Algorithmic Efficiency Comparison of Centralized and Distributed Arrival Management (AMAN) Problem in Terminal Airspace

Aniket Deshmukh ,Ying HUO, Daniel DELAHAYE, Philippe Notry, Mohammed SBIHI
OPTIM-Team/ENAC-LAB
ENAC – Université de Toulouse
Toulouse France
firstname.lastname@recherche.enac.fr

***Abstract*—Terminal Maneuvering Area (TMA) is one of the most complex and busiest airspaces where the air traffic is managed with the help of decision support tools such as Arrival Manager (AMAN) in many airports. The objective of AMAN involves regulating the inbound air traffic flow and providing the expected scheduled time for each flight considering the impact of the environment. As we know, the aircraft scheduling problem is NP-hard and requires the implementation of advanced algorithms. The heuristic algorithms are more likely to provide a quick and satisfying result, in which simulated annealing (SA) has been proven to be efficient and easily adapted to a complex problem that involves large-dimensional state-space. Previously, in our work [1], the problem is considered from a centralized point of view in which the integrated information of all flights is investigated. The optimization process involves population-based and computational consuming evaluation for each simulation. Considering the isolated decisions of each flight, it is evident that the system is naturally distributed. By focusing on the individual performance of each flight, the optimization process can be guided with prioritization and no integration of flight information is required, therefore the efficiency and flexibility of the algorithm will be increased. Since the centralized and distributed AMAN develop different implementations for simulated annealing, the properties of both adapted algorithms are demonstrated. The performances are analyzed in terms of the execution time and quality of result based on a case study on Paris-Charles de Gaulle airport.**

***Keywords*—Air Traffic Optimization, Centralized AMAN, Distributed AMAN, Terminal Maneuvering Area**

I. Introduction

Nowadays, many airports rely on AMAN to balance demand and available resources. The main aims of AMAN are to regulate the flow of aircraft entering the surrounding airspace of an airport and provide predictability for its users. To meet these objectives, AMAN provides the expected schedule time or trajectory prediction for each flight at different fixes to ensure the spacing and make full use of the capacity. However, as the flight operations are affected by the environment, the arisen of some unpredictable factors such as runway closure, severe weather changes need a quick response, therefore the improvement of computational time of the AMAN can benefit the whole system.

Normally, information of all the flights is aggregated to trace the status of one flight with respect to the others so as to ensure system safety. In this context, the flights are centralized in the network, which introduces some unnecessary underlying communications and extra computation time in the simulation [2]. One possible approach to overcome this drawback is to change the strategy of centralized evaluation to a decentralized (distributed) one. In a distributed perspective, aircraft self organize in order to reach the safety in an efficient way. Besides, they can also benefit from real time information coming from other aircraft (current positions, TAS, wind, etc...) to improve their trajectory predictions. The centralized AMAN and distributed AMAN will lead to different implementations on the chosen solution algorithm, simulated annealing (SA). In the application of SA, the distributed version focuses on the individual flight that is essential for the network safety which can potentially increase the computational efficiency.

The organization of this paper is summarized as follows. Section I introduces the addressed problem with background. Section II presents some works that are related to our problem. Then, in Section III, the mathematical model is introduced, while due to the structure differences, the objective functions of centralized AMAN and distributed AMAN are formulated respectively. Simulated annealing is then adapted to both models with the main differences presented in Section IV. The results are analyzed and compared in terms of execution time and quality in Section V. Conclusion and perspectives are discussed in Section VI.

II. State of the art

In previous works, the problems of TMA arrival management were studied extensively. By viewing the reviews in multiple works [3–5], we roughly summarized the works into two categories: The establishment of detailed models for solving specific problems considering the local network or relevant characteristics, and the development of novel algorithmic approaches to facilitate the computational efficiency or accuracy for solving the problem.

Lots of related studies of air scheduling problems were modeled as a job shop scheduling problem in which the resources and flights were considered as machines and jobs [3, 4, 6]. This provided a flexible environment that can be adapted to the problem with the required conditions or limitations. As different airports have their own features, different objective functions with weights or penalties are applied in the models. Some works established the objective functions and constraints by regarding the conflict elimination as the prior issue while taking delay and variable deviations into consideration for the interest of operation efficiency [7, 8]. The aircraft scheduling problem is known to be strongly NP-hard, and both exact and heuristic solution approaches were applied. The exact approach such as mixed integer programming was proposed [9, 10]. [11] summarizes the meta-heuristic algorithms for solving problems of airside operation research. Other techniques such as time decomposition, sliding windows are usually incorporated in the algorithms to improve computational efficiency [12, 13].

After reviewing both applicable preferred models or algorithmic focused works, it is clear that we need to incorporate details and realistic features in the model while easing a certain level of computation burden. The works of [14] and [15] has proposed to use the distributed simulated annealing according to the implemented problem. A classical job shop scheduling problem is tackled, and the final results stated the scalability and flexibility of the use of distributed simulated annealing. [16] compared the performance of different levels of centralization based on a typical distributed optimization problem, which shows the constraints endowed by the centralization will cause extra computational burden and derive less flexibility.

Inspired by the aforementioned work, and considering the distributed nature of our problem, the proposed model aims to provide a new perspective to convert the centralized AMAN to a distributed one by implementing different objectives in the optimization process. Two optimization models with different levels of centralization in terms of flight information aggregation in the simulation are established.

III. Mathematical modeling

In our research, a problem of flight scheduling taking the conflict detection and resolution into account in TMA is presented.

A. Network

In order to facilitate the simulation and reduce the computational burden, the arrival route in TMA is abstracted as a graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents the set of nodes and $\mathcal{L}$ represents the set of links. Nodes are waypoints on the arrival route including the runway thresholds and TMA entry node. Links are the arcs connecting two nodes according to the route structure. Each flight follows a designated route denoted as $\mathcal{U}_f = \{\mathcal{U}_f^l, \mathcal{U}_g^n\}$, where $\mathcal{U}_f^l$ and $\mathcal{U}_g^n$ represent the links and nodes that flight f passes through. Given a set of arrival flights $\mathcal{F} = \{F_1, ..., F_N\}$, the information of each flight ($f \in \mathcal{F}$) are specified:

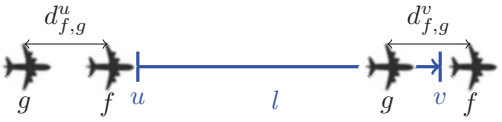

Figure 1. Link conflict detection

- C_f: Wake turbulence category of flight f.
- E_f: TMA Entry node of flight f.
- T_f^o: Initial RTA (Required Time of Arrival) when flight f entering the TMA through corresponding entry node.
- V_f^o: Initial speed of flight f when entering the TMA through entry node.

B. Decision variables and constraints

In this problem, two types of decision variables are proposed, and the relevant constraints are applied to the decision variables.

1) Entry time: flights can adjust the arrival time to TMA by changing their en-route speed, using an alternative route or considering a ground delay control in the departure airport [17]. A discrete time slot denoted as ΔT is used as one unit to measure the total entry time range. The maximum tardiness and earliness are represented as $\Delta T_{\max}$ and $\Delta T_{\min}$ which are composed of multiple of ΔT. Therefore, for a given flight f, a time slot decision variable $t_f \in \mathcal{T}_f$ has a flexible range of: $t_f = \{T_f^o + j\Delta T | \Delta T_{\min}/\Delta T \leqslant j \leqslant \Delta T_{\max}/\Delta T, j \in \mathbb{Z}\}$, where j denotes the number of time slots deviated from the initial arrival time.

2) Entry speed: for an arrival flight f, an entering speed decision variable $v_f \in \mathcal{V}_f$ has a constraint of: $v_f = \{V_f^{\min} + j\Delta_f^v | j \in \mathbb{Z}, |j| \leqslant (V_f^{\max} - V_f^{\min})/\Delta_f^v\}$, where j performs the deviation from the original speed, Δ_f^v is the discretized speed slot.

To summarize, the decision variable vector associated with the flight set $\mathcal{F}$ is denoted by $\mathbf{x}$, and we have:

$$\mathbf{x} = (\mathbf{t}, \mathbf{v})$$

where $\mathbf{t}$ and $\mathbf{v}$ denote the TMA entry time vector and entry speed vector respectively. The decision variables selected for flight f is denoted by $\mathbf{x}_f$.

C. Conflict evaluation

The conflict detection can be referred to the model established in [2], in which the conflicts are evaluated on specific locations in the network to determine whether there are separation violations between the consecutively operated aircraft or not. The conflict detection are categorized into two types due to the spatial separation requirements. Link conflict detection dedicated to the longitudinal separation violation and node conflict detection is introduced for the lateral separation violation.

1) Link conflict detection: as shown in Fig. 1, for each link $l = (\mu, \nu)$, where μ and ν represent the link entry and exit point. The conflict detection is carried out on both sides of the

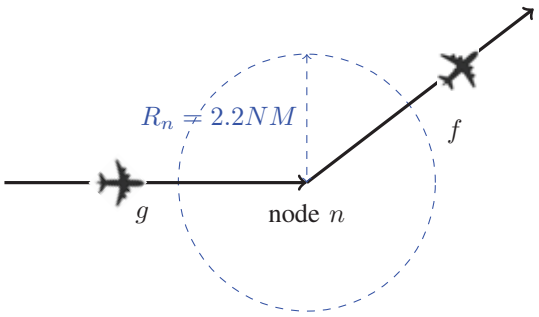

Figure 2. Node conflict detection

link. In order to evaluate the separation violation severity, the percentage of violated distance over the required separation is considered in our objective. The violation severity can provide extra information for the interest of the optimization process. Suppose that two aircraft f and g passing through link l, where aircraft f is ahead of g, the conflict will be detected and the corresponding costs for this conflict are given as follows:

$$C^l_{fg}(\mathbf{x}) = \begin{cases} 1 + \frac{D_{f,g} - d^{\mu}_{f,g}}{D_{f,g}}, & \text{if } d^{\mu}_{f,g} < D_{f,g} \\ 1 + \frac{D_{f,g} - d^{\nu}_{f,g}}{D_{f,g}}, & \text{if } d^{\nu}_{f,g} < D_{f,g} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $d^{\mu}_{f,g}$ and $d^{\nu}_{f,g}$ are the separation between aircraft f and g when aircraft f arrives at the entry and exit points of the current link. $D_{f,g}$ is the required separation associated with the categories of aircraft f and g that is indicated in Tab. I.

2) Node conflict detection: due to the fact that the intersection of two links might build an angle and the horizontal separation requirements of flights have risks to be violated, the lateral separation should be taken care of.

The nodes are considered as a disk with a radius of 2.2NM to ensure the minimum separation between two consecutive flights [2]. This area is defined as the *detection zone*, if two aircraft are observed in the detection zone simultaneously, a conflict is detected. The violation severity on nodes is calculated by taking into account the violation time and the flying time in the detection zone of both aircraft.

Fig. 2 shows the node structure where two consecutive aircraft f and g pass through the node n. The conflict cost

TABLE I
MINIMUM WAKE TURBULENCE SEPARATION REQUIREMENTS, IN NM.

Categories		Leading Aircraft, f		
		Heavy	Medium	Light
Trailing Aircraft, g	Heavy	4	3	3
	Medium	5	3	3
	Light	6	5	3

for aircraft f and g on node n can be given as:

$$C^n_{fg}(\mathbf{x}) = \begin{cases} 1 + \frac{t^{Out}_f - t^{In}_g}{2\max(T^n_f, T^n_g)}, & \text{if } t^{Out}_f > t^{In}_g \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where t^{Out}_f and t^{In}_g denote the node exit time of aircraft f and node entry time of aircraft g respectively on the current node n. T^n_f and T^n_g are the flying time of aircraft f and g when passing through the detection zone of node n.

D. Delay and speed deviation

In real operations, not only the conflicts are important, there are other factors that might have big effects on the whole control system. In our objective functions, the delay and the speed deviation are also included. Flight delay D_f is determined by the time deviation between the RTA and the assigned arrival time of flight f, thus we have:

$$D_f(\mathbf{x}) = |t_f - T^o_f|. \tag{3}$$

.

The constraint of speed deviation can produce fewer changes and ease the workload for both the pilots and the air traffic controllers. The speed deviation V_f of flight f is derived from the following equation:

$$V_f(\mathbf{x}) = \frac{|v_f - v^o_f|}{v^o_f}. \tag{4}$$

E. Objectives

Two objective functions are proposed in accordance with the degree of centralization. The corresponding models referred as the centralized model and the distributed model are therefore established.

1) Objective function of centralized model: The objective function is a weighted sum of the total number of conflicts in the network, the total delay and speed deviations for the flight set. Therefore, it can be represented as:

$$G_C(\mathbf{x}) = \alpha \sum_{\substack{f,g \in \mathcal{F} \\ f \neq g}} \left(\sum_{l \in \mathcal{U}^l_f \cap \mathcal{U}^l_g} C^l_{fg} + \sum_{n \in \mathcal{U}^n_f \cap \mathcal{U}^n_g} C^n_{fg} \right) + \beta \sum_{f \in \mathcal{F}} (D_f + V_f) \tag{5}$$

In our problem, safety is the most important issue that needs to be addressed, thus α should be set much more bigger than β.

2) Objective function of distributed model: The objective function of the distributed model identifies the performance of an individual flight in order to build different objective functions. The flights with the worst performance will be considered with priority. For a given flight f, the weighted sum of the costs of conflicts, delay and speed deviation is regarded as the total cost of this flight and computed as follows.

$$g(\mathbf{x})_f = \alpha \left(\sum_{l \in \mathcal{U}^l_f \cap \mathcal{U}^l_g} C^l_{fg} + \sum_{n \in \mathcal{U}^n_f \cap \mathcal{U}^n_g} C^n_{fg} \right) + \beta \left(D_f + V_f \right), \quad f, g \in \mathcal{F} \tag{6}$$

Where aircraft f and g are supposed to be operated successively. The objective function for the distributed model should guarantee that the cost of the aircraft with the worst performance can be minimized. So the most expensive cost among the flight set is set as the objective:

$$G_D(\mathbf{x}) = \max(g(\mathbf{x})_f), \quad f \in \mathcal{F} \tag{7}$$

IV. Simulated annealing

A. General introduction of simulated annealing

Simulated annealing is a meta-heuristic method to approximate global optimum in a large search space for an optimization problem in a fixed amount of time [18]. The method involves heating and controlled cooling that mimics the process of metal annealing. A critical control parameter in this optimization process is the temperature. The heating process explores the solution space to search for a temperature that can ensure a sufficient and deep exploration during the cooling process. The cooling process refers to the decreasing of the temperature which can be interpreted as a slow decrease in the probability of accepting worse solutions while exploring the solution space.

In SA, several factors should be specified in the implementation:

- The temperature, in a cooling process for instance, follows a geometric pattern such as $T_k = \gamma T_{k-1}$ for consecutive iterations where k denotes the number of iteration and γ is a cooling parameter.
- Neighborhood generation randomly selects candidate solutions in order to sufficiently explore the state space. At each step, the quality of the current solution and neighboring solution will be measured and decided whether the new solution will be accepted or not.
- Acceptance probability for the neighboring solution is specified with a Metropolis criterion. The steps mentioned above are repeated until the temperature is lower than a certain value.

Algorithm 1 provides the pseudo-codes of the cooling procedure for the SA.

In accordance with the structural features when incorporating the models into the solution algorithm, Centralized Simulated Annealing (CSA) and Distributed Simulated Annealing (DSA) are introduced. As the objective function indicated, the costs of all flights are summed up in the centralized model, and flights associate with different levels of costs can not be distinguished. Consequently, the neighborhoods are chosen blindly which yields a large computational burden. However, from a distributed perspective, the cost of each flight is treated individually at each iteration. The flights that associate with costs higher than a standard will be selected as critical flights. The critical flights are targeted with priority for undergoing changes. Moreover, with a guided searching process for critical flights, a better solution can be found much easier and faster. The details of both centralized and distributed simulated annealing are specified as follows.

Algorithm 1 Simulated annealing applied for centralized AMAN

Require: Initialize(initial temperature T_o obtained from heating procedure, number of transitions for each iteration $I = 100$, random number $\beta \in [0, 1]$);

Calculate the initial sequence objective $G(\mathbf{x})$;
$G(\mathbf{x})_c \leftarrow G(\mathbf{x})$
Put the flights who have cost in the set of $\mathcal{F}_c$;
while $T_c > 0.0001 \cdot T_0$, $G(\mathbf{x})$ is not optimum solution **do**
  for $i = 1$ to I **do**
    Choose one flight in $\mathcal{F}_c$, randomly change one of its decision variables;
    Calculate the new objective $G(\mathbf{x})_n$;
    if $G(\mathbf{x})_c > G(\mathbf{x})_n$ **then**
      $G(\mathbf{x})_c \leftarrow G(\mathbf{x})_n$;
    else if $\beta < \exp(\frac{G(\mathbf{x})_c - G(\mathbf{x})_n}{T_c})$ **then**
      $G(\mathbf{x})_c \leftarrow G(\mathbf{x})_n$;
    end if
  end for
  Update flight set $\mathcal{F}_c$;
  $T_c = T_c \cdot \gamma_c$;
end while
return $G(\mathbf{x})$

B. Simulated annealing based on centralized structure

1) Heating procedure: opposite to the cooling procedure, the temperature in the heating loop starts from a small value, then this value is progressively multiplied by a control parameter δ that is slightly greater than 1. Each iteration corresponding to a certain temperature is composed of multiple transitions. As the number of transitions increases, the acceptance rate ζ that is defined as the number of neighborhood solutions accepted divided by the total number of transitions varies. Once the acceptance rate reaches a certain value typically 0.8, the heating process stops and the derived temperature is used as the initial temperature for the cooling procedure.

2) Cooling procedure: during the evaluation of the objective function, the individual cost of each flight is accessible. For each evaluation, a flight set $\mathcal{F}_c$ includes all the flights that have a cost different from 0. The neighborhood solution is chosen by randomly selecting a flight in $\mathcal{F}_c$ and changing one of its decision variables arbitrarily, then a new state is produced and the neighborhood solution can be obtained accordingly. In order to get the new solution, the related information of the chosen flight needs to be removed from the network, after the changing of the decision variable, the flight with new information will be injected in the overall network simulation, where objective of this neighborhood solution is then yielded.

The acceptance of the neighborhood solution is based on a Metropolis criterion which is highly dependent on the current temperature. In the Metropolis algorithm, at each temperature, a large number of transitions are generated in order to achieve the balance in exploring the state space and exploiting solutions.

C. Simulated annealing based on distributed structure

DSA aims to decentralize the flight information by releasing the integration of all the flights in the simulation and focusing on the critical ones which are essential in coping with unexpected situations.

1) Heating procedure: in this model, one simulated annealing algorithm is associated with each flight. The performance of individual flight is used in the heating procedure. In each transition, a flight f is chosen, the associated cost $g(\mathbf{x})_f$ is considered as the current solution. The neighborhood solution $g(\mathbf{x}')_f$ is generated by changing one of the decision variables of flight f. Then the acceptance criterion is applied to determine the acceptability of the neighborhood solution. Flights in the flight set are chosen successively to repeat the aforementioned procedure until the end of this iteration. Similar to the CSA, temperature increases as each iteration finished, and once the acceptance rate reaches a certain value (typically 0.8), the heating process terminates.

2) Cooling procedure: the main difference between the cooling procedure of CSA and DSA is the neighborhood evaluation. To initiate the process, all flights are first evaluated and the highest cost is specified as a reference. Then, for a regular iteration, each flight compares its performance with a cost threshold λC_m. Consequently, the flight that has a cost higher than the cost threshold is selected for the critical flight set. Through targeting the critical flights, the algorithm is efficient in reducing the randomness of selecting the neighborhood. This process is summarized in Fig. 3. The flight set is listed on the x-axis with their associated costs indicated on the y-axis. The maximum value of cost in this iteration is 19.71, thus the critical flight set is composed of the flights that have a cost bigger than 19.71λ. The horizontal red line denotes the threshold of accepting critical flights and the flights with their costs marked by red circles are chosen.

Flights in the critical set are selected successively for each transition, and one of its decision variables is randomly changed which will derive a neighborhood. Instead of evaluating the costs of all flights, a simulation for this particular flight is conducted. The costs of this flight before and after

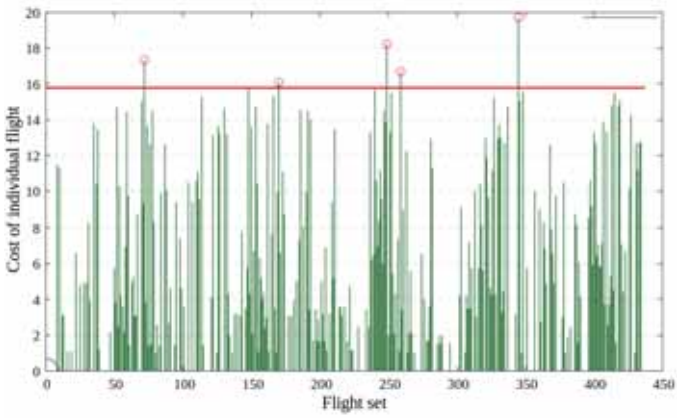

Figure 3. Critical flights selection process in DSA

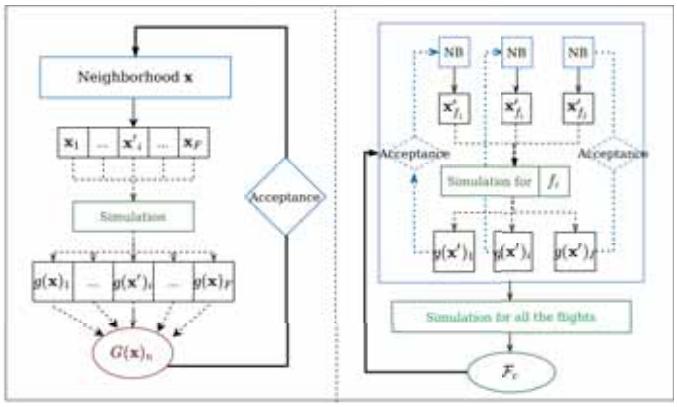

Figure 4. Neighborhood choosing and simulation frame of CSA and DSA

the decision changing are taken as the current solution and the neighborhood solution. The qualities of the two solutions are then measured to decide the result of acceptance. Fig.4 illustrates the structure of CSA and DSA in terms of neighborhood searching and simulation structure.

V. Results and Observations

A. Case study

Actual arrival flights that are operated to runway 26L from a west flow configuration on 11th July 2017 in Pairs CDG airport are applied for our case study. Table. II provides the

Algorithm 2 Distributed neighborhood solution selecting method

Require: Initialize: Initial temperature T_o obtained from heating procedure, random number $\beta \in [0, 1]$;

while $T_c > 0.0001 \cdot T_o$, $C_m > 0$ **do**
  For each flight f, compute the associated cost $g(\mathbf{x})_f$;
  Specifying the maximum cost of an individual flight C_m, and
  for $f \in \mathcal{F}$ **do**
    if $c_f \geqslant \lambda C_m$ **then**
      Put flight f in the critical flight set $\mathcal{F}_c$;
    end if
  end for
  for $f \in \mathcal{F}_c$ **do**
    One of t_f and v_f is chosen with equal probability and change its value.
    Calculating the new cost of f, $g(\mathbf{x}')_f$.
    if $g(\mathbf{x})_f > g(\mathbf{x}')_f$ **then**
      $g(\mathbf{x})_f \leftarrow g(\mathbf{x}')_f$;
    else if $\beta < \exp(\frac{g(\mathbf{x})_f - g(\mathbf{x}')_f}{T_c})$ **then**
      $g(\mathbf{x})_f \leftarrow g(\mathbf{x}')_f$;
    end if
  end for
  $T_c = T_c \cdot \gamma_d$;
  Specifying the maximum cost of an individual flight C_m;
end while

TABLE II
ARRIVAL FLIGHTS INFORMATION

Entry point	Number of flight	Heavy	Medium
MOPAR	60	26	34
LORNI	74	20	54
OKIPA	195	36	159
BANOX	108	20	88
Total number	437	102	335

flight information which contains the flying route and mixed ratio of aircraft categories. Fig. 5 depicts the arrival route network of Paris-Charle de Gaulle (CDG). Flights come from four entry nodes are mostly composed of heavy and medium aircraft. In our case study, chosen parameters for the problem formulation are given in Tab. III. Mention that $\Delta T_{\max}$ has been set as 30 mins to provide a large enough margin to make sure a conflict free result can be achieved. This tardiness seems unreal for short-haul flights as a result of the speed regulation, while if a high tardiness time is required, the ground delay can be considered as a time control measurement. Besides, the parameters of the solution algorithms also need to be specified taking into account the problem size and the desired configuration. The related information is equally give in Tab. IV. The overall process is run on a 2.50GHz core i7 CPU, under Linux operating system PC with Java code. The derived results are investigated in terms of execution time and the solution quality for the two models.

B. Execution time

The main factor that affects the execution time for the optimization process is the time consumed for each simulation in the CSA and the DSA. As defined previously, in CSA, the costs of all the flights are evaluated and summed up, while in DSA, only one flight is fully involved in the simulation without interacting with irrelevant information. This makes the distributed SA hundreds of times faster than the centralized SA in terms of simulation time for each transition.

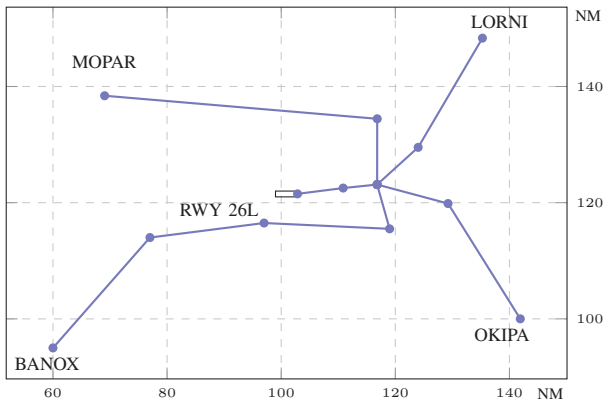

Figure 5. Arrival route structure for runway 26L in Paris CDG airport.

TABLE III
CHOSEN PARAMETER VALUES FOR THE OPTIMIZATION PROBLEM

Parameters of the optimization problem	Values
Maximum tardiness regarding RTA, $\Delta T_{\max}$	30 mins
Maximum earliness regarding RTA, $\Delta T_{\min}$	-5 mins
Time slot, ΔT	5 seconds
Maximum speed in TMA, $V_f^{\max}$	$1.1V_f^o$
Minimum speed in TMA, $V_f^{\min}$	$0.9V_f^o$
Speed slot, Δ_f^v	$0.01V_f^o$
Weighting parameter, α	1
Weighting parameter, β	0.001

TABLE IV
CHOSEN PARAMETERS VALUES FOR SOLUTION ALGORITHMS

Parameters of the solution approaches	Values
Heating control parameter for CSA and DSA, δ	1.1
Heating acceptance rate for CSA and DSA, ζ	0.8
Cooling parameter for CSA, γ_c	0.99
Number of transition in each iteration for CSA, I	100
Critical flights threshold factor for DSA, λ	0.8
Cooling parameter for DSA, γ_d	0.999

In the solution algorithm, each transition contains a simulation for a new state space, therefore the number of transitions in the whole optimization process is essential for the execution time. In the DSA, at the beginning of each iteration, all the flights are evaluated in the simulation to update the overall optimization performance, which updates the value of C_m for the current iteration. Since C_m defines different threshold for the selection of critical flights, the number of transitions which is determined by the flight number in $\mathcal{F}_c$ varies in each iteration. Considering another related parameter $\lambda = 0.8$, it is expected that only a small proportion of flights lies in the critical flight set. However, in the CSA, the transitions for each iteration is fixed as 100. Since the optimization process ends when the temperature reduces to a certain level, the number of iterations is determined by the evolution of temperature. Table.V gives the execution time information of centralized SA and decentralized SA, in which we can see no matter the total number of iterations or the total execution time, the DSA outperforms the CSA.

C. Quality of result

In our problem, the delay and the speed deviations are included in the objective functions. Fig. 6 depicts the conflict

TABLE V
COMPUTATIONAL PROPERTIES FOR CENTRALIZED AND DISTRIBUTED SA

	Distributed SA	Centralized SA
Nb of iterations	9227	1146
Nb of transitions	32386	114600
Total execution time (s)	46.876	1157.663

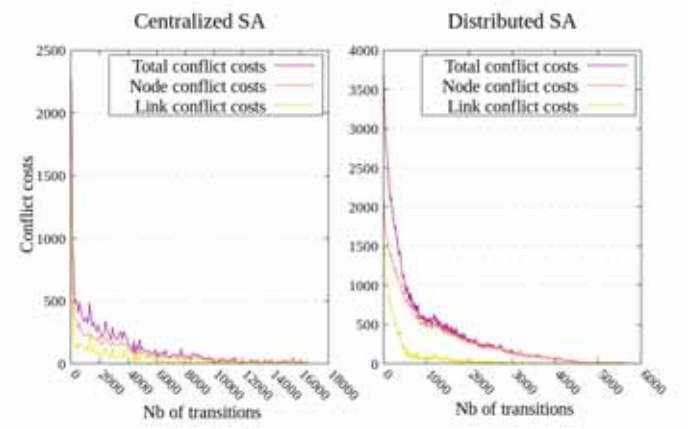

Figure 6. conflicts evolution of the centralized SA and the distributed SA.

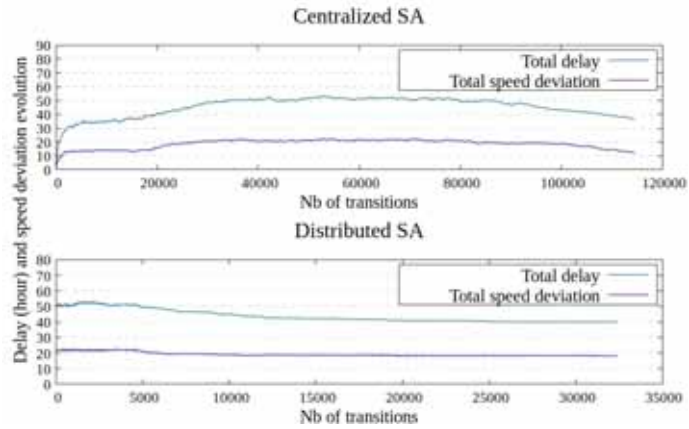

Figure 7. Delay and speed deviation evolution of the centralized SA and the distributed SA.

TABLE VI
COMPUTATIONAL TIME FOR CENTRALIZED AND DISTRIBUTED SA

	Distributed SA	Centralized SA
Delay (hour)	39.9625	36.675
Speed deviation (%)	18.1699	12.16

evolution of CSA and DSA respectively in which the conflicts have been eliminated in both cases. In Fig. 7, the total delays and speed deviations with respect to the number of transitions are displayed. Since the conflicts have a higher weight, the delay and speed deviation are mainly targeted after the elimination of conflicts which explains the variation trends in the figure. The final results in Tab. VI show that under the current setting of parameters, the CSA provides a result with less delay time and speed deviation than DSA, where the longer execution time certainly helps the optimization of the results. Besides, those numbers represent the cumulative delay for all the flights and we can expect a small value for a single flight.

VI. CONCLUSION

This paper addresses the scheduling problem for arrival flights in TMA. The TMA entry time and the TMA entry speed are set as decision variables. In order to be consistent with the real operations, safety has been considered as our priority. Besides, the delay and speed deviations are considered with a minor weight in the objective function as well. Since this problem is a complex NP-hard one, the computational efficiency is highly required to adapt to real cases.

Our work focuses on the improvement of algorithmic efficiency to better fit in the possible decision support tools. The traditional centralized AMAN are reconsidered with respect to the level of centralization, then a model with a rather distributed structure is proposed. Opposite to the centralized model which integrates the information of all flights and gets a full awareness of the performances of the flights at each transition, the distributed AMAN targets the individual flights and reduces unnecessary underlying communication between irrelevant flights. Both models are implemented for a heuristic simulated annealing to evaluate the efficiency. The full-day data of Paris CDG on one landing runway has been applied as the case study. The results of the two models are presented, where the execution times and the qualities of the results are demonstrated. Tests show that the distributed AMAN has an absolute advantage over centralized AMAN in computational time with a slightly dropped performance in minimizing the delay and speed deviation.

The future work lies in providing more tests with different scenarios and conducting a deep investigation on the attributes of distributed AMAN. We hope that the algorithm can be applied to the online pre-tactical support tool. The decentralized AMAN may be also very useful in the Unmanned Traffic Management (UTM) framework for which there is no centralized AMAN.

ACKNOWLEDGMENT

The author gratefully acknowledges the support of our colleagues who have provided tremendous suggestions on the paper writing.

REFERENCES

[1] Y. Huo, D. Delahaye, M. Sbihi, and Y. Wang, "Air traffic flow management under uncertainty in terminal maneuvering area," in *ICRAT 2020, 9th International Conference for Research in Air Transportation*, 2020.

[2] J. Ma, D. Delahaye, M. Sbihi, and M. Mongeau, "Integrated optimization of terminal manoeuvring area and airport," in *6th SESAR Innovation Days (2016).*, pp. ISSN–0770, 2016.

[3] L. Bianco, P. DellOlmo, and S. Giordani, "Scheduling models for air traffic control in terminal areas," *Journal of Scheduling*, vol. 9, no. 3, pp. 223–253, 2006.

[4] M. Sama, A. DAriano, F. Corman, and D. Pacciarelli, "Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 485–511, 2017.

[5] Y. Hong, *Determinstic and Stochastic Optimization for Aircraft Arrival Sequencing and Scheduling under Uncertainty*. PhD thesis, Seoul National University, 2018.

[6] A. D'Ariano, P. D'Urgolo, D. Pacciarelli, and M. Pranzo, "Optimal sequencing of aircrafts take-off and landing at a busy airport," in *13th International IEEE Conference on Intelligent Transportation Systems*, pp. 1569–1574, IEEE, 2010.

[7] J. Ma, D. Delahaye, M. Sbihi, and M. Mongeau, "Merging flows in terminal moneuvering area using time decomposition approach," 2016.

[8] C. Zuniga, D. Delahaye, and M. A. Piera, "Integrating and sequencing flows in terminal maneuvering area by evolutionary algorithms," in *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*, pp. 2A1–1, IEEE, 2011.

[9] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Scheduling aircraft landingsthe static case," *Transportation science*, vol. 34, no. 2, pp. 180–197, 2000.

[10] M. Samà, A. D'Ariano, P. D'Ariano, and D. Pacciarelli, "Comparing centralized and rolling horizon approaches for optimal aircraft traffic control in terminal areas," *Transportation Research Record*, vol. 2449, no. 1, pp. 45–52, 2014.

[11] K. Ng, C. K. Lee, F. T. Chan, and Y. Lv, "Review on meta-heuristics approaches for airside operation research," *Applied Soft Computing*, vol. 66, pp. 104–133, 2018.

[12] J. Ma, D. Delahaye, M. Sbihi, P. Scala, and M. A. M. Mota, "Integrated optimization of terminal maneuvering area and airport at the macroscopic level," *Transportation Research Part C: Emerging Technologies*, vol. 98, pp. 338–357, 2019.

[13] M. Xiangwei, Z. Ping, and L. Chunjin, "Sliding window algorithm for aircraft landing problem," in *2011 Chinese Control and Decision Conference (CCDC)*, pp. 874–879, IEEE, 2011.

[14] K. Krishna, K. Ganeshan, and D. J. Ram, "Distributed simulated annealing algorithms for job shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 7, pp. 1102–1109, 1995.

[15] M. E. Aydin and T. C. Fogarty, "A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application," *Journal of intelligent manufacturing*, vol. 15, no. 6, pp. 805–814, 2004.

[16] J. Davin and P. J. Modi, "Impact of problem centralization in distributed constraint optimization algorithms," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 1057–1063, 2005.

[17] N. Hasevoets and P. Conroy, "Extended amansesar deployment managerworkshop18thseptember 2018, brussels," *EUROCONTROL, Brussels, Tech. Rep*, 2010.

[18] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*, pp. 7–15, Springer, 1987.