Memetic Comp.
DOI 10.1007/s12293-015-0172-z

REGULAR RESEARCH PAPER

Sequence searching and evaluation: a unified approach for aircraft arrival sequencing and scheduling problems

Xiao-Peng Ji[1] · **Xian-Bin Cao**[1] · **Ke Tang**[2]

Received: 9 June 2015 / Accepted: 13 November 2015
© Springer-Verlag Berlin Heidelberg 2015

Abstract Arrival sequencing and scheduling (ASS) is an important part of air traffic control. In the literature, various formulations of the ASS problems have been established by taking different scheduling requirements into account, and various methods have been developed to cope with these ASS problems. However, it is usually uneasy to generalize a method designed for one ASS formulation to another, while an approach that is able to handle different ASS problems is of great significance since air traffic controllers may need to switch among different scheduling requirements in practice. Motivated by this observation, an approach that is applicable to a number of different problem formulations of ASS is proposed in this paper. Specifically, the ASS problems that include different objective functions and constraints are firstly abstracted as a constrained permutation-based problem. After that, a Sequence Searching and Evaluation (SSE) approach is developed for the constrained permutation-based problem. The SSE solves different ASS problems by separating the sequence searching in one stage using an Estimation of Distribution Algorithm framework, and evaluating sequences in the second stage. Experiment results show that SSE is capable of obtaining competitive solutions for a variety of ASS problems.

✉ Xian-Bin Cao
jixiaopengyz@163.com

Ke Tang
ketang@ustc.edu.cn

[1] School of Electronic and Information Engineering, Beihang University, Beijing 100191, People's Republic of China

[2] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, People's Republic of China

Keywords Air traffic control (ATC) · Aircraft arrival sequencing and scheduling (ASS) · Permutation-based problem · Estimation of distribution algorithm (EDA) · Sequence Searching and Evaluation (SSE)

1 Introduction

Due to the constant development in air traffic over the past few decades, the workload of air traffic controllers in terminal areas is increasing heavily. Especially in high-density airports, congestion is more frequently generated, which leads to less operation efficiency and more airborne delay [16]. Aiming at improving the air traffic control (ATC) system, a major reform has been occurred in the world, such as NextGen in US, SESAR in Europe and transformation of the Chinese airspace [61]. One commonly adopted approach to ensure the safety and order of the arrival traffic is to reassign the landing time of the arriving aircraft to alleviate congestion, namely aircraft arrival sequencing and scheduling (ASS). Due to its complexity and importance, ASS is still one of the core problems in the research of ATC [26].

In the past few decades, many research efforts have been made to tackle the ASS problems. According to different focuses, the ASS problems are formulated as various kinds of mathematical models. Considering of the optimization objective functions, the ASS models can be classified into minimizing the total airborne delay (total delay time of all arriving aircraft) [5–7], extra costs (costs generated by the early/ late landings) [1–3] or process landing time (time required for all landings) [6–8] and so on. At the same time, aimed at one of these models, diverse optimization algorithms have been proposed, such as mixed-integer programming [1–3] and various intelligent optimization algorithms [5–13]. However, the forms of current ASS mod-

Published online: 26 November 2015

els are not unified since many factors, including different decision variables, objectives, and constraints, need to be taken into account while formulating an ASS problem. Different considerations of these factors will lead to different optimization problems. All these existing ASS models are surely reasonable that the researchers considered the ASS problem from different perspectives. Meanwhile, the methods proposed for these models have also achieved great effects through specially designed optimization operations. In fact, due to the constant change of the practical operation environment in the airport, such as the amount of arriving aircraft and airport capacity, different scheduling requirements may be generated. When faced with different situations, it is difficult for air traffic controllers to change strategy rapidly to make appropriate scheduling decisions. Therefore, designing a general approach that is not limited to the scenario description is of great significance.

Most of the current studies are carried out with "directly rearranging the assigned landing times (ALT) for the aircraft". In this way, the landing order of aircraft and the related constraints will be considered at the same time in the algorithm process. That is why a method proposed for one model may be not suitable for another, since the algorithm for each model is designed specially based on handling the constraints. In other words, if one algorithm does not contain the operator for specific constraints, it will not be suitable for the models that have considered them. For instance, the same route ordering constraint, which requires the aircraft coming from the same jet route to remain their order in the landing sequence [8,9,12,14], is not considered in the model investigated by Xiao-bing Hu et al. [5–7]. Consequently, applying the receding horizon control (RHC) and its several extended algorithms to ASS problems that involving this constraint may get infeasible solutions. For example, suppose that two aircraft A and B come from the same jet route and A is in the front. Without considering the ordering constraint, if a lower airborne delay is obtained when B is scheduled in front of A, the algorithm will probably obtain a landing sequence to schedule B in the front. However, it is actually an infeasible solution since it does not satisfy the same route ordering constraint.

Differing from previous works, this paper focuses on designing a general algorithm framework that is not subject to the limits of the specific model and can be easily adapted to various ASS models. This approach is not required to perform much better than the existing methods, but it should be able to generalize well to different ASS models and generate acceptable results in terms of both efficacy and efficiency. Concretely, since all different ASS problems can be formulated as constrained permutation-based optimization problems [4,32], an extensible framework to solve the permutation-based ASS problems is proposed. Accordingly, a number of methods, namely Sequence Searching and Evaluation (SSE) algorithm, are developed. It is a memetic algorithm that combines the estimation of distribution algorithm (EDA) and a local search approach [41,42]. In most ASS problems, when a landing sequence is determined, the optimal landing time of each aircraft in the sequence will also be obtained accordingly. In view of this, the solving process can be divided into two parts. The optimal sequence is firstly obtained by a searching progress of a general optimization method, in which the related objective and constraints will be temporarily ignored. After that, the solution form, namely the landing time of each aircraft, is obtained by optimizing the objective with consideration of the landing order and constraints. In this way, the landing time of each aircraft in the sequence will be assigned one by one. Although the procedure of landing time assignment may vary depending on different objectives or constraints, it can be done in polynomial time for most cases investigated in the literature. The detailed operations of transforming a sequence into a solution form is given in Sect. 3.3. In this way, various kinds of ASS problems can be solved by the SSE framework. Generally, the problem solving process will be the iteration of two steps, as shown in Fig. 1:

1. Abstract the ASS problem into a permutation-based problem, and search new specific optimization sequences according to certain rules;

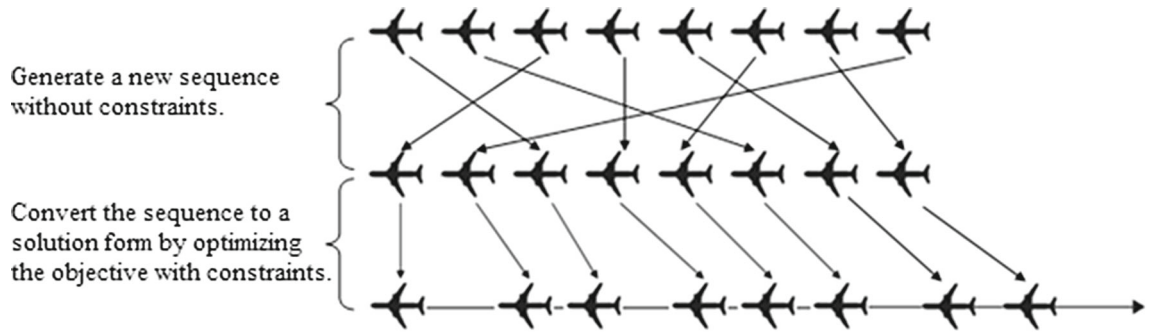

Fig. 1 An example of the two steps in the iteration

2. Based on the description of the objective and constraints, transform the sequences into the solution form, namely the assign landing time for each aircraft, then evaluate their qualities and reserve better ones.

Compared with current studies, SSE has the following characteristics: The position relation of the aircraft is considered directly instead of being described in constraints. Furthermore, the constraints are considered separately and the final solution can be obtained by converting the sequence to an optimal answer according to the scenario described. These characteristics make the algorithm suitable for various kinds of ASS problems. In order to verify the feasibility of this method, a series of experiments have been made. The results indicate that SSE performs well in solving different kinds of ASS models.

The rest of this paper is organized as follows. Section 2 describes the details of the ASS problem and summarizes the previous work. Then the framework of SSE is presented in Sect. 3. Simulation results and the performance comparison with other algorithms are given in Sect. 4. Finally, a conclusion is summarized with future work in Sect. 5.

2 Background

In airports, a simple way to solve the ASS problem is to schedule the arriving aircraft on a rule of first-come-first-served (FCFS) by the predicted landing time (PLT). Although the FCFS scheduling gives a fair order, it ignores large amounts of useful information that can make use of the capacity of the airport [15,16]. For instance, shifting aircraft position according to a landing time interval (LTI), that is the minimum permissible time interval between two successive landings, will significantly reduce airborne delay [17–19]. For this reason, current research about ASS problem is mostly based on rearranging the assigned landing time (ALT), thereby changing the landing sequence. Various models have been established including kinds of objectives and constraints as follows.

2.1 Objective functions of existing ASS problems

Many studies only focus on those delayed aircraft. Sometimes, the aircraft are considered to be unallowable to land in advance. An objective function to minimize the total airborne delay is described as

$$T_{delay} = \sum_{i=1}^{N} [\max(ALT(i) - PLT(i), 0)], \tag{1}$$

where N is the number of all the landing aircraft, *PLT(i)* and *ALT(i)* denote the predicted and assigned landing time of aircraft i, so that T_{delay} represents the total airborne delay of all the arriving aircraft [5–7,20–22].

Meanwhile, more researchers have considered all the aircraft whether they are in advance or delay. It is thought that when the aircraft land before or after their PLTs, extra costs will be generated. In order to minimize the total cost of deviations from the PLTs, an objective function of

$$c = \sum_{i=1}^{N} (g_i \max\{0, PLT(i) - ALT(i)\} + h_i \max\{0, ALT(i) - PLT(i)\}) \tag{2}$$

is given by Beasley et al. to represent the total extra costs. In the above function, g_i is the penalty cost per unit of time for aircraft i landing before*PLT(i)*, and h_i is for landing after *PLT(i)* [1–3,9–13,23–25,43–48,51,53,54].

Different from the above, a few studies focus on the required landing time. Sometimes when the terminal area is busy, air traffic controllers always tend to land all the aircraft as soon as possible in order to ease congestion. For this purpose, some researchers have focused on the ASS problem that is formulated to minimize the process landing time. That is

$$T_{length} = \max[ALT(1), \ldots ALT(N)] - \min[ALT(1), \ldots ALT(N)]. \tag{3}$$

In Eq. (3), only the ALTs of the first and last landing aircraft are considered. This objective function will make aircraft finish landing as soon as possible [6,7,26,27]. Alternatively, some researchers also formulated this problem as another form:

$$T = \sum_{i=1}^{N-1} \delta_{i,i+1}.$$

where $\delta_{i,i+1}$ is the LTI between aircraft i and $i+1$ [8,14]. However, it is quite different from Eq. (3) that this function supposes aircraft are landing one after another without other time constraints. Then the sum of the LTIs in the sequence will be the total processing landing time. Similarly, the makespan of all the landings is also chosen as the objective in some works. It concentrates on the "absolute time" instead of "relative time" of all landings [50,52], namely

$$T = \max[ALT(1), \ldots ALT(N)]$$

2.2 Constraints of existing ASS problems

Taking different scenarios into account, various constraints have been considered in current studies. The landing time interval (LTI) is an indispensable constraint for almost all

Table 1 Minimal LTI between the aircraft [18]

LTI (i, j) Seconds		Type of the later landing aircraft j			
		1	2	3	4
Type of the earlier landing aircraft i	1	96	200	181	228
	2	72	80	70	110
	3	72	100	70	130
	4	72	80	70	90

1 Boeing 747, *2* Boeing 727, *3* Boeing 707, *4* Mc Donnel Douglas DC9

research works. The minimum safety separation between each pair of aircraft in the sequence must be satisfied [1–3,9,10,13,24,25,28,43–48,51–53].

$$ALT(i) - ALT(j) \geq \delta_{j,i} \; or \; ALT(j) - ALT(i) \geq \delta_{i,j}, \quad i, j \in \{1, \ldots, N\}, i \neq j. \tag{4}$$

The LTI is a function of the types of two successive landing aircraft. One common standard of the LTI is shown in Table 1. Note that, an aircraft will affect only one adjacent aircraft landing after it. In other words, $\delta_{i,j} + \delta_{j,k} \geq \delta_{i,k}$ is always satisfied for any types i, j and k ($i, j, k = \{1, 2, 3, 4\}$). As a consequence, only the LTIs between adjacent landing aircraft are considered in some works. Then the constraint above can be rewritten as

$$ALT(j) - ALT(i) \geq \delta_{i,j}, \tag{5}$$

if i, j are successive landing aircraft that aircraft j is after aircraft $i, i, j \in \{1, \ldots, N\}$ [5–8,12,14,20–23,26,27,29,30,50,54].

In a few works, aircraft are not allowed to land before their PLTs. The constraint can be described as

$$ALT(i) \geq PLT(i). \tag{6}$$

Namely, all the aircraft should not land in advance under the premise of the LTI constraints [5–7,20–22,26,27,29,30].

Some researchers have also mentioned that aircraft are coming from several different jet routes. For those aircraft coming from the same route, they should remain their order in the landing sequence. Back aircraft are not allowed to land before the front aircraft. Otherwise, it will bring security risks and it is also contrary to the principle of fairness. This constraint

$$(ALT(j) - ALT(i))(PLT(j) - PLT(i)) > 0 \tag{7}$$

describes the position relations between i and j, if i, j are from the same jet route [8,9,12,14,50].

The earliest/latest landing time is also considered in many studies. Since aircraft can not put up acceleration and deceleration in large range duo to the restriction at the terminal area and aircraft fuel capacity, they are only allowed to land in a fixed time interval, namely

$$ALT(i) \in [E_i, L_i], \tag{8}$$

in which E_i and L_i denote the earliest and latest landing time for aircraft i respectively [1–3,10,23,24,27,31,43–48,50–54].

2.3 Existing methods for ASS problems

In order to solve different ASS problems, many methods including the use of mathematical programming and intelligent optimization algorithms have been proposed.

An early research of the ASS is the work of Psaraftis' [55,56], which investigated a simplified version of ASS and modeled it as a TSP problem. These works assumed that all the flights are available to land immediately. The flights are classified by their aircraft types, and the same type of flights are considered to be equivalent in the scheduling progress. By merging the lists of flights within different aircraft types, an algorithm is proposed to solve this simplified ASS problem.

The constrained position shifting (CPS) environment, which is first introduced by Dear [57], is improved for solving the ASS problem [40]. The CPS restricts that each flight can only change their positions of the FCFS order within a specific range in the optimal sequence. Considering the CPS and time window constraints, a dynamic programming approach is presented to minimize the makespan of the arrival scheduling.

Beasley et al. attempt to solve the ASS problem by using mixed-integer zero-one formulations [1–3]. The mixed-integer programming is flexible to take more constraints into account, such as the permissible landing time window and the minimum separation time. Taking the makespan as the objective, the ASS can be viewed as a job shop scheduling problem that can be solved by memetic algorithms [59,60]. This formulation is acknowledged as one of the most classic models in the field of ASS research. Therefore, many research efforts on the mathematical programming have been made to solve this problem [43,49,51,53,54]. Although mathematical programming also has great universality, its relatively low efficiency will become the greatest performance bottlenecks of this method, especially used to solve a large-scale problem.

In more research works, intelligent optimization algorithms have been applied to solve the ASS problem [5–7,10,14,22,23,31]. For instance, taking the function (1) as the objective and (5), (6) as the constraints, the receding horizon control (RHC) is adopted by Xiao-bing Hu to

solve the ASS problem [7]. The RHC is an N-step-ahead optimization strategy and it divides the ASS problem into a number of sub-problems. He also introduces the RHC into Genetic Algorithm to solve the dynamic ASS problem, including both the permutation-representation and binary-representation GAs [5,6]. Furthermore, an ant colony system (ACS) based on the RHC is also developed by Zhi-hui Zhan et al. to improve the performance of the algorithm [20]. However, the application of these intelligent optimization algorithms still has some problems. As mentioned earlier, while dealing with different models, the corresponding operator needs to be specially designed to handle the constraints. A method adopted by one model cannot be simply applied for solving another problem. In addition, if the characteristics of the algorithm do not fit the essence of the problem, the efficacy and efficiency of the method will also be greatly affected.

3 A general framework for solving ASS problems

3.1 Basic idea

Since the ASS problem aims at determining the ALT for each aircraft, the existing ASS models are mostly expressed in the form of ALTs. In such a formulation, the position relationship of the aircraft between each other is not fully considered but described by the constraints. However, a common characteristic of the existing formulations for ASS is that they are essentially constrained permutation-based optimization problems. The relative position relations between the landing aircraft should be investigated and utilized. No matter in which ASS model, the final solution contains two kinds of information, namely the position of the aircraft in the sequence and the time constraints. Moreover, if the optimal landing sequence is determined, the solution can also be obtained according to the objective and constraints, namely the ALT for each aircraft. From this viewpoint, the difference between each model mainly lies in the assignment of ALTs for a landing sequence. For this reason, the process of solving ASS problem can be divided into following two steps:

1. The landing sequences are viewed as decision variables in the searching process so that the solution space is composed with all possible sequences, i.e., all permutations of the landing aircraft. A population of randomly generated sequences will be firstly initialized. Then new sequences will be generated according to a generally approach. In this step, the constraints are temporarily ignored.
2. An evaluation module is designed for converting a sequence to the solution form and assessing the quality of the solution. After determining a landing sequence, firstly the landing time for the first aircraft in the sequence is assigned. Then it assigns the landing time of the second aircraft according to optimizing the objective with constraints. By that analogy, the landing time of each aircraft in the sequence can be obtained step by step. After evaluating the quality of all the sequences, those sequences with better quality will be retained in the population and participate in the next sequence searching process.

By iteratively executing the above two steps, a final best solution for the problem will be obtained. A process diagram to describe the thought of the Sequence Searching and Evaluation algorithm (SSE) is given in Fig. 2.

A complete solution form of the ASS problem contains two aspects of information: the landing order and specific landing time. A sequence (permutation) can only give the landing order of the related aircraft so that the landing time of each aircraft need to be assigned in other ways. For example, a sequence of "$a-b-c$" represents that Aircraft a is the first to land, b is the second and c is the third. But the actual

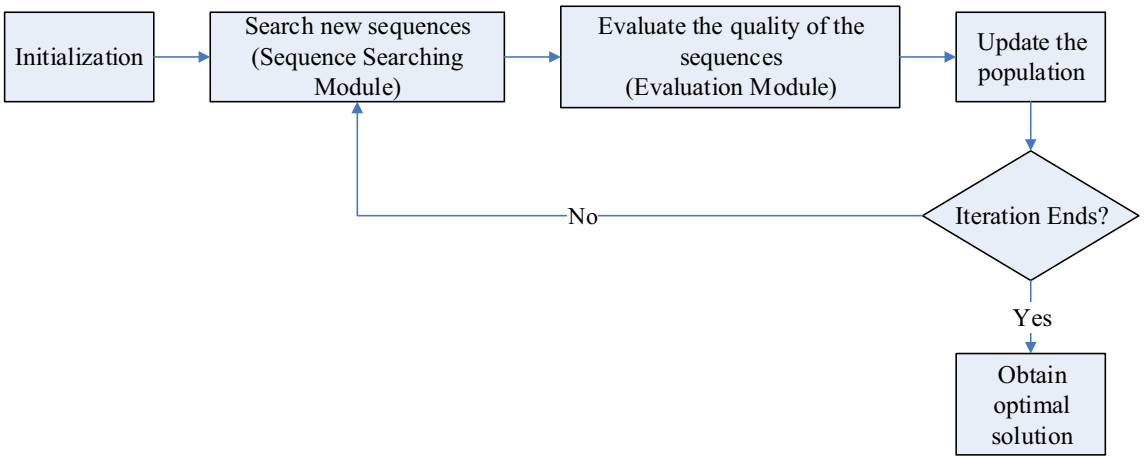

Fig. 2 The flowchart of SSE

landing time of each aircraft is still unknown by this representation. The process of converting a sequence to a solution form is not exactly the same in different ASS models. In this paper, a converting approach is designed by the classification of optimization objectives and the detail process is given in Sect. 3.3. Then the complete solution form will be obtained by the converting process.

In this method, no certain operator for constraints needs to be specially designed in the searching process. The feasibility and quality of the solution are only reflected in the evaluation module. For different factors, simply changing the computing process of the evaluation module accordingly will make various kinds of ASS problems turn into special cases of our framework. As can be seen from the above diagram, the key of SSE lies in the design of the two modules, namely the sequence searching module and the evaluation module. In next parts, the design process will be introduced in detail.

3.2 Sequence searching

Considering the universal characteristic of the method in various mathematical models, estimation of distribution algorithm (EDA) is adopted as the sequence-searching module in this paper. EDA is a kind of novel stochastic optimization algorithms and has recently become a hot topic in field of evolutionary computation [35]. Unlike other algorithms, a probabilistic model is learnt from the population to express the interrelations between the solutions in EDA. By sampling this probabilistic model, a new offspring is obtained. It iterates to get the optimal solution until the pre-defined stop point is met. This algorithm offers a general framework to deal with complex system optimization without lying on detailed domains. In addition, it can settle the problem that has multivariate interaction itself and does not need any information except the objective function. In view of these characteristics, the feature of EDA is quite consistent with our requirements. In recent years, many efforts have been made to deal with permutation-based problems by means of EDA, such as Univariate Marginal Distribution Algorithm (UMDA), Estimation of Bayesian Networks Algorithm (EBNAs), and Mutual Information Maximization for Input Clustering (MIMIC) [34–36,62].

In this paper, the Mallows Model and Kendall-τ Distance are used under EDA since they are applicable for the permutation-based ASS formulation [41,42]. The Mallows Model is a distance-based exponential probability model over permutation spaces. Given a distance d over permutations, the Mallows Model can be defined by the central permutation σ_0 and a spread parameter θ [33]. Specifically, an explicit form of the probability distribution over the space of permutations is shown as follows:

$$P(\sigma) = \frac{1}{\psi(\theta)} e^{-\theta d(\sigma,\sigma_0)},$$

where σ represents a permutation and $\psi(\theta)$ is a normalization constant, namely

$$\psi(\theta) = \prod_{j=1}^{n-1} \frac{1 - e^{-(n-j+1)\theta}}{1 - e^{-\theta}}.$$

The Kendall-τ Distance is the most commonly used distance with the Mallows Model [33]. Given two permutations σ_1 and σ_2, the Kendall-τ distance counts the total number of pairwise disagreements between them, i.e., the minimum number of adjacent swaps to convert one into another, namely

$$d(\sigma_1, \sigma_2) = \sum_{i<j} I\{(\sigma_1(i) - \sigma_1(j))(\sigma_2(i) - \sigma_2(j)) < 0\}.$$

An initial population is firstly generated by a certain amount of random permutations. Each permutation represents a possible (but not necessarily feasible) landing sequence of the current related flights. At each step of EDA, a Mallows model will be learnt from the population, i.e., the set of permutations, and then new permutations will be randomly generated by sampling the probabilistic model.

Specifically, the process of SSE includes following steps:

1. A central permutation σ_0 is firstly obtained according to all the individuals in the population. The problem of finding the maximum likelihood estimation of the central permutation is NP-hard. There are several methods for solving this problem. In [37], 104 algorithms have been compared and the best one is adopted in this paper: The average value at each position is firstly calculated. Then the first position of σ_0, i.e. $\hat{\sigma}_0(1)$ is assigned to the position with the lowest average value and $\hat{\sigma}_0(2)$ is assigned to the second lowest position and so on. The central permutation will be finally obtained when all the values are assigned;
2. After obtaining the central permutation σ_0, new sequences are generated by sampling the probability distribution P defined by σ_0 and a predefined spread parameter θ. Once determining a distance d by the probability model, the SSE should generate a permutation which has a distance of d with the central permutation;
3. At last, the feasibility and quality of both the original and new generated sequences are evaluated and the population are updated by truncation selection;

The iteration of the above three step ends when a given stop criterion is met.

The spread parameter θ is the key to control the search performance of exploration and exploitation. When the value of θ increases, the probability tends to concentrate on the

central permutation σ_0. In order to find an appropriate value of θ, a series of values have been tested in the experiment to analyze the behaviors. The detailed operation and results are given in Sect. 4.3.

3.3 Sequence evaluation

Another important part of SSE is the evaluation module. This module converts the sequences into a feasible solution form depending on the scenario described in the ASS problem, and then evaluates the quality of the sequences. The computing process may be slightly different in different ASS problems. Generally, it can be classified by the objectives as follows.

(a) Total airborne delay.

(a.1) Set the PLT of the first aircraft in the sequence as its ALT;
(a.2) According to relevant constraints (such as the minimum of safety separation, the earliest/latest landing time and the relations between PLT and ALT), select a point in time that is closest to the PLT of the second aircraft in the sequence. Then set it as the ALT of the second aircraft;
(a.3) By analogy, determine the ALTs for the rest of the aircraft in the sequence.

(b) Process landing time.

(b.1) Set the PLT of the first aircraft in the sequence as its ALT;
(b.2) According to relevant constraints, select an earliest point in time. Then set it as the ALT of the second aircraft;
(b.3) By analogy, determine the ALTs for the rest of the aircraft in the sequence.

(c) Total extra costs.

(c.1) Set the PLT of the first aircraft in the sequence as its ALT;
(c.2) For the PLT of aircraft i in the sequence ($i \geq 2$), judge whether it meets the separation constraints with its previous aircraft.
(c.3) If the constraint is satisfied, Set the PLT of the aircraft i in the sequence as its ALT;
Otherwise, according to relevant constraints, judge whether the previous aircraft can be scheduled to land earlier or not.
(c.3.1) If it could be and $g_{i-1} \leq h_i$, namely less costs will be caused if we schedule aircraft $i-1$ to land earlier instead of delay aircraft i, select a time point for aircraft $i-1$ that causes less costs. Then set the ALT of aircraft i according to the separation constraints;
(c.3.2) If not, delay aircraft i to make it meet the separation constraints with aircraft $i-1$.
(c.4) By analogy, determine the ALTs for the rest of the aircraft in the sequence.

According to the above rules, a sequence will be converted to a solution form. But in some kinds of problems, not all permutations can be converted to a feasible solution. For instance, a few constraints such as the earliest/latest landing time restrictions and the jet route ordering constraints will apparently make some permutations infeasible. For this reason, the module must also have the ability to eliminate infeasible solutions in these cases. Hence, after converting process by the above method, a judgment is made according to these relevant constraints. If the solution does not satisfy the constraints, the fitness of the sequence will be set as an infinitesimal manually. Then the infeasible solutions will soon be eliminated during the iteration.

3.4 Other improvements for ASS problems

1. Initialization.

Due to the iterative nature, the individuals in the original population will also affect the quality of the solution. According to this characteristic, the initialization process of the original sequence individuals can be improved to accelerate convergence, so as to make the algorithm more suitable for applications. It is obvious that in various kinds of the ASS models, the fitness of a sequence will be much lower if the original positions of the aircraft have changed a lot. Inspired from the work [40], a position restriction has been added in the initialization process of SSE. Suppose a population of N_k aircraft needs to be initialized and these aircraft could be firstly divided into several groups according to their PLTs. For example, the aircraft that arrive in every 10 minutes can be considered as a group so that a series of groups can be obtained, in which the number of aircraft may be slightly different. Then permutations are randomly generated in each group so that the complete permutations of the N_k aircraft can be obtained by combining the permutations of different groups. This simple operation restricts the positions of the aircraft in a certain range on the basis of the randomly generated permutations, so as to make the initial individuals have higher fitness. Meanwhile, a few sequences obtained by FCFS are also added to the original population since the best results can always be obtained by this way in non-busy situations.

2. Local search module.

Since there is certain randomness in the EDA searching process, a local search module is also needed to improve the efficiency. A quite simple but effective way to get a bet-

ter solution is "sliding and local permutations". Concretely, a sliding window consisting of k aircraft is used to factorize the sequence to a few sub-sequences. Full permutations of this k aircraft will be made to get a current optimal solution. Then, the sliding window will push ahead one position to get another k aircraft and repeat the operations until the end of the sequence. Since the local search module does not contain any specific operators for constraints, it is also universal for various kinds of ASS problems naturally. However, it may be much time consuming since a permutation of N aircraft needs $(N-k+1)*k!$ times of fitness estimations. Therefore, it should be applied with very few times in the whole solving process. In this paper, it is set to work in every certain number of EDA iterations.

3.5 Flowchart of the algorithm

Integrating the modules mentioned above, it can be seen that the SSE is a kind of memetic algorithm that includes a global and a local search modules [58]. The process of SSE for ASS problems is as follows and the pseudocode is shown in Fig. 3.

1. Population generating: According to the initialization module, randomly generate M different sequences to initialize the population *pop*. M is the size of the population;
2. Probability model initialization: Initialize the probability model P based on Mallow's model and Kendall-τ distance. Then set *gen=0*;
3. Sequence searching: Get the central permutation σ_0 of the current population. According to the probability distribution P, M new sequences are randomly generated by confirming the respective distance d;
4. Sequence evaluation: Sort the $2M$ individuals according to their fitness. Then keep M higher fitness individuals and update *pop*. Then *gen=gen+1*;
5. Local search: To get better solutions, put the sequences in *pop* into the local search module in every certain number of iterations;

```
Input: Original flight data, datasize N, population size M, maxgen
Output: A feasilbe solution pop(1)
// Initialization:
1 Randomly generate M different sequences according to the initialization module:
  pop=[pop(1), pop(2), ..., pop(M)];
2 Initialize the probability model based on Kendall-tau distance:
  P=[P(1), P(2), ..., P(N*(N-1)/2+1];
// Main Loop:
3  Set gen=0;
4  while gen<maxgen do
5    Get the central permutation σ0 of pop;
6    for i=1:M do
7      t=rand(1); 0<=rand(1)<=1;
8      if t<P(1) then
9        newpop(i)=σ0;
10     else then
11       for j=2:N*(N-1)/2+1 do
12         if P(j-1)<t<=P(j) then
13           Generate a new sequence newpop(i), and its distance between σ0 is j;
14         end if
15       end for
16     end if
17   end for
18   Evaluate the fitness of all the sequences in pop and newpop;
19   Keep M sequences which have higher fitness, and update pop;
20   gen=gen+1;
21   if mod(gen)==10 then
22     Put pop in the local search module;
23   end if
24 end while
25 Output the best solution in pop, and convert it to the ALT form.
```

Fig. 3 Pseudocode of SSE

6. If the iteration ends, output the best solution in *pop*, otherwise return to Step. 3.

4 Simulation results

In order to verify the efficacy of SSE, empirical studies have been conducted. Firstly, a series of experiments have been made to compare the performance with other classic algorithms. Two kinds of ASS problems are chosen in the comparison experiments, which are commonly adopted in many research works. Then, the effects of parameters on the performance of SSE are analyzed. All the simulation is implemented in MATLAB 7.1 on a notebook computer running an Intel(R) Core(TM) i7-Q720 CPU at 1.60GHz with 4.0-GB RAM.

Due to the great flexibility, mathematical programming is widely used in solving various optimization problem including the ASS. One representative study in recent years may be Faye's new approach, which is based on an approximation of the separation time matrix and on time discretization [53]. Meanwhile, intelligent optimization algorithms are also adopted in a large amount of ASS studies. A series of classic studies may be the RHC with its extended works (RHC-GA, RHC-BRGA, RHC-ACS-ASS) [5–7,20]. By using a sliding time window, the RHC framework divides the original ASS problem into several sub-problems and the intelligent algorithms (GA, ACO, etc) are applied in each sub-problem. In this paper, the above representative studies are chosen to compare the performance with SSE. It should be noted that the ASS models of these chosen works are different so that the universality of SSE can be well indicated. In order to embody fairness, the test instances in this paper are obtained from the same way with these related works. Namely, the initial data in the comparison with mathematical programming are from Beasley's OR-Library [38,39], and the data in the comparison with the extended works of RHC are randomly generated.

The spread parameter θ and the length of the sliding window k in the local search module will affect the performance of SSE significantly. Therefore, the analyze tests mainly aim at these two parameters. Separately, the values of the two parameters have been tested for finding appropriate value range, as well as analyzing the performance influence in the SSE. In the comparison experiments (Section 4.1 and 4.2), we set $\theta = 0.5, k = 3$.

4.1 Comparison with mathematical programming

By using mixed-integer zero-one formulations, Beasley et al. have made great efforts to solve the ASS problem. The model has utilized Eq. (2) as the objective function and Eqs. (4) and (8) as the constraints [3], namely

$$
\begin{aligned}
c = & \sum_{i=1}^{N} (g_i \max\{0, PLT(i) - ALT(i)\} \\
& + h_i \max\{0, ALT(i) - PLT(i)\}) \\
\text{s.t.} \quad & ALT(i) - ALT(j) \geq \delta_{j,i} \\
& or \ ALT(j) - ALT(i) \geq \delta_{i,j}, \\
& \quad i, j \in \{1, \ldots, N\}, i \neq +j \\
& ALT(i) \in [E_i, L_i], \quad i \in \{1, \ldots, N\}.
\end{aligned}
$$

Adopting this classic formulation, Faye has proposed a new approach based on an approximation of the separation time matrix and on time discretization [53]. The data sets used in this work are chosen to compare the performance with SSE, and the results of time discretization approach are directly obtained in the original article [53]. We have collected the results of SSE based on 30 independent runs and marked them in bold in Table 2.

In Table 2, the results of the time discretization approach are obtained in CPLEX 12.6 on a quad-core processor Intel(R) Core(TM) i5-3470 CPU at 3.2 GHz with 4.0-GB RAM. The "LP value" column reports the value of the proposed LP relaxation and the "CPU Time" column gives time in seconds for computing the exact solution. The problem that cannot be solved in an hour (Case 7) is indicated by "-". For the results obtained by SSE, "Mean" and "Std" stand for

Table 2 Results compared with mathematical programming

Case	Number of Aircraft	Optimal Value	Time Discretization Approach		SSE		
			LP value	CPU Time (s)	Mean	Std	CPU Time (s)
1	10	700	550	1.1	**810.0**	**0**	**2.2**
2	15	1480	1110	5.1	**1630.0**	**0**	**4.5**
3	20	820	610	2.4	**820.0**	**0**	**7.4**
4	20	2520	2450	3.1	**2540.0**	**0**	**8.3**
5	20	3100	3030	4.9	**3620.0**	**0**	**7.7**
6	30	24442	5807	42.2	**24442.0**	**0**	**19.8**
7	44	1550	89	–	**1550.0**	**0**	**44.1**

the average results and standard deviations respectively, in which the population size is set as 100 and the number of iteration is set as 50.

The comparison shows that the LP relaxation of time discretization approach has proposed a lower bound. But the solution proposed by the LP values is invalid that the hard constraints are unsatisified. In practice, a feasible scheduling decision is much more important to ensure safety. Hence the results obtained by the time discretization approach are just for an optimization reference. On the contrary, the solution obtained by SSE is feasible and it is close to the optimal value, although there is no specially designed operators for this ASS model (objective (2) and constraints (4) and (8)). Meanwhile, at a rough estimate, the computing time can also be acceptable compared with the time discretization approach, since the performance of the experiment platform is different. A seemingly strange phenomenon is that the standard deviations of the results in all the cases are zero. It means that SSE always gets the same solution in all 30 runs. As mentioned in Sect. 3, SSE contains a local full permutation module. It can be observed that although SSE is a stochastic algorithm, the sequences obtained by the full permutation module are always the same in the above cases, namely local optimum solutions.

In this ASS model, the aircraft are allowed to land either before or after their PLTs. Under this circumstance, a landing sequence can be converted to more than one solution form, namely the ALT of each aircraft. We can see in Sect. 3.3(c) the solution is obtained by a somewhat greedy algorithm that the design of the evaluation module will also affect the quality of final solutions. For this reason, improving the evaluation module by modifying the rule to transform a sequence will make SSE get better solutions.

4.2 Comparison with RHC and its extended works

Proposed by Hu et al., the RHC algorithm divides the ASS problem into several sub-problems to solve the ASS problem dynamically [7]. Introducing the RHC framework, RHC-GA [5], RHC-BRGA [6] and RHC-ACS-ASS [20] are also proposed. In these works, Eq. (1) and Eqs. (5), (6) are used as the objective function and constraints of the ASS model respectively, namely

$$
\begin{aligned}
T_{delay} &= \sum_{i=1}^{N} [\max(ALT(i) - PLT(i), 0)] \\
\text{s.t.} \quad & ALT\left(\prod(i+1)\right) - ALT\left(\prod(i)\right) \geq \delta_{i,i+1}, \\
& \quad i \in \{2, \ldots, N\} \\
& ALT(i) \geq PLT(i), \quad i \in \{1, \ldots, N\}.
\end{aligned}
$$

In a specific scheduled landing sequence $\prod$, $\prod(i)$ denotes the order of the i^{th} flight in the original sequence and $\delta_{i,i+1}$ represents the LTI between the i^{th} and $i+1^{th}$ flight in the optimal sequence.

Different from the cases in Sect. 4.1, no infeasible permutation exists in this model that each permutation can be converted to a unique feasible solution form (the ALT for each aircraft). Twelve randomly generated cases are tested in the experiment, in which each aircraft is assigned with a random type and a random PLT. Then different combinations of aircraft amount and time range represent different busy degree situations of the TMA. The results obtained by implementing RHC and its extended works are collected and shown in Table 3. In this table, "Num" stands for the number of the aircraft in each case and "Time Span" represents the considered time span that these aircraft arrived at the TMA.

Table 3 Results compared with the extended works of RHC

Case	Num	Time Span (s)	RHC-GA		RHC-BRGA		RHC-ACS-ASS		SSE		
			Result	CPU Time (s)	Result	CPU Time (s)	Result	CPU Time (s)	Mean	Std	CPU Time (s)
1	20	1800	4383	12.4	4910	22.2	4731	10.5	**3584.7**	**80.0**	**6.4**
2	20	2400	5860	16.1	5876	29.5	5851	12.4	**2702.0**	**0**	**6.8**
3	20	3000	769	19.2	769	29.6	769	12.9	**769.0**	**0**	**6.6**
4	30	2400	30897	24.5	26631	37.1	28294	19.1	**22969.2**	**361.7**	**13.3**
5	30	3000	6380	20.8	6267	38.7	7017	19.1	**5190.4**	**91.5**	**14.3**
6	30	3600	5881	22.8	4908	42.6	5877	18.6	**3721.0**	**0**	**14.3**
7	40	2400	39963	27.2	37640	51.4	35948	24.3	**29769.4**	**327.2**	**31.3**
8	40	3000	19995	28.2	20483	48.2	20383	19.5	**15856.1**	**262.8**	**32.2**
9	40	3600	19292	31.4	23458	51.8	20030	22.0	**15150.5**	**267.5**	**34.5**
10	50	2400	89539	39.9	84446	68.1	80316	32.8	**62154.1**	**599.5**	**53.4**
11	50	3000	45000	37.7	42209	62.0	43297	29.0	**34394.1**	**478.3**	**54.3**
12	50	3600	42737	37.6	42707	67.0	37210	30.6	**26865.3**	**367.4**	**57.4**

The results of SSE are also obtained from 30 independent runs and marked in bold in Table 3. In order to embody fairness, the population size and the number of iteration are all set as 100 and 50 in SSE and RHC-GA, RHC-BRGA and RHC-ACS-ASS. Note that, SSE performed much better compared with the other algorithms. Especially in non-busy cases (Case 2, 3 and 6), SSE can always obtain the best solutions in different runs. It is because the extended algorithms of RHC have divided the original problem into several subproblems that only part of the information can be utilized during the solving process. On the contrary, SSE is a global search algorithm that it considers all aircraft at the same time. This characteristic makes SSE be able to obtain better solutions than the other three algorithms. However, it also makes SSE a bit more time consuming compared with RHC-GA and RHC-ACS-ASS, but its computing time is still acceptable from the view of applications.

4.3 Test of the parameters

As mentioned earlier, the spread parameter θ and the length of the sliding window k in the local search module will be tested in this set of experiments. To investigate the influence caused by these two parameters separately, we change the value of one parameter while keeping another unchanged. For instance, the parameter θ is tested under the condition of $k = 3$. Likewise, k is also tested while $\theta = 0.5$.

In our algorithm framework, the spread parameter θ is definitely the key to control the search performance. As the value of θ increases, the probability tends to concentrate on the central permutation σ_0. In order to find an appropriate value, two different problem cases have been chosen and a series of values from 0.1 to 1 have been tested to analyze those behaviors, as shown in Figs. 4 and 5. The results show that the performance can be affected by both the value of θ and the scale of the problem. Besides, the computing time will also decrease when the value of θ increases. According to the scale of most ASS problems ($20 \leq N \leq 50$), the best solutions are often obtained in a fixed value range of [0.1, 0.5].

Another important parameter is the length of sliding window in the local search module, namely the value of k. If the value is too large, the efficiency of the algorithm will be significantly affected, since the number of full permutation will be $k!$. However, if the length of the sliding window is too short, the quality of the final solution will decrease. A test shows the results in Figs. 6 and 7 that the value of $\{2, 3, 4\}$ is more suitable to obtain a satisfactory solution.

In order to investigate the coupling effects between these two parameters, a statistical analysis is also made in which the two parameters are changing simultaneously. The value of θ is set to fluctuate in [0.1, 0.5] and the value of k is set to fluctuate in $\{2, 3, 4\}$. According to the above analysis, this setting of the value range is more helpful to obtain the best solution. The statistical results are given in Tables 4 and 5.The results show that the most suitable combinations of the parameter values are different in different problems. It is actually associated with the problem instances. However, it should be noted that a larger value of θ and a lower value of k will significantly reduce the computing time. Since efficiency is another important criterion in air traffic control, the parameter setting should take the application value into account. According to the statistical results, setting θ as 0.5 and setting k as 3 will make a great compromise between efficacy and efficiency.

In addition, population size and number of iterations also have influence on the algorithm performance. However, the numerical value can be adjusted according to the scale of the

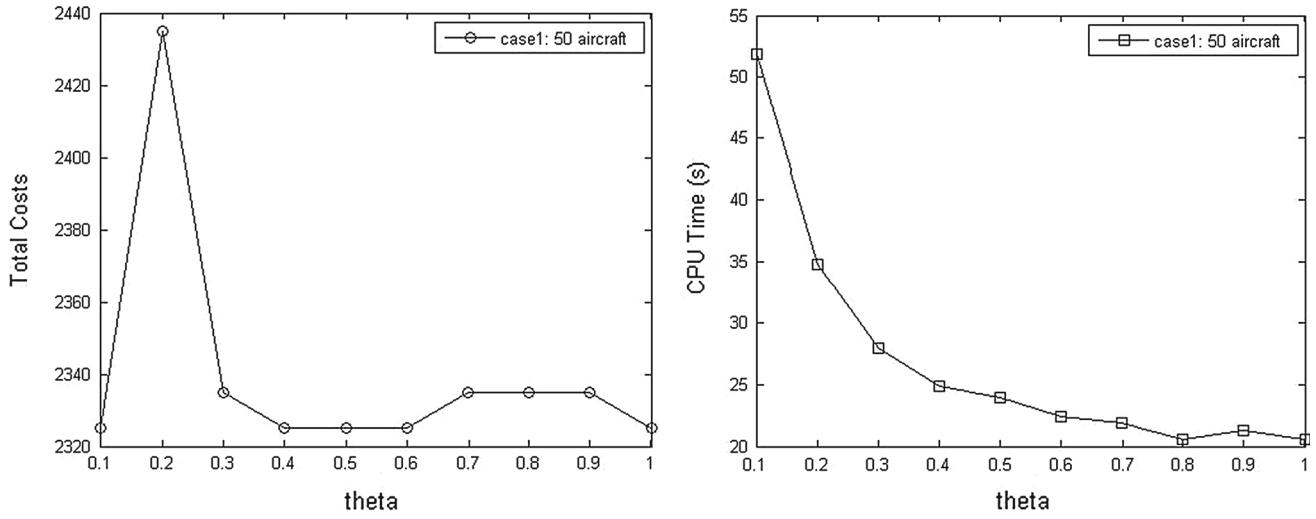

Fig. 4 Influence of the parameter θ in Problem 1 (Mean Total Costs and Mean CPU time)

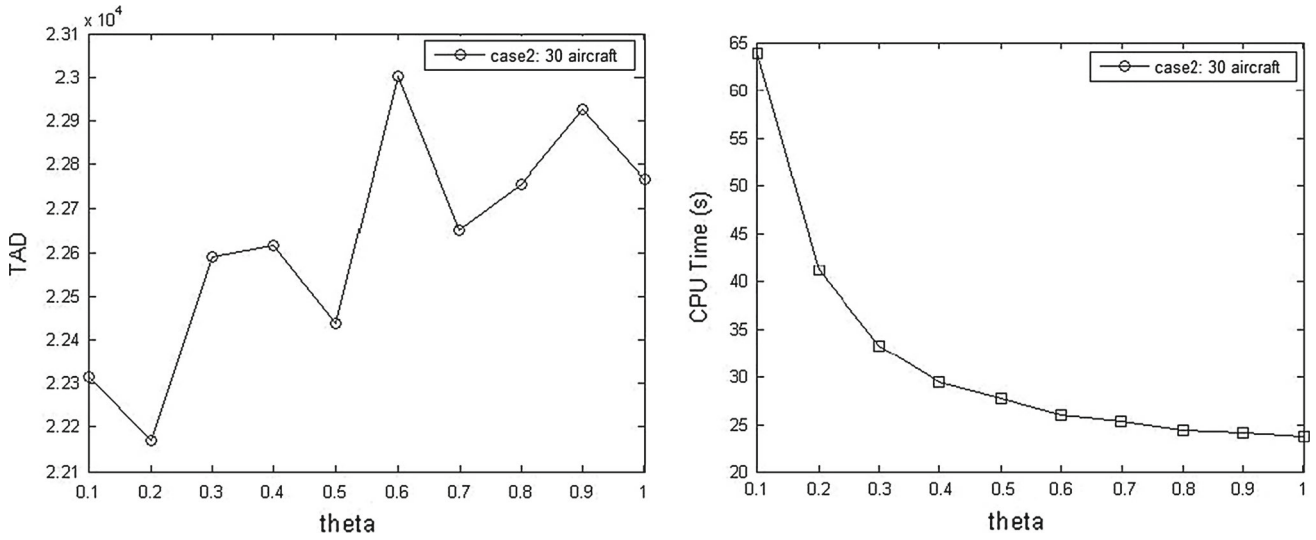

Fig. 5 Influence of the parameter θ in Problem 2 (Mean Total Airborne Delay and Mean CPU time)

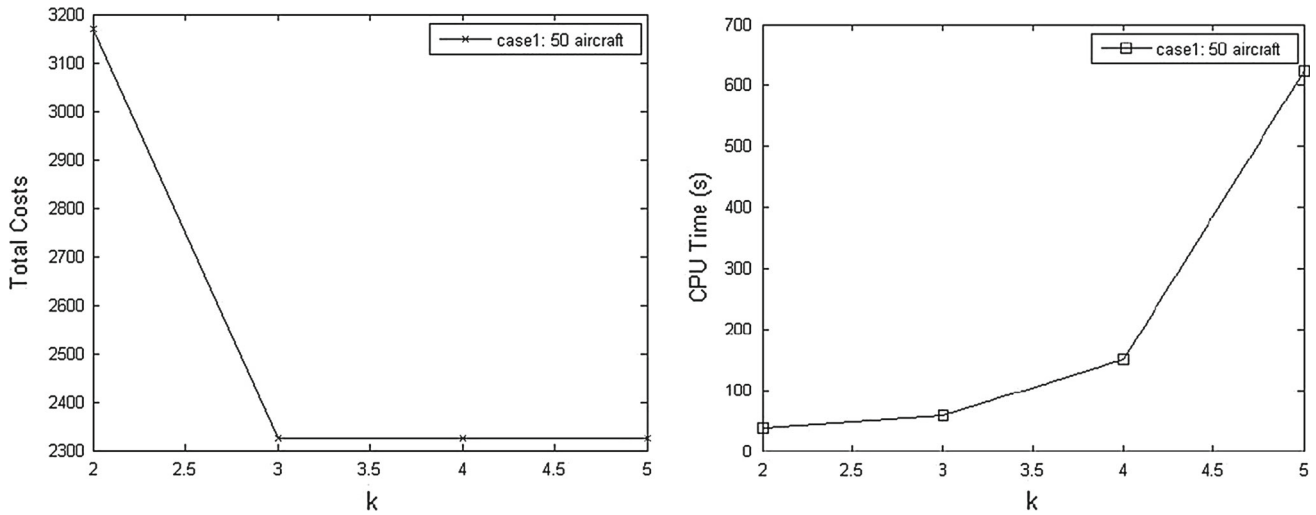

Fig. 6 Influence of the parameter k in Problem 1 (Mean Total Costs and Mean CPU time)

problem. When the scale is larger, appropriately increasing the value of these two parameters will make the algorithm obtain higher quality solutions.

4.4 Time complexity analysis

The time complexity of SSE is mainly related to the number of aircraft and the population size. Suppose the number of aircraft is N and the population size of SSE is M. The time complexity of getting central permutation, sampling the probability model, evaluating the fitness of the individuals, sorting the individuals according to their fitnesses and local search are $O(M^*N^2)$, $O(M^*N^2)$, $O(M^*N)$, $O(M^*N^2)$ and $O(M^*N)$ respectively.

Note that sampling the probability model seems to be the most time consuming part of SSE since it needs more operations. It is because that for a permutation with a length of$N$, a permutation with the maximum distance of $N^*(N-1)/2$ can be found so that the probability model has $N^*(N-1)/2+1$ intervals. After determining a distance d ($d \in [0, N^*(N-1)/2]$) by the probability model, the SSE should generate a permutation with a distance of d with the central permutation. It will cost long time to generate a new permutation if the value of d is large. In fact, according to Section 3.2, it can be seen that the probability model is analogous to the Gaussian distribution on the space of permutations. When the value of spread parameter θ increases, the curve of the probability distribution becomes more peaked at the central permutation σ_0. Therefore, the distance of d can hardly determined to a large value with an appropriate value of θ. Therefore, it makes SSE not so much time consuming when solving the permutation-based problems.

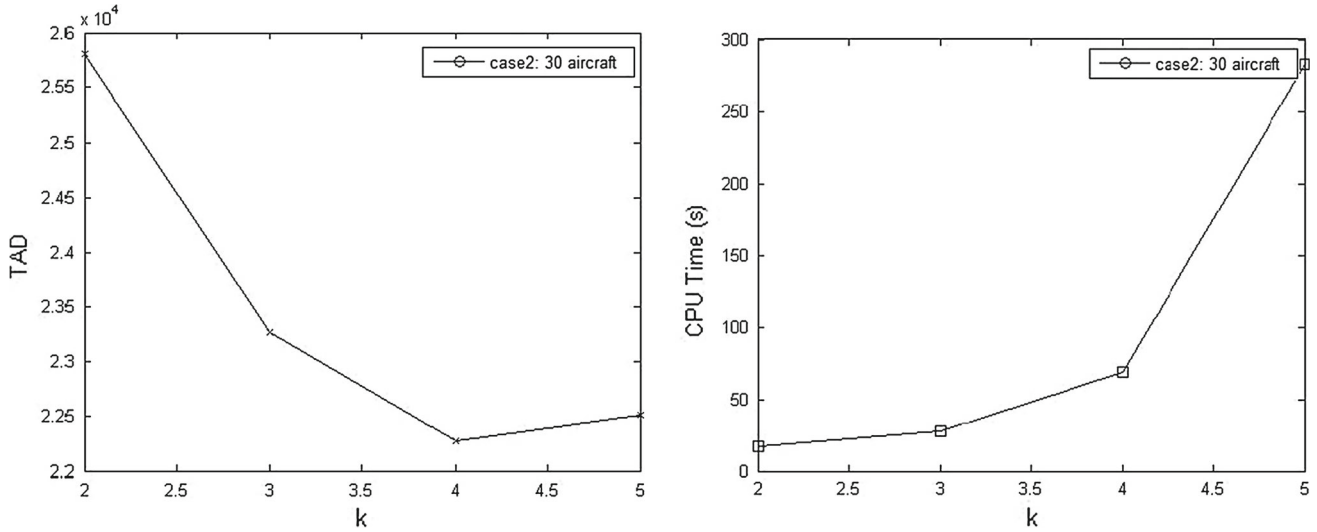

Fig. 7 Influence of the parameter k in Problem 2 (Mean Total Airborne Delay and Mean CPU time)

Table 4 Performance influence of the parameters in Problem 1

Average value / CPU time	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$
$k = 2$	2325.0/ 431.17	2430.0/ 126.74	2430.0/ 104.26s	2430.0/ 90.11s	2430.0/ 77.83s
$k = 3$	2325.0/ 474.86s	2325.0/ 224.37s	2360.0/ 131.94s	2395.0/ 104.88s	2430.0/ 98.65s
$k = 4$	2325.0/ 650.80	2325.0/ 432.74s	2325.0/ 234.78s	2325.0/ 212.04s	2325.0/ 197.32s

Table 5 Performance influence of the parameters in Problem 2

Average value / CPU time	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$
$k = 2$	26917.6/ 233.44s	26213.3/ 107.91s	27005.5/ 64.77s	27885.3/ 48.92s	27519.7/ 36.66s
$k = 3$	26317.0/ 239.40s	26262.0/ 117.65s	26500.0/ 66.96s	26729.0/ 52.01s	28391.7/ 40.67s
$k = 4$	26971.6/ 255.89s	26180.7/ 118.22s	27247.0/ 73.35s	28009.7/ 56.40s	26771.0/ 43.66s

4.5 Summary

It can be seen from the above results that SSE is capable to solve various kinds of ASS problems. Furthermore, its average performance is great in solving different models. It seems that the solution speed of SSE seems lower than a few existing methods. In fact, the computation time may be influenced by various factors besides the programming software, such as the computers, operating systems, coding skills of the programmers, etc. Ignoring the impact of these factors, SSE provides a common method for solving different kinds of ASS problems in an acceptable period of time. In addition, this algorithm can always get a better solution in different models so that it shows great universality.

5 Conclusion

In this paper, different formulations for ASS problems were classifed by the objectives and constraints. Essentially, these models can all be formulated as permutation-based problems. According to this characteristic, a unified algorithm framework named SSE was developed. The SSE was designed based on a Mallows Model EDA and a local improvement operator. It offered a unified algorithm framework to solve different ASS problems and facilitates decision making of air traffic controllers in different situations. The simulation results indicated that the SSE algorithm has great universality and satisfactory performance compared with exsting algorithms.

Further research work includes: (1) the acceleration of the algorithm framework; (2) proposing new operating rules of the evaluation module to get better solutions in certain cases; (3) improving the algorithm to adapt to dynamic cases with uncertainties and disturbances; (4) the extension of the algorithm to be suitable in multi-runway systems.

Acknowledgments This paper is supported by the National Basic Research Program of China (Grant No.2011CB707000), the National Science Fund for Distinguished Young Scholars (Grant No. 61425014) and the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 61221061).

Compliance with ethical standards

Conflict of interest The authors declare no conflict of interest.

References

1. Beasley JE, Krishnamoorthy M, Sharaiha YM et al (2004) Displacement problem and dynamically scheduling aircraft landings. J Operat Res Soc 55(1):54–64
2. Beasley J, Sonander J, Havelock P (2001) Scheduling aircraft landings at London Heathrow using a population heuristic. J Opera Res Soc 52(5):483–493
3. Beasley JE, Krishnamoorthy M, Sharaiha YM et al (2000) Scheduling aircraft landings–the static case. Trans Sci 34(2):180–197
4. Hu XB, Paolo E (2011) A ripple-spreading genetic algorithm for the aircraft sequencing problem. Evolu Comp 19(1):77–106
5. Hu XB, Di Paolo E (2008) Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. Intell Trans Syst IEEE Trans 9(2):301–310
6. Hu XB, Chen WH (2005) Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. Eng Appl Art Intell 18(5):633–642
7. Hu XB, Chen WH (2005) Receding horizon control for aircraft arrival sequencing and scheduling. Intell Trans Syst IEEE Trans 6(2):189–197
8. Zheng L, Zhang J, Zhu Y B (2009) Affinity propagation clustering classification method for aircraft in arrival and departure sequencing. In: Digital Avionics Systems Conference. DASC'09. IEEE/AIAA 28th. IEEE, 2009: 7. A. 4-1-7. A. 4-8
9. Zhang X, Zhang X, Zhang J et al (2007) Optimization of sequencing for aircraft arrival based on approach routes. In: Intelligent Transportation Systems Conference. ITSC 2007. IEEE. IEEE 2007:592–596
10. Yu SP, Cao XB, Zhang J (2011) A real-time schedule method for Aircraft Landing Scheduling problem based on Cellular Automation. Appl Soft Comp 11(4):3485–3493
11. Guo YP, Cao XB, Zhang J (2009) Constraint handling based multiobjective evolutionary algorithm for aircraft landing scheduling. Int J Innov Comp Inform Cont 5(8):2229–2238
12. Tang K, Wang Z, Cao X et al (2008) A multi-objective evolutionary approach to aircraft landing scheduling problems. In: Evolutionary Computation. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on. IEEE 2008:3650–3656
13. Guo YP, Cao XB, Zhang J (2008) Multiobjective evolutionary algorithm with constraint handling for aircraft landing scheduling. In: Evolutionary Computation. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on. IEEE 2008:3657–3662
14. Wang L J, Hu D W, Gong R Z (2009) Improved genetic algorithm for aircraft departure sequencing problem. In: Genetic and Evolutionary Computing. WGEC'09. 3rd International Conference on. IEEE, 2009: 35-38
15. Carr GC, Erzberger H, Neuman F (1999) Delay exchanges in arrival sequencing and scheduling. J Aircraft 36(5):785–791
16. Carr GC, Erzberger H, Neuman F (2000) Fast-time study of airline-influenced arrival sequencing and scheduling. J Guid Cont Dyn 23(3):526–531
17. Bianco L, Bielli M (1993) System aspects and optimization models in ATC planning. In: Large Scale Computation and Information Processing in Air Traffic Control. Springer, Berlin Heidelberg 1993:47–99
18. Bianco L, Dell'Olmo P, Giordani S (1997) Scheduling models and algorithms for TMA traffic management. In: Modelling and simulation in air traffic management. Springer, Berlin Heidelberg 1997:139–167
19. Bianco L, Ricciardelli S, Rinaldi G et al (1988) Scheduling tasks with sequencec-dependent processing times. Naval Res Log (NRL) 35(2):177–184
20. Zhan ZH, Zhang J, Li Y et al (2010) An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem. Intell Trans Syst IEEE Trans 11(2):399–412
21. Sun S G, Hua K Q (2009) An Aircraft Sequencing Approach Based on Fuzzy Petri-net. In: Computational Sciences and Optimization. CSO 2009. Int Joint Conf. IEEE 1:1008–1011
22. Wang F, Xu XH, Zang J (2008) Strategy for aircraft sequencing based on artificial fish school algorithm. In: Control and Decision Conference. CCDC 2008. Chinese. IEEE 2008:861–864
23. Wang SL (2009) Solving Aircraft-Sequencing Problem Based on Bee Evolutionary Genetic Algorithm and Clustering Method. In: Dependable, Autonomic and Secure Computing. DASC'09. Eighth IEEE International Conference on. IEEE 2009:157–161
24. Soomer MJ, Franx GJ (2008) Scheduling aircraft landings using airlines' preferences. Euro J Oper Res 190(1):277–291
25. Ding YY, Valasek J (2007) Aircraft landing scheduling optimization for single runway non-controlled airports: Static case. J Guid Cont Dyn 30(1):252–255
26. Harikiopoulo D, Neogi N (2011) Polynomial-time feasibility condition for multiclass aircraft sequencing on a single-runway airport. Intell Trans Syst IEEE Trans 12(1):2–14
27. Bojanowski L, Harikiopoulo D, Neogi N (2011) Multi-runway aircraft sequencing at congested airports. In: American Control Conference (ACC). IEEE 2011:2752–2758
28. Eun Y, Hwang I, Bang H (2010) Optimal arrival flight sequencing and scheduling using discrete airborne delays. Intell Trans Syst IEEE Trans 11(2):359–373
29. Malaek SMB, Naderi E (2008) A new scheduling strategy for aircraft landings under dynamic position shifting. In: Aerospace Conference, IEEE 2008:1–8
30. Chen S, Xia XZ (2009) Researches on Optimal Scheduling Model for Aircraft Landing Problem. In: Information Engineering, 2009. ICIE'09. WASE International Conference on. IEEE 1:418–421
31. Jia X L, Cao X B, Guo Y P et al (2008). Scheduling aircraft landing based on clonal selection algorithm and receding horizon control. In: Intelligent Transportation Systems. ITSC 2008. 11th International IEEE Conference on. IEEE, 2008: 357-362
32. Ciesielski V, Scerri P (1998). Real time genetic scheduling of aircraft landing times. In: Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, 1998:360–364
33. Ceberio J, Mendiburu A, Lozano JA (2011) Introducing the mallows model on estimation of distribution algorithms. In: Neural Information Processing. Springer, Berlin Heidelberg 2011:461–470

34. Bengoetxea E, Larrañaga P, Bloch I et al (2002) Inexact graph matching by means of estimation of distribution algorithms. Pattern Recog 35(12):2867–2880
35. (2002) In: Larrañaga P, Lozano JA (eds.) Estimation of distribution algorithms: a new tool for evolutionary computation. Springer
36. Robles V, de Miguel P, Larranaga P (2002) Solving the traveling salesman problem with EDAs. In: Estimation of Distribution Algorithms. Springer, US 2002:211–229
37. Ali A, Meilă M (2012) Experiments with Kemeny ranking: What works when? Math Soc Sci 64(1):28–40
38. Beasley JE (1990) OR-Library: distributing test problems by electronic mail. J Opera Res Soci 1990:1069–1072
39. Beasley JE (1996) Obtaining test problems via internet. J Global Optim 8(4):429–433
40. Balakrishnan H, Chandran B (2006). Scheduling aircraft landings under constrained position shifting. AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO
41. Ceberio J, Irurozki E, Mendiburu A et al (2014) A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. Evolu Comp IEEE Trans 18(2):286–300
42. Ceberio J, Irurozki E, Mendiburu A et al (2012) A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Prog Art Intell 1(1):103–117
43. Tavakkoli-Moghaddam R, Yaghoubi-Panah M, Radmehr F (2012) Scheduling the sequence of aircraft landings for a single runway using a fuzzy programming approach. J Air Trans Manag 25:15–18
44. Salehipour A, Modarres M, Naeni LM (2013) An efficient hybrid meta-heuristic for aircraft landing problem. Comp Oper Res 40(1):207–213
45. Hancerliogullari G, Rabadi G, Al-Salem AH et al (2013) Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem. J Air Trans Manag 32:39–48
46. Xie J, Zhou Y, Zheng H (2013) A hybrid metaheuristic for multiple runways aircraft landing problem based on bat algorithm. J Appl Math
47. Awasthi A, Kramer O, Lassig J (2013) Aircraft landing problem: An efficient algorithm for a given landing sequence. In: Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on. IEEE, 2013:20–27
48. Vadlamani S, Hosseini S (2014) A novel heuristic approach for solving aircraft landing problem with single runway. J Air Trans Manag 40:144–148
49. Sölveling G, Clarke JP (2014) Scheduling of airport runway operations using stochastic branch and bound methods. Trans Res Part C Emerg Technol 45:119–137
50. Ma W, Xu B, Liu M, et al (2014) An efficient approximation algorithm for aircraft arrival sequencing and scheduling problem. Math Prob Eng
51. Briskorn D, Stolletz R (2014) Aircraft landing problems with aircraft classes. J Sched 17(1):31–45
52. Ghoniem A, Sherali HD, Baik H (2014) Enhanced models for a mixed arrival-departure aircraft sequencing problem. INFORMS J Comp 26(3):514–530
53. Faye A (2015) Solving the Aircraft Landing Problem with time discretization approach. Euro J Oper Res 242(3):1028–1038
54. Lieder A, Briskorn D, Stolletz R (2014) A dynamic programming approach for the aircraft landing problem with aircraft classes. Euro J Oper Res
55. Psaraftis HN (1978) A dynamic programming approach to the aircraft sequencing problem. In: Cambridge, Mass.: Massachusetts Institute of Technology, Flight Transportation Laboratory
56. Psaraftis HN (1980) A dynamic programming approach for sequencing groups of identical jobs. Oper Res 28(6):1347–1359
57. Dear RG (1976) The dynamic scheduling of aircraft in the near terminal area. Cambridge, Mass.: Massachusetts Institute of Technology, Flight Transportation Laboratory
58. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report, 826
59. Hasan SMK, Sarker R, Essam D et al (2009) Memetic algorithms for solving job-shop scheduling problems. Memetic Comp 1(1):69–83
60. Kobti Z (2012) A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic. Memetic Comp 4(3):231–245
61. Chaimatanan S, Delahaye D, Mongeau M (2014) A Hybrid Metaheuristic Optimization Algorithm for Strategic Planning of 4D Aircraft Trajectories at the Continental Scale. Comp Intell Mag IEEE 9(4):46–61
62. Liang X, Chen H, Lozano J (2012) A Boltzmann-based Estimation of Distribution Algorithm for a General Resource Scheduling Model