoff Problem
BCO	Bee Colony Optimization
FCFS	First Come First Served
GA	Genetic Algorithm
RHC	Receding Horizon Control

I. Introduction

Runways are one of the biggest bottlenecks that any airport faces when trying to expand its traffic capacity. It is often infeasible to construct new runways because of the prohibitive costs involved. Currently busy airports such as the Indira Gandhi International Airport experience traffic of up to 80 movements per hour. In such situations, a practical approach to improving the capacity is to optimize the use of existing infrastructure. Various companies do offer software scheduling solutions, however the use of metaheuristic based approaches has not yet been considered.

The ASS is an important aspect of Air Traffic Control. It involves the allocation of runways and the assignment of arrival and departure times to incoming and outgoing aircraft, respectively. The problem is usually divided into two, the ALP and the ATP. However, the approach to scheduling both of these is similar and no special treatment is provided as to whether the aircraft is landing or taking off. The difference between these problems lies in the existence of additional constraints. In this paper we only consider the ASP which reduces both these problems to the simple assignment of aircraft to runways. The simplest solution to such a problem is the FCFS approach which involves assigning aircraft runways in the order of their arrival or departure times. However, such an approach does not provide the optimal solutions in the case of congestion at the airport. Air Traffic Controllers must also adhere to certain separation requirements between different classes of aircraft to maintain safety standards. This problem is known to be NP-hard [10]; hence finding an optimal ordering of aircraft is often infeasible. The problem is usually framed as an optimization problem with constraints on landing or takeoff times, fuel requirements and turnaround times.

Most currently available schedulers make use of FCFS based approaches after ordering based on various constraints. These approaches lend themselves well to the dynamic nature of the airport settings, allowing for rapid rescheduling owing to various unpredictable factors. Various approaches of the genetic algorithm to this problem have already been studied. The application of various metaheuristic based algorithms is currently being explored. Artificial be colony optimisation is one such algorithm. The ABC algorithm is a nature-inspired optimization algorithm that is based on the foraging behaviour of honeybees. It was proposed by Derviş Karaboğa in 2005. The algorithm mimics the way real bees forage for food sources and share information within the hive. ABC is a population-based algorithm and belongs to the category of swarm intelligence algorithms.

II. Related Work

ASP is a well-studied problem. Much of the literature on this topic has been reviewed in [1]. The authors begin with an overview of various exact solution approaches to solve the problem, such as mixed integer programming, stochastic learning processes, and dynamic programming. The paper also mentions some metaheuristic-based approaches, such as Ant Colony Optimization and Simulated Annealing, proposed in this domain. Metaheuristics helps tackle the dynamic nature of the ASP by providing fast-running algorithms that provide near-optimal solutions.

GA based approaches have been proposed in [2] to solve a static version of the ALP where all information about arriving aircraft is assumed to be known. The paper proposes three approaches using different chromosome representations and compares their performance. The paper also discusses a fourth approach using genetic programming. [3] proposes an approach using Ant Colony Optimization (ACO), which also uses Receding Horizon Control, a predictive control strategy. They essentially try to model the uncertainty associated with the arrival of aircraft and search for ways to solve this using ACO. The paper also compares the performance of this local search algorithm with genetic algorithms and the classical first come, first served (FCFS). This paper, however, considers only one runway. [4] extends the ACO approach to the multiple runway case. The authors considered a bi-level graph in which the first level of nodes represented the runways and the second level represented aircraft and applied ACO to assign weights to edges and obtain an optimized path. The authors discuss a Particle Swarm Optimization technique to solve the ALP in [5].

The Center TRACON Automation System (CTAS) [6] is a currently prevalent industry solution for air scheduling purposes. The Traffic Management Advisor (TMA) subsystem handles aircraft scheduling and sequencing. The system currently uses an FCFS approach [7] with possible changes in the Scheduled Time of Arrival (STA) to minimize the total delay that all aircraft must undergo. The system also considers dynamic scheduling and will change the schedule to accommodate new aircraft when they enter the airport's airspace.

III. Problem Formulation

A. Aircraft Scheduling Problem

The ASP involves scheduling a number of aircraft that are scheduled to arrive at a particular airport on the available runways. Assume there are N aircraft and M runways. Each aircraft has a preferred time of landing denoted by $p(i)$. Any deviation from this landing time leads to a cost. Arriving before the scheduled time leads to a cost of $pre(i)$ per second and delay leads to a cost of $post(i)$ per second. Suppose the scheduled time of landing of an aircraft is $s(i)$, then the cost for that plane is given by

$$c(i) = \begin{cases} pre(i) * (p(i) - s(i)), & \text{if } s(i) < p(i) \\ post(i) * (s(i) - p(i)), & \text{if } s(i) > p(i) \end{cases} \tag{1}$$

For any valid ordering $\prod$ and runway assignment r, the cost of the ordering is given by

$$c(\prod, s) = \sum_{i=1}^{n} c(i) \tag{2}$$

Any valid ordering must also respect the wake vortex separation between the aeroplanes i.e., if $wv(i,j)$ is the required separation between aircraft i and aircraft j where aircraft i lands before aircraft j on the same runway, then

$$s(j) \geq s(i) + wv(i,j), if \prod(i) > \prod(j) \text{ and } r(i) = r(j) \tag{3}$$

where r(i) and r(j) are the runways on which aircraft i and j are scheduled to land. The objective is to find a sequence π and a runway assignment r such that the cost function in (2) is minimized.

B. Receding Horizon Control

The Receding Horizon Control is an optimization technique used to improve the quality of attainable solutions through meta-heuristic approaches. It is a computational simplification of the problem, breaking it down into simpler problems with smaller domains. RHC reduces the search domain available for randomized search heuristics and increases the likelihood of finding acceptable solutions. RHC can also be used to model the dynamicity in situations, such as unavailable information in time-bound problems. The width of the receding horizon (w_d) denotes the size of the problem that is to be solved by the optimization algorithm. The horizon is broken down into smaller windows (T_I), and only the parts of the solution that lie within the first window are implemented. The horizon then adds a window to its end and discards the first window.

The advantages of this approach are twofold. The first advantage lies in breaking the entire problem down into receding horizons, which reduces the time taken for each iteration of the optimization algorithm and the search domain of the algorithm. The second advantage lies in the observation that each iteration of the algorithm reduces the work that needs to be done by successive iterations because the optimization objects in all windows but the last have been subject to ordering. Solution reuse by successive horizons is better than the random start commonly adopted by metaheuristic-based algorithms.

C. Artificial Bee Colony Algorithm

Algorithm 1 Bee Colony Optimization

```
0: procedure BCO(number_of_bees, max_iter, trial_limit, max_scouts)
0:     Initialize employed and unemployed bees
0:     Initialize best solution
0:     for iteration ← 1 to max_iter do
0:         EmployedBeePhase
0:         probabilities ← Probabilities based on fitness
0:         OnlookerBeePhase(probabilities)
0:         ScoutPhase
0:         Update best solution
0:     end for
0:     return Best solution
0: end procedure=0
```

The ABC Optimisation Algorithm was proposed in [8] by Dervis Karaboga. It is a nature-inspired swarm optimization algorithm that is based on the foraging behaviour of honey-bees. The algorithm mimics the way real bees forage for food

Algorithm 2 Bee Colony Optimisation Phases

```
0: procedure EMPLOYEDBEEPHASE
0:   for each employed bee do
0:     Generate next solution
0:     if next solution is better then
0:       Update solution and reset trials
0:     else
0:       Increment trials
0:     end if
0:   end for
0: end procedure=0

0: procedure ONLOOKERBEEPHASE(probabilities)
0:   Update unemployed bees based on probabilities
0:   for each onlooker bee do
0:     Update solution and type
0:   end for
0: end procedure=0

0: procedure SCOUTPHASE
0:   Sort employed bees by trials
0:   for each scout candidate do
0:     if trials exceed trail limits then
0:       Generate new solution and reset trials
0:     end if
0:   end for
0: end procedure=0
```

sources and share information within the hive. It defines 3 types of bees:

- **Employed Bees:** Employed bees explore the solution space and find food sources (solutions).
- **Onlooker Bees:** Onlooker bees observe the employed bees and select a food source based on their quality. The probability of selecting a particular food source is proportional to its fitness.
- **Scout Bees:** Scout bees are responsible for discovering new food sources. If an employed bee exhausts its limit without finding a better solution, it becomes a scout bee and explores a new solution randomly.

The algorithm is divided into three phases. To start the process, we create a population of bees. Half of the bees are labeled as employed and assigned a random solution and the rest are marked as onlookers.

1) **Employed Bee Phase:** Each employed bee uses another population member to generate a new solution. This new solution is then selected if it performs better compared to the current solution. If the solution remains the same, the trial counter at the current position is incremented.
2) **Onlooker Bee Phase:** Each of the onlooker bees selects a solution obtained by the employed bees based on probabilities proportional to the utility of each solution. The onlooker bees then become the employed bees, and the employed bees take the role of onlookers.
3) **Scout Phase:** In case the trial counter of a given solution exceeds a given limit, the employed bee at that solution is turned into a scout and is made to move randomly, irrespective of the quality of the new solution. This can potentially find better solutions.

These steps are repeated a finite number of times or till the desired solution quality is attained. The algorithm has been shown in (Algorithm 1).

IV. Proposed Solution

A. Constructing the Solution

The solution is denoted by an ordered set $((A_1, R_1), (A_2, R_2), ...(A_n, R_n))$ where A_i represents the aircraft and R_i is the assigned aircraft. The aircraft land in the order they are present in the list. For example, the solution $((A_1, R_1), (A_2, R_2), (A_3, R_2), (A_4, R_1))$ means that Aircraft A_1 lands on Runway R_1 followed by Aircraft A_4 and Aircraft A_2 lands on runway R_2 followed by Aircraft A_3 .Each aircraft lands at the time that has the least possible delay from their requested time of arrival/departure while following the wake vortex separation rules. The fitness of the solution is then calculated as the sum of $delay \times cost$ for each aircraft.

B. Initial State Configuration

To generate the initial solution population of the bee colony optimiser, we adopt a randomised approach. For each solution instance, we use the heuristic of ordering the aircraft by their landing times. We then assign a random runway to each aircraft which may then be changed while running the Bee Colony Optimiser. For this work, we have only considered the heuristic of ordering aircraft by their landing time due to its similarity to the earliest deadline first heuristic but the implementation of the ordering heuristic has been implemented such that it might be changed.

C. New Solution Construction

The BCO algorithm generates a new solution from an existing solution by randomly selecting a population member and combining the information from both to generate a new solution. For the Aircraft Scheduling Problem, We generate a new solution S' from a given pair S and C by selecting a random aircraft that is moved to a different runway. This can be represented as follows:

$$S'_i = \begin{cases} S_i + \phi \times (S_i - C_i) & \text{if } k = r \\ S_i & \text{otherwise} \end{cases} \tag{4}$$

Where

S' = New Solution

ϕ = A random number between -1 and 1

r = A random number between 0 and $num\_runways$

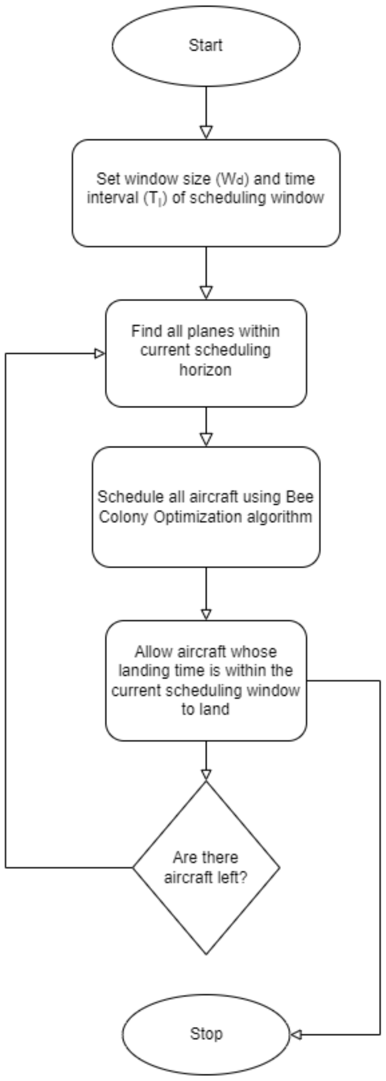
Start
Set window size (W_d) and time interval (T_I) of scheduling window
Find all planes within current scheduling horizon
Schedule all aircraft using Bee Colony Optimization algorithm
Allow aircraft whose landing time is within the current scheduling window to land
Are there aircraft left?
Stop

Fig. 1. Outline of Proposed Solution

D. BCO and RHC_BCO for Aircraft Scheduling Problem

To solve the ASP using Bee Colony Optimisation, we initialise a population of employed bees by using the methodology mentioned above. The bees then explore the solutions by selecting a random companion and moving to a new position in the solution space if the new solution is better than their current solution. Each of the explorers then selects an employed bee by weighing its fitness in the following manner.

$$weight_S = \frac{1}{1 + fitness(S)} \tag{5}$$

Scouts try to move the solutions out of any local maxima that the bees might be stuck in by selecting a random solution when the number of trials has been exceeded.

In RHC_BCO, we select the number of aircraft whose desired landing/ takeoff times lie in the current horizon (time window of size $T_I \times w_d$ starting from the earliest landing time of the first aircraft). This smaller set of aircraft is then taken as a subproblem and is solved using BCO. Of all the aircraft on the horizon, only the ones that are assigned a landing time in the first window are scheduled for landing. For any aircraft that is assigned a landing time that lies outside the first scheduling window, its earliest landing time is set to the starting time of the next window. This process is repeated after shifting the horizon forward by the length of one time window.

V. Experiments and Simulation

A. Dataset

The IKLI dataset was used for simulations on all the algorithms. The dataset is drawn from real traffic from the Paris-Orly airport. It makes available four different datasets with varying numbers of aircraft over varying time intervals. It models the cost as a piece-wise linear function; however, for our purposes, we have only considered the cost in the first columns. The dataset also considers the categorisation of planes into heavy, medium and light and provides the scheduled time of arrival, which acts as the ideal landing time for the problem in our simulations. The information has been drawn from the OpenSky network.

B. Simulation Framework

We constructed a problem-agnostic library that can be used to represent and find a solution to any local search problem. The Bee Colony Optimiser has then been implemented using this library. All code has been written in Python-3.11[1] and has been executed on a Windows 10 Machine with an Intel core i5 processor with 8 GB of RAM. We implemented functions to read data from the above-mentioned dataset.

VI. Results and evaluation

A. Effect of Number of Bees on Performance

The objective here is to view the reducing impact of increasing the number of bees on the performance of the system. We considered an instance of the problem with 50 aircraft and fixed the number of iterations to 100, the trial limit to 10 and the maximum number of scouts to 1. On increasing the number of bees, the cost of the solution obtained decreases. This is because the number of random solutions initially considered increases and the search space is searched more efficiently. As information from more regions of the search space is available to each bee when they share information, a better decision can be made as to which location of the search space contains the best solutions. However, there is a tradeoff between the time and the fitness required. On increasing the number of bees the time required to perform the optimization also increases. This can be attributed to the fact that more computational resources are required to perform the simulation as the number of bees increases.

We can see a linear increase in the time taken as the number of bees increases. However, the cost of the solution does not show a linear decrease on increasing the number of bees. The cost of the solution shows a decreasing trend but the slope of

[1]All code can be found at https://github.com/Abhijith-TR/Aircraft-Scheduling

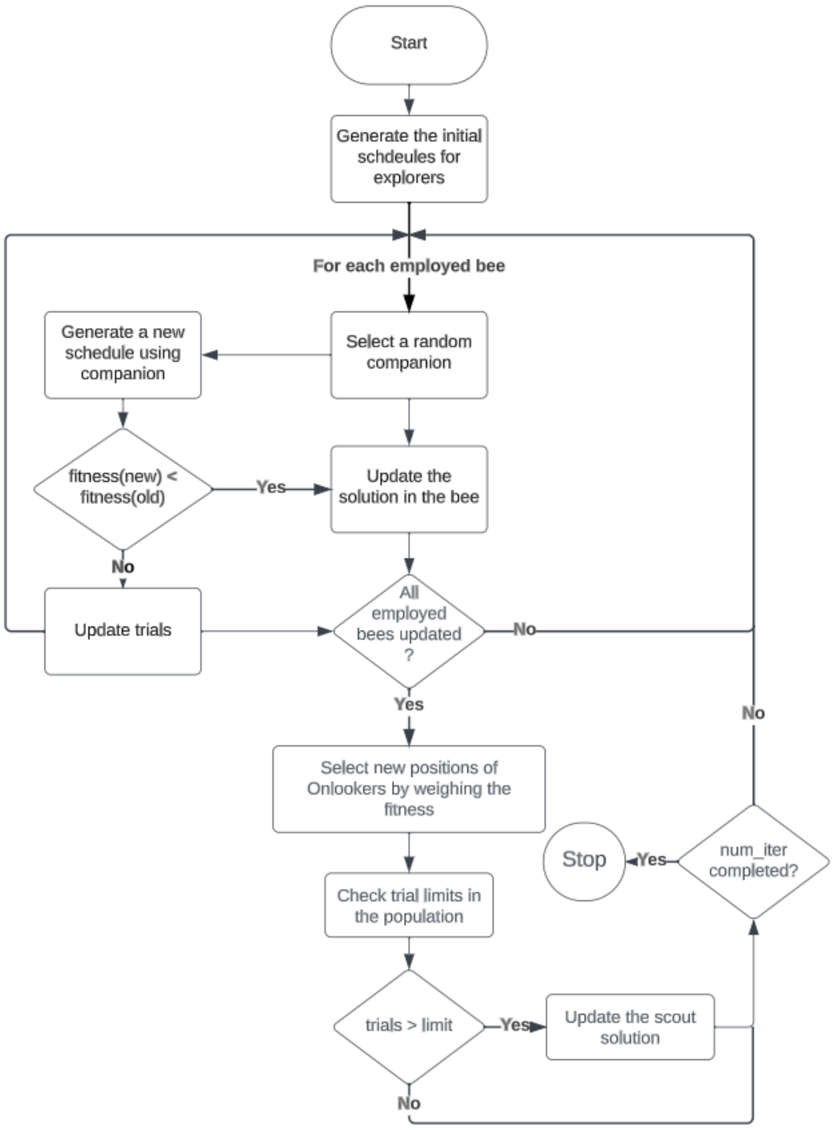
Start
Generate the initial schdeules for explorers
For each employed bee
Select a random companion
Generate a new schedule using companion
fitness(new) < fitness(old)
Yes
Update the solution in the bee
No
Update trials
All employed bees updated ?
No
Yes
No
Select new positions of Onlookers by weighing the fitness
Stop
Yes
num_iter completed?
Check trial limits in the population
trials > limit
Yes
Update the scout solution
No

Fig. 2. BCO for Aircraft Scheduling Problem

the curve decreases as we increase the number of bees. Hence a judgement has to be made with regards to the given problem size as to the necessary number of bees.

B. Effect of Maximum Iterations on Performance

The objective here is to view the reduced impact of increasing the number of iterations on the performance of the system. We considered an instance of the problem with 50 aircraft and fixed the number of bees to 500, the trial limit to 10 and the maximum number of scouts to 1. On increasing the number of maximum iterations, the fitness of the solution increases. This can be attributed to the fact that more time is available for the bees to perform their search and information sharing. This allows them to identify the best regions of the search space to find better solutions. On increasing the number of iterations, the time taken to run the algorithm increases.

We see a linear increase in the time taken as the number of iterations increases. However, the cost of the solution does not decrease linearly on increasing the number of iterations. The cost of the solution decreases rapidly and then shows a negligible change on increasing the number of iterations beyond a certain point. Therefore, there is a tradeoff between the time and the fitness of the solution when considering the number of iterations to be taken in a particular instance of the ASP.

C. Effect of RHC on Performance

We consider the effect of RHC on the performance of the BCO algorithm. On applying BCO algorithm with and without RHC on a problem size of 50 aeroplanes and varying the number of bees, we observe that the BCO algorithm with RHC consistently finds better solutions than the algorithm applied

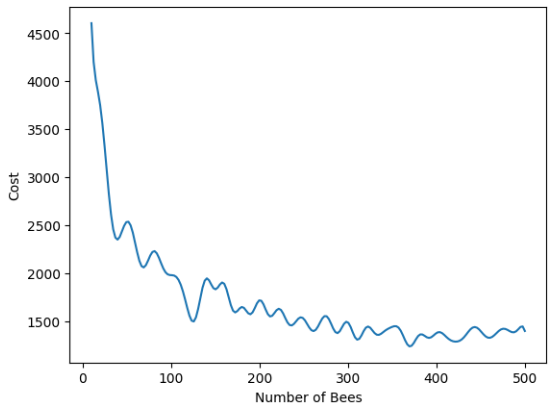

Fig. 3. Influence of number of bees on solution cost

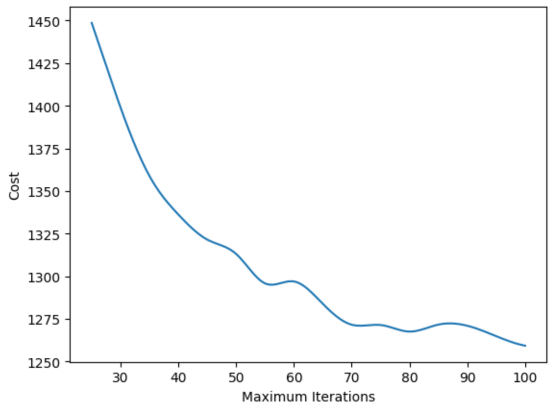

Fig. 4. Influence of number of iterations on solution cost

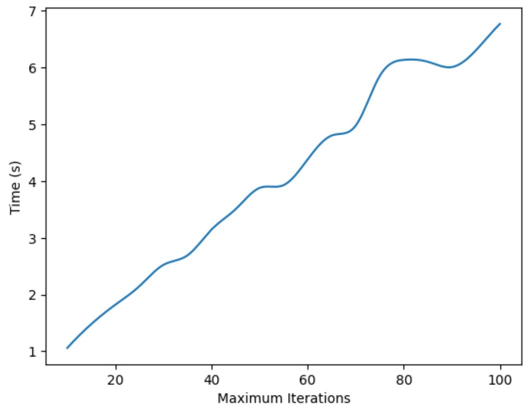

Fig. 5. Influence of number of iterations on time taken

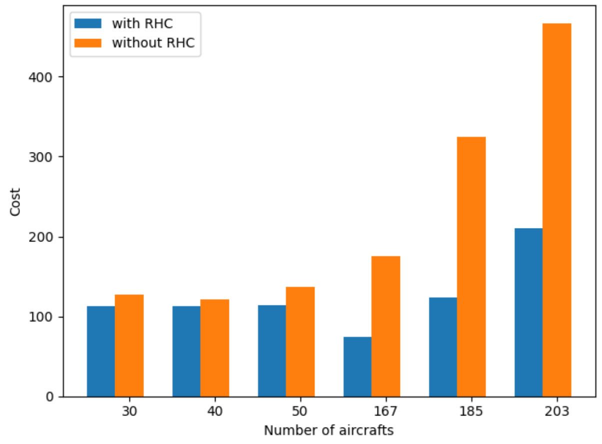

Fig. 6. Influence of number of bees on BCO with and without RHC

without RHC. This could be because RHC breaks the entire problem into smaller subproblems with smaller domains where the solution is easier to find. The RHC algorithm also allows for information sharing between runs of the algorithm because the initial assignment available to successive iterations of BCO contains some information embedded by the previous run of the algorithm. The algorithm without RHC has to work in a larger domain and will have a lesser probability of finding a similar solution inspite of having the same number of bees. BCO without RHC would require a larger number of iterations to find a similar solution to the one found by BCO with RHC.

We also considered varying the number of aircraft to find the performance differences of BCO with and without RHC. On increasing the number of aircraft, the gap between the solution found by BCO with RHC and BCO without RHC increases. This can be attributed to the fact that the size of the search space increases on increasing the number of aircraft. As BCO with RHC breaks the subproblem into smaller subproblems, the impact of the increase in the size of the search space is not as pronounced as in the case of BCO without RHC. Thus, the advantages of using an RHC-based approach to scheduling become more apparent as the number of aircraft increases.

D. *Effect of Varying Runways on Performance*

We consider the effect of varying the number of runways on the cost of the best solution found by the BCO algorithm. On increasing the number of runways the cost of the optimal solution found decreases. This is to be expected as the arriving aircraft can now be scheduled over a larger number of runways and thus, there are fewer possibilities of having to delay aircraft due to wave vortex separation requirements. Another notable observation is that BCO was able to find a better solution even on increasing the number of runways in the same number of iterations as before. This is indicative of the fact that BCO is capable of finding acceptable solutions even by increasing the size of the problem in similar amounts of time. Another observation was that BCO tends to use the available runways more efficiently than other approaches, such as FCFS. The almost uniform utilization of runways is another appreciable improvement of the BCO algorithm over traditional algorithms like FCFS.

E. *Comparison with Other Algorithms*

We compared the cost of the solution found by BCO with RHC with the cost of the solution found by other

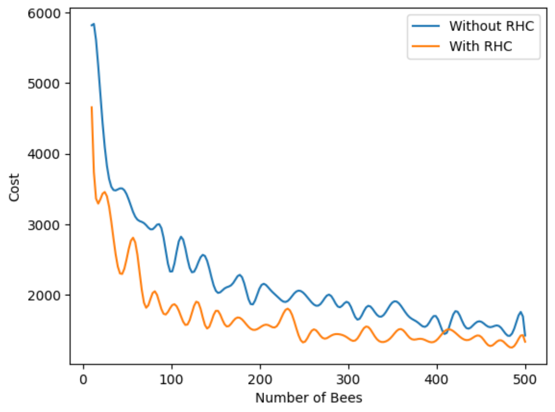

Fig. 7. Influence of number of aircraft on solution cost on BCO with and without RHC

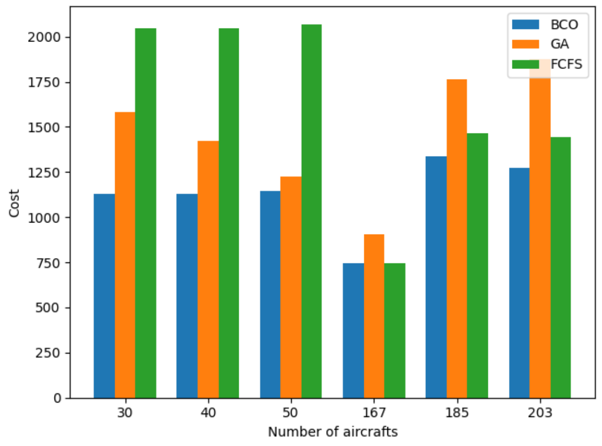

Fig. 8. Costs of best solutions found by GA, FCFS and BCO on various problem sizes

algorithms such as GA and FCFS. We considered 1000 bees, 100 maximum iterations and a trial limit 10 in the case of BCO. We considered a population size of 100 with a limit of 500 generations in the case of the Genetic algorithm. The FCFS algorithm tried to assign each flight as close as possible to its requested time of landing.

We found that on small problem sizes, BCO gives significantly better solutions than both FCFS and GA. This can be attributed to the fact that the small size of the search space allows BCO to find good solutions easily. The same can be said of GA, which also seems to be finding better solutions than FCFS. However, BCO does find better solutions than FCFS due to the trial limit imposed by the algorithm. This allows the algorithm to abandon local minima after a fixed number of iterations and pick up another random location to search. This gives the algorithm a better chance of finding global optima.

On larger problem sizes, BCO performs better than FCFS and GA. However, FCFS now gives better solutions than GA. This can be attributed to the fact that in a larger search space, GA has a higher chance of getting stuck at a local maxima. However, BCO will abandon local maxima after the trial limit.

Algorithm	Fitness
GA-Heuristic 1	11.25 (5000 iterations)
GA-Heuristic 2	11.25 (600 iterations)
GA-Heuristic 3	11.25 (25 iterations)
BCO_RHC	5.0 (25 iterations, 100 bees)

TABLE I
Comparing Results with Past Work in [2]

The advantage over FCFS is not as pronounced owing to the larger size of the search space.

We also compared the performance of our solution to the metaheuristic search method proposed in [2]. This solution uses Genetic Algorithms and implements three heuristics to generate solutions. It uses a slightly different method to calculate the fitness, which is by summing up the squares of the deviation in landing time. BCO was able to provide a much better solution with a lower total deviation from the scheduled landing times in the same number of iterations. We have compared our solutions in Table I. It is to be noted, however, that the aforementioned work only contains a limited dataset of 12 aircraft landing on 3 runways.

VII. Conclusion and Future Work

This paper has modulated the aircraft sequencing and scheduling problem in the form of an optimisation problem and has proposed a new solution approach in the form of BCO. The paper suggested specific implementations of various stages of the BCO algorithm such as initial states, new solution construction and selection. The results obtained are encouraging and imply that BCO is a viable algorithm in this particular scenario. The paper has studied experimentally the performance of bee colony hyperparameters on the best solution found. For varying problem sizes, the simulation results seem to suggest the existence of a tradoff between number of bees or number of iterations and the time taken. This seems to indicate diminishing returns on approach to optimality. The use of RHC to break the problem down into subproblems further enhances the performance of the algorithm. On application of RHC, experimental results suggest that the algorithm outperforms both genetic algorithms and FCFS.

Future work in this field could be the following a) study of multi-point crossover methods during the solution construction phase, b) comparing various ordering heuristics for ordering of aircraft, c) comparing the algorithm to other swarm-based algorithms such as Particle Swarm Optimisation and d) studying heuristic based methods to optimise the construction of landing schedule from a given solution sequence.

Acknowledgement

We would like to thank Dr Shashi Shekhar Jha for his valuable inputs and guidance throughout the project. We would also like to thank our TAs for guiding us in the right direction and helping us formulate our ideas clearly.

REFERENCES

[1] Ikli, Sana & Mancel, Catherine & Mongeau, Marcel & Olive, Xavier & Rachelson, Emmanuel. (2021). The aircraft runway scheduling problem: A survey. Computers & Operations Research. 132. 105336. 10.1016/j.cor.2021.105336.

[2] [2] V. H. L. Cheng, L. S. Crawford and P. K. Menon, ”Air traffic control using genetic search techniques,” Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328), Kohala Coast, HI, USA, 1999, pp. 249-254 vol. 1, doi: 10.1109/CCA.1999.806209.

[3] Z. H. Zhan et al., ”An Efficient Ant Colony System Based on Receding Horizon Control for the Aircraft Arrival Sequencing and Scheduling Problem,” in IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 2, pp. 399-412, June 2010, doi: 10.1109/TITS.2010.2044793.

[4] Bencheikh, Ghizlane & Boukachour, Jaouad & Alaoui, Ahmed. (2011). Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem. International Journal of Computer Theory and Engineering. 3. 224 - 233. 10.7763/IJCTE.2011.V3.309.

[5] Girish B.S., An efficient hybrid particle swarm optimization algorithm in a rolling horizon framework for the aircraft landing problem, Applied Soft Computing, Volume 44, 2016, Pages 200-221, ISSN 1568-4946, doi: 10.1016/j.asoc.2016.04.011.

[6] Erzberger, H. (1992). CTAS: Computer Intelligence for Air Traffic Control in the Terminal Area. NASA Technical Memorandum 103959. National Aeronautics and Space Administration, Ames Research Center.

[7] Neuman, F.; and Erzberger, H.: Analysis of Sequencing and Scheduling Methods for Arrival Traffic. NASA TM-102795, April 1990, Ames Research Center.

[8] Karaboga, D, (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[9] Ikli, S., 2020. Library of codes, instances and data sets for the aircraft landing problem (ALP). http://data.recherche.enac.fr/ikli-alp/ (On line; accessed January 23, 2021).

[10] R. Prakash, R. Piplani, J. Desai, An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput, Transportation Research Part C: Emerging Technologies 95 (2018) 570–581