Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling

Xiao-Bing Hu and Wen-Hua Chen

Department of Aeronautical and Automotive Engineering

Loughborough University

Loughborough

Leicestershire LE11 3TU

UK

Tel.: +44 1509 227230

Fax: +44 1509227275

Email: X.Hu@lboro.ac.uk, W.Chen@lboro.ac.uk

Abstract:

The concept of Receding Horizon Control (RHC) is introduced into Genetic Algorithm (GA) in this paper to solve the problem of Arrival Scheduling and Sequencing (ASS) at a busy hub airport. A GA based method is proposed for solving the dynamic ASS problem, and the focus is put on the methodology of integrating the RHC strategy into the GA for real-time implementations in a dynamic environment of air traffic control (ATC). Receding horizon and terminal penalty are investigated in depth as two key techniques of this novel RHC based GA. Simulation results show that the new method proposed in this paper is effective and efficient to solve the ASS problem in a dynamic environment.

Key words: Receding Horizon Control, Genetic Algorithm, Air Traffic Control, Arrival Scheduling and Sequencing, Terminal Penalty.

1. Introduction

Arrival sequencing and scheduling (ASS) is one of the standard problems in Air Traffic Control (ATC). Simply speaking, ASS is the function of generating efficient landing sequences and landing times for arrivals at the airport so that the safety separation between arrival aircraft is guaranteed, the available capacity at the airport is efficiently used and airborne delays are significantly reduced. [1] A simple way to perform ASS is to schedule arrival aircraft in a first-come-first-served (FCFS) order based on a predicted landing time (PLT) at the runway. Although FCFS scheduling establishes a fair order based on predicted landing time, it ignores other useful information which can be used to efficiently make use of the capacity of the airport, reduce airborne delays and/or improve the service to airlines. For example, "delay exchange" is introduced into the ASS problem in [2], and in [3], individual airline priorities among in-coming flights are taken into account and the concept of "priority scheduling" is then defined.

The most widely accepted concept in the practices of ASS is "position shifting" [4]-[11], which is based on two facts: First, safety regulations state that any two co-altitudinal aircraft must maintain a "minimum horizontal separation", which is a function of the types and of the relative positions of the two aircraft; Second, the "landing speed" of a type of aircraft is generally different from that of another type of aircraft. As a consequence of the variability of the above parameters, the Landing Time Interval (LTI), which is the minimum permissible time interval between two successive landings, is a variable quantity. These differences in separation are mandatory and recognized by ATC regulations. For the sake of simplicity, like [10], aircraft waiting to land are classified into a relatively small number of distinct categories, according to speed, capacity, weight and other technical characteristics. Table 1 shows the minimum LTIs relative to main categories of commercial aircraft. In particular, 4 categories are considered: category number 1 identifies Boeing 747 (B747), category number 2 corresponds to Boeing 727 (B727), category number 3 identifies Boeing 707 (B707) and finally category number 4 corresponds to Mc Donnel Douglas DC9 (DC9).

Table 1. Minimum landing time intervals (LTI) [10]

S_{ij} (seconds)		Category of following aircraft: *j*			
		1	2	3	4
Category of leading aircraft: *I*	1	96	200	181	228
	2	72	80	70	110
	3	72	100	70	130
	4	72	80	70	90

Note: 1=B747; 2=B727; 3=B707; 4=DC9.

It is evident that the LTIs in Table 1 are asymmetric. For example, a minimum LTI of 200 seconds is required for a B727 to follow a B747, while a minimum LTI of only 72 seconds needs to be satisfied for the same pair of aircraft in reverse order. By taking advantage of the asymmetries of the LTIs, in other words, by shifting positions of aircraft in an FCFS landing sequence, it is possible to reduce delays and to improve the capacity of the airport. This paper focuses on the problem of “position shifting” based ASS.

Many efforts have been made to study the problem of “position shifting” based ASS in the past decades, for example, the ASS problem is modeled as a Traveling Salesman Problem (TSP) and dynamic programming algorithms are developed, e.g., see [7], [8], [10] and [11]. The above papers mainly focus on developing effective static algorithms. Although the dynamic case is also discussed in [7], [8] and [10], the proposed dynamic algorithms are simple extensions of the associated main static results by introducing some constraints. In the real world of ATC, ASS is always carried out in a dynamic environment. Therefore, modeling and developing algorithms directly based on the dynamic feature of the ASS problem could bring advantages.

In this paper, we propose a Receding Horizon Control (RHC) based Genetic Algorithm (GA) for solving the position-shifting based ASS problem in a dynamic ATC environment. As is well known, GA is a large-scale parallel stochastic searching and optimizing algorithm, and it is effective for solving NP-complete problems such as the ASS problem ([17] and [18]). The conventional way to apply GA to the dynamic case of the ASS problem is to simply use a GA to repeat optimizing the entire arrival flow during the rest of the operating day. Since a huge number of aircraft arrive at a busy airport every day, the GA used conventionally can hardly meet the need of real-time properties in practice. On the other hand, in a dynamic ATC environment, there is inevitably unreliable information in the predicted arrival flow. For example, some flights may be canceled while some unanticipated aircraft may ask for emergency landing. The conventional way of taking all aircraft into account can not necessarily result in actually optimal or even sub-optimal solutions. To overcome these drawbacks of a conventional GA, we introduce the concept of RHC to GA for the ASS problem. Simply speaking, RHC is an *N*-step-ahead online optimization strategy. At each time interval, based on current available information, RHC optimizes the particular problem for the next *N* intervals in the near future, but only the part of solution corresponding to current interval is implemented. At the next interval, RHC repeats the similar optimizing procedure for another *N* intervals in the near further based on updated information. Clearly, since not all arrival aircraft are considered in the optimization process of each time interval, the RHC strategy can effectively improve the real-time properties of GA and reduce the influence of unreliable information. However, the introduction of RHC could also make the new GA short-sighted. To guarantee the solution quality of RHC based GA, some techniques which are widely used by RHC algorithms in control engineering are applied in this paper. One of them is the terminal penalty technique.

The remainder of this paper is organized as follows. The basic idea of RHC is briefly explained in Section 2, and then the novel RHC based GA for the ASS problem is proposed in Section 3. Section 4 reports some interesting simulation results. The paper ends with some conclusions in Section 5.

2. The basic idea of RHC

Receding horizon control has proved to be a very effective online optimization strategy in the area of control engineering, and is very successful when compared with other control strategies.[12] It is easy for RHC to handle complex dynamic systems with various constraints. It also naturally exhibits promising robust performance against uncertainties since the online updated information can be sufficiently used to improve the decision. Within this framework, decisions are made by looking ahead for N steps in terms of a given cost/criterion, and only the decision for the first step is actually implemented. Then the implementation result is checked, and a new decision is made by taking into account of updated information and looking ahead for another N steps. RHC has now been widely accepted in the area of control engineering. Recently, attention has been paid to applications of RHC to areas like management and operations research. For example, theoretical research work on how to apply MPC (model predictive control, another name for RHC) to a certain class of discrete-event systems was presented in [13] and [14], and many practical implementations of RHC in the area of commercial planning and marketing were reported in [15]. However, as mentioned in [16], the research work on applying RHC to areas other than control engineering is just beginning.

The basic idea of RHC for dynamical optimization problems is illustrated by the flow chart given in Figure 1. Figure 2 compares the RHC strategy with some other conventional optimization strategies in an intuitive way. It is evident that the offline optimization strategy, as shown in Figure 2.(a), is not suitable for dynamic optimization processes. However, most algorithms in literature on the ASS problem are mainly tested by using offline strategy, so-called static version, e.g., see [7], [8] and [10]. The one-step-ahead adjustment strategy in Figure 2.(b) is often used in the real practice of ASS, due to its simplicity. One-step-ahead adjustment can be considered a special case of RHC, i.e., the length of receding horizon is N=1. This special RHC is always criticized for being short-sighted. Those dynamic versions of algorithms for the ASS problem in literature follows the conventional dynamic optimization strategy shown in Figure 2.(c). This strategy often suffers from heavy online computational burden, and its performance is relatively too sensitive to disturbances and/or uncertainties included in the current information. Figure 2.(d) illustrates that the RHC strategy is the best natural trade-off.

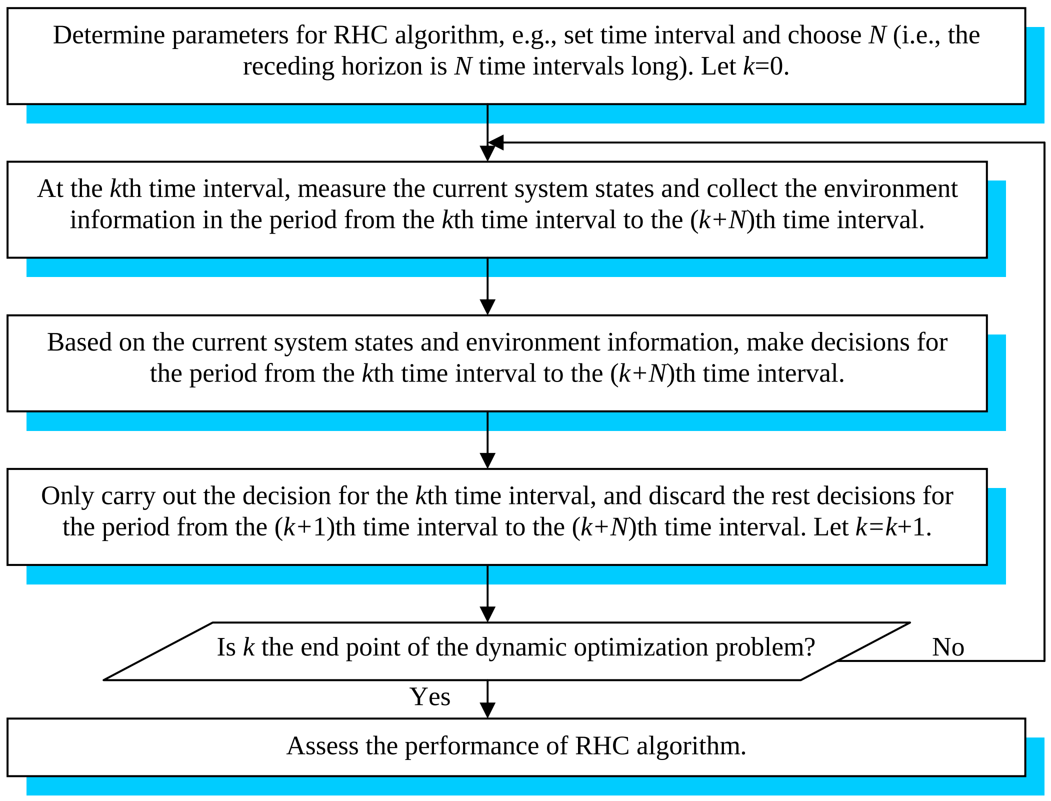
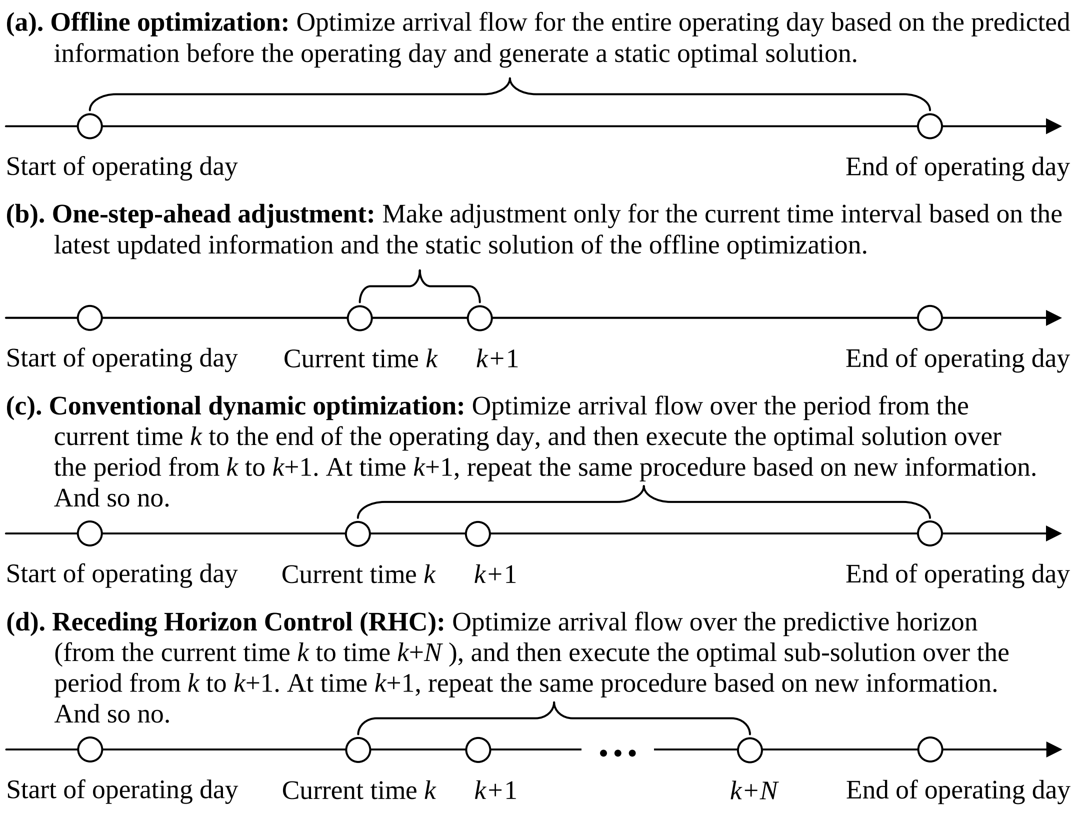

Determine parameters for RHC algorithm, e.g., set time interval and choose *N* (i.e., the receding horizon is *N* time intervals long). Let *k*=0.
At the *k*th time interval, measure the current system states and collect the environment information in the period from the *k*th time interval to the (*k*+*N*)th time interval.
Based on the current system states and environment information, make decisions for the period from the *k*th time interval to the (*k*+*N*)th time interval.
Only carry out the decision for the *k*th time interval, and discard the rest decisions for the period from the (*k*+1)th time interval to the (*k*+*N*)th time interval. Let *k*=*k*+1.
Is *k* the end point of the dynamic optimization problem?
No
Yes
Assess the performance of RHC algorithm.

Figure 1. The flow chart of RHC



(a). Offline optimization: Optimize arrival flow for the entire operating day based on the predicted information before the operating day and generate a static optimal solution.

(b). One-step-ahead adjustment: Make adjustment only for the current time interval based on the latest updated information and the static solution of the offline optimization.

(c). Conventional dynamic optimization: Optimize arrival flow over the period from the current time *k* to the end of the operating day, and then execute the optimal solution over the period from *k* to *k*+1. At time *k*+1, repeat the same procedure based on new information. And so no.

(d). Receding Horizon Control (RHC): Optimize arrival flow over the predictive horizon (from the current time *k* to time *k*+*N*), and then execute the optimal sub-solution over the period from *k* to *k*+1. At time *k*+1, repeat the same procedure based on new information. And so no.

Figure 2. Some optimization strategies

However, how to integrate the RHC strategy into GA to develop an effective and practicable method to solve the ASS problem in a dynamic ATC environment requires much more than simply to use whatever kind of GA as the online optimizer in the RHC. To make them work in harmony, in the first place, the GA based online optimizer should be designed from a dynamic point of view, more precisely speaking, from an RHC point of view. Therefore, unlike other literature, we do not have a so-called static version of algorithms in this paper, but directly present our RHC based GA for solving the ASS problem in a dynamic ATC environment. As will be explained in depth later, some techniques, particularly terminal penalty, used by the RHC in control engineering are adopted to design the online GA based optimizer.

3. RHC based GA for the ASS problem

3.1 Formulation of the ASS problem

A number of aircraft are supposed to land at the same runway of an airport during an operating day. Assume the number of the aircraft of concern is N_{AC}, and the operating day is T_{range}-minute long. N_{AC} and T_{range} can be used to estimate the degree of congestion at the runway of the airport. For the *i*th aircraft $AC(i)$ in the original sequence, $i = 1, \dots, N_{AC}$, there is a Predicted Landing Time (PLT) at the runway, denoted as $t_{\mathrm{PLT}}(i)$. Based on this set of PLTs, i.e., $t_{\mathrm{PLT}}(i)$, $i = 1, \dots, N_{AC}$, an FCFS landing sequence can be directly worked out with respect to safety regulations. As mentioned in Section 1, the safety separation, i.e., minimum LTI, between a pair of successive aircraft, is a function of the type and of the relative positions of the two aircraft. The asymmetries of the LTIs in Table 1 imply that, by shifting positions of aircraft in an FCFS landing sequence, it is possible to reduce delays and to improve the capacity of the airport. The potential benefits resulting from position shifting, considering airborne delay, are illustrated by Figure 3, where $t_{\mathrm{ALT}}(i)$ is the Allocated Landing Time (ALT) given to the *i*th aircraft in the original landing sequence by an optimization process. T_{delay} and T_{length} are the total airborne delay and the length of an optimized landing sequence, respectively, and they are calculated as

$$T_{delay} = \sum_{i=1}^{N_{AC}} (t_{ALT}(i) - t_{PLT}(i)) . \tag{1}$$

$$T_{length} = t_{ALT}(N_{AC}) - t_{ALT}(1) , \tag{2}$$

The goal of ASS is usually to minimize T_{delay}. T_{length} is sometimes also adopted as the index for optimization. The index T_{delay} emphasizes the operating cost of airlines, while the index T_{length} focuses on the capacity of the airport. In many cases, a minimum T_{delay} occurs simultaneously along with a minimum T_{length}. However, this does not mean they are equivalent to each other. Due to the space limit, proposed RHC based GA proposed in this paper will be described only based on the index T_{delay}.

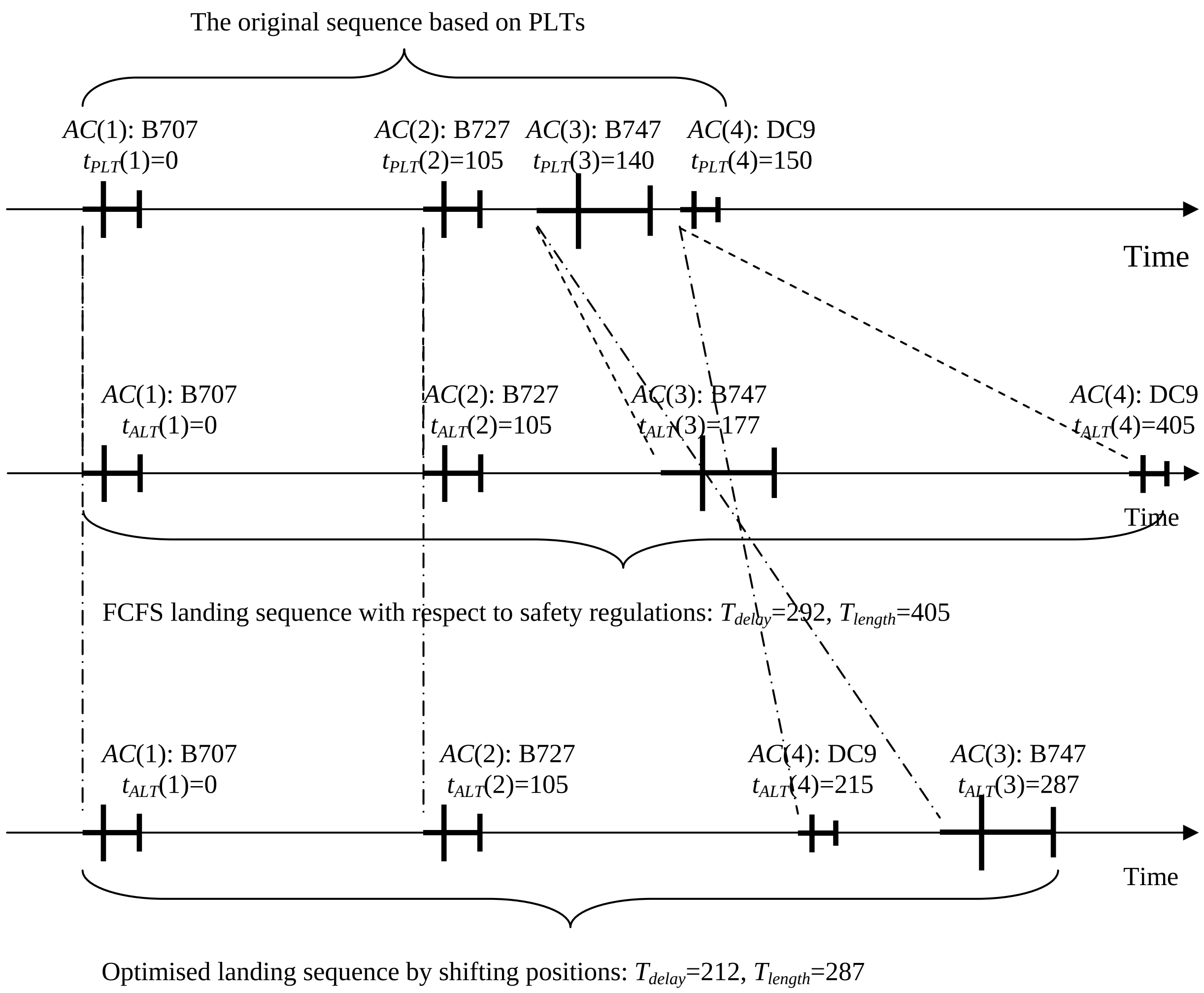

The original sequence based on PLTs
AC(1): B707 $t_{PLT}(1)=0$
AC(2): B727 $t_{PLT}(2)=105$
AC(3): B747 $t_{PLT}(3)=140$
AC(4): DC9 $t_{PLT}(4)=150$
Time
AC(1): B707 $t_{ALT}(1)=0$
AC(2): B727 $t_{ALT}(2)=105$
AC(3): B747 $t_{ALT}(3)=177$
AC(4): DC9 $t_{ALT}(4)=405$
Time
FCFS landing sequence with respect to safety regulations: $T_{delay}=292$, $T_{length}=405$
AC(1): B707 $t_{ALT}(1)=0$
AC(2): B727 $t_{ALT}(2)=105$
AC(4): DC9 $t_{ALT}(4)=215$
AC(3): B747 $t_{ALT}(3)=287$
Time
Optimised landing sequence by shifting positions: $T_{delay}=212$, $T_{length}=287$

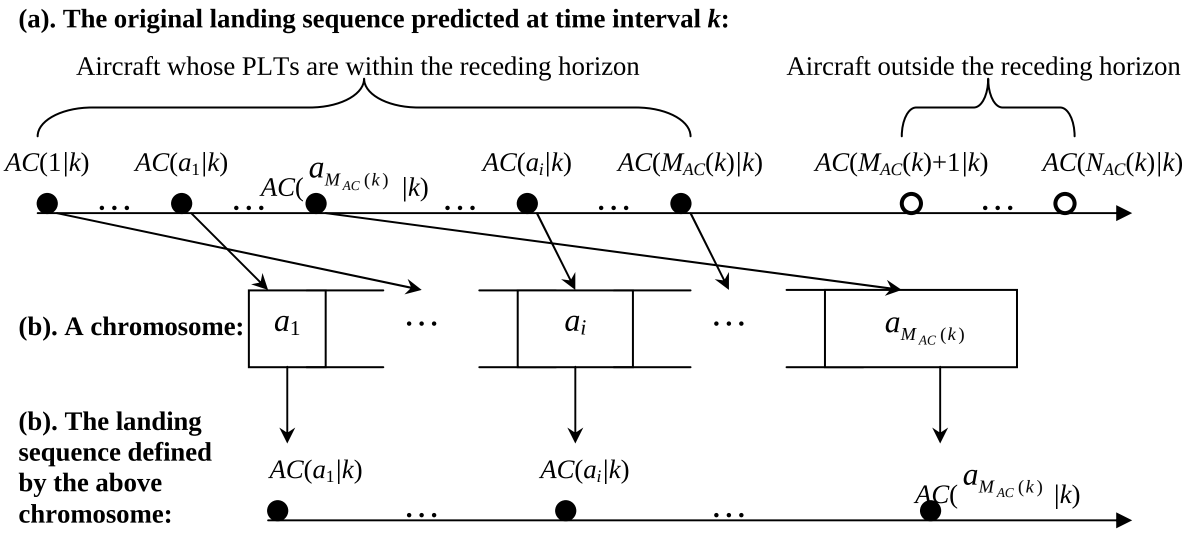
Figure 3. FCFS landing sequence and position shifting

3.2 RHC based GA

The methodology of designing our RHC based GA follows the common practice of GA:- design the structure of chromosomes, choose fitness function, define genetic operators, and introduce some necessary heuristic rule. In addition for each step, we need to take an extra factor into account: how to integrate the concept of RHC.

3.2.1 The structure of chromosomes

In the RHC strategy, generally, not all aircraft in the current arrival flow will be included in the online optimization process at each time interval. Only those aircraft whose PLTs are within the receding horizon will be taken into account. Suppose at time interval *k*, the number of those aircraft whose PLTs are within the receding horizon is $M_{AC}(k)$, then each chromosome at time interval *k* has $M_{AC}(k)$ genes. Each gene stands for a certain aircraft. The structure of chromosomes is given in Figure 4, where a_i is the serial number in the original landing sequence for the aircraft represented by the *i*th gene in the chromosome, and $(.|k)$ means the associated variable is predicted or calculated at the *k*th time interval.



(a). The original landing sequence predicted at time interval k:
Aircraft whose PLTs are within the receding horizon
Aircraft outside the receding horizon
$AC(1|k)$
$AC(a_1|k)$
$AC(a_{M_{AC}(k)}|k)$
$AC(a_i|k)$
$AC(M_{AC}(k)|k)$
$AC(M_{AC}(k)+1|k)$
$AC(N_{AC}(k)|k)$
(b). A chromosome:
a_1
a_i
$a_{M_{AC}(k)}$
(b). The landing sequence defined by the above chromosome:
$AC(a_1|k)$
$AC(a_i|k)$
$AC(a_{M_{AC}(k)}|k)$

Figure 4. Structure of chromosomes

Each chromosome defines a possible landing sequence over the receding horizon. Based on the landing order determined by the chromosome and the LTIs listed in Table 1, the ALT for each aircraft in the possible landing sequence is calculated by following FCFS principle. If k=0, $t_{\text{ALT}}(a_1|k)= t_{\text{PLT}}(a_1|k)$; otherwise, $t_{\text{ALT}}(a_1|k)$ must make sure that the LTI between $AC(a_1|k)$ and the last aircraft cleared in the (k-1)th time interval is satisfied. With the ALTs, the airborne delay of the possible landing sequence is calculated by

$$T_{delay}(k) = \sum_{i=1}^{M_{AC}(k)} (t_{ALT}(i|k) - t_{PLT}(i|k)) . \qquad (3)$$

3.2.2 Fitness function

After the airborne delay of a possible landing sequence is calculated according to (3), the fitness of the associated chromosome can be simply defined as

$$f = (T_{\max} - T_{delay}(k)) / T_{\max} \tag{4}$$

where $T_{\max}$ denotes the maximum airborne delay in a certain generation of chromosomes. However, due to the concept of RHC, the possible landing sequence probably does not include all aircraft in the original sequence. The implementation of this possible landing sequence will obviously have a certain influence on the scheduling and sequencing of those aircraft outside the current receding horizon. Neither Eq. (3) nor Eq. (4) does anything to assess this influence. This could end up with a short-sighted algorithm.

In our RHC based GA, the idea of using terminal penalty in the RHC of control engineering is borrowed to defined a novel fitness function as

$$\tilde{T}_{delay}(k) = T_{delay}(k) + \alpha \frac{T_{delay}(k)}{M_{AC}(k)} (N_{AC}(k) - \tilde{M}_{AC}(k)) (\frac{N_{AC}(k) - \tilde{M}_{AC}(k)}{K - \min(K-1, (k+N))}) / (\frac{M_{AC}(k)}{N}) \quad ,(5)$$

$$f = (\tilde{T}_{\max} - \tilde{T}_{delay}(k)) / \tilde{T}_{\max} \tag{6}$$

where $\alpha \geq 0$ is a weighting coefficient, K is the discrete-time index corresponding to the end of operating day, $\tilde{M}_{AC}(k)$ is the number of those aircraft whose ALTs are within the receding horizon, and $\tilde{T}_{\max}$ denotes the maximum $\tilde{T}_{delay}(k)$ in a certain generation of chromosomes.

The second term on the right side of Eq. (5) is the terminal penalty, which assesses the influence of the current landing sequence on those aircraft outside the receding horizon. From Eq. (5), it can be seen that the terminal penalty is a function in terms of the average delay and the density of those aircraft whose PLTs are within the receding horizon, and the number and the density of those aircraft which are outside the receding horizon after the current run of the online optimization routine. A larger average delay usually means less aircraft will be cleared during the current time interval, and therefore more aircraft will be left for future optimization processes. $\tilde{M}_{AC}(k)$, the number of those aircraft whose ALTs are within the receding horizon, is probably different from $M_{AC}(k)$, the number of those aircraft whose PLTs are within the receding horizon, particularly at a busy airport. Basically, a smaller $\tilde{M}_{AC}(k)$

means more aircraft are delayed for future scheduling and sequencing. If more aircraft are left for future optimization, or if the density of such aircraft, i.e. $(N_{AC}(k) - \tilde{M}_{AC}(k))/(K - \min(K-1,(k+N))$, is larger than the density of those aircraft whose PLTs are within the receding horizon, i.e. $M_{AC}(k)/N$, the quality of the current solution determined by the chromosome will have a stronger negative influence on the future. α is a constant weighting coefficient, which determines the contribution of the terminal penalty to $\tilde{T}_{delay}(k)$. α needs to be chosen carefully. If the predicted information is more reliable, α can be set relatively larger to avoid short-sighted performance. However, if α is too large, the influence of uncertain information in the future will become unnecessarily significant. The choice of α also depends on the maximum $N_{AC}(k)$. Specially, for given a α, if the maximum $N_{AC}(k)$ is very large, then the algorithm could also become sensitive to uncertainties. Basically, for each different airport, extensive simulation studies or experiments are necessary to find a proper value for α. In this paper, we set α=0.01 for $N_{AC}(k) \leq 30$.

From Eq. (6), one has that if $\tilde{T}_{delay}(k)$ is smaller, the fitness of the corresponding chromosome is larger, and consequently, it is relatively more likely to survive through the evolution and get more chance to reproduce offspring.

3.2.3 Genetic operators

Basically, there are two kinds genetic operators in GA: crossover and mutation. As pointed out in [19], there has long been a strong debate about the usefulness of crossover, and the general agreement today is that the answer is problem-dependent. In the ASS problem, each aircraft can appear only once in a chromosome. When information is exchanged between two chromosomes during crossover, an additional checking process has to apply to make sure that no aircraft appears more than once in any offspring chromosome. Actually, if more than 3 genes are exchanged randomly between two chromosomes, the offspring chromosomes are very likely invalid. If only two genes are exchanged, then the effect of crossover will be quite similar to that of mutation. Therefore, with the structure of chromosomes given in Section 3.2.1, it is difficult to define an effective crossover operator. In our RHC based GA for the ASS Problem, we only adopt mutation operation, which is illustrated by Figure 5.

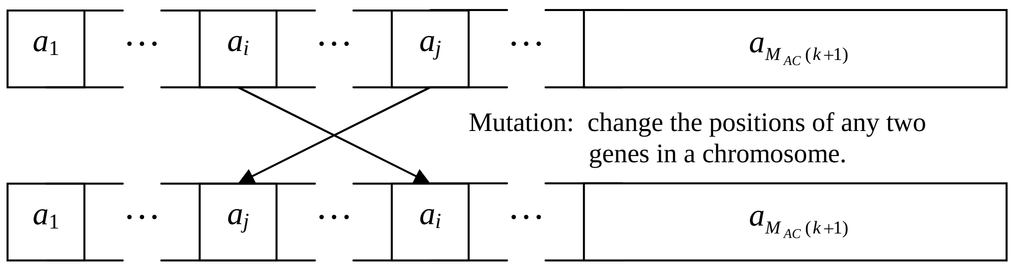

a_1
a_i
a_j
$a_{M_{AC}(k+1)}$
Mutation: change the positions of any two genes in a chromosome.
a_1
a_j
a_i
$a_{M_{AC}(k+1)}$

Figure 5. Mutation operation

3.2.4 Heuristic rules for setting algorithm parameters

To improve the solution quality as well as to increase the converging speed of GA, special problem-orient heuristic rules are always introduced for setting algorithm parameters in various practices of GA. The following are some heuristic rules proposed for our RHC based GA to resolve the position-shifting based ASS problem.

- Mutation can increase the variety of chromosomes and therefore improve the solution quality of GA. However, if mutation is more often than necessary, besides the increase of computational burden, the solution quality could degrade since good chromosomes are likely to be destroyed. To achieve a good trade-off, we employ self-adjusting mutation probability. Assuming the basic mutation probability is P_b=0.3, the actual probability P_a for mutating between gene i and gene j (j>i) is calculated in steps as follows.

  (a). Firstly, set P_a=P_b.

  (b). If $AC(a_i|k)$ and $AC(a_j|k)$ belong to the same category, set P_a=0 and the calculation of mutation probability for these two genes is over.

  (c). Otherwise, if the LTI between $AC(a_i|k)$ and $AC(a_j|k)$ is larger than the LTI between $AC(a_j|k)$ and $AC(a_i|k)$, let P_a= P_a+0.2.

  (d). Adjust P_a according to the difference between $t_{PLT}(a_i|k)$ and $t_{PLT}(a_j|k)$

$$P_a = P_a + 0.2\min(1, \frac{10}{|t_{PLT}(a_i\,|\,k) - t_{PLT}(a_j\,|\,k)|}) sign(200 - |t_{PLT}(a_i\,|\,k) - t_{PLT}(a_j\,|\,k)|)\,. \tag{7}$$

  (e). Adjust P_a according to $|a_i$-$a_j|$

$$P_a = P_a + 0.1\min(1, \frac{3}{|a_i - a_j|}) sign(10 - |a_i - a_j|)\,. \tag{8}$$

  The physical meaning in Eq. (7) and Eq. (8) is obvious: the closer two aircraft are to each other, the more likely they should change positions.

- The population in a generation N_p and the maximum number of generations in evolution N_g are another two important parameters when we design a GA. Generally, large N_p and N_g lead to high quality of solutions, but at the cost of heavy computational burden. A proper choice of N_p and N_g should be a trade-off between solution quality and computational efficiency. This usually depends on the size of solution space, in the ASS problem, the number of aircraft whose PLTs are within the receding horizon, i.e., $M_{AC}(k)$. Since $M_{AC}(k)$ changes in terms of k, N_p and N_g should also change with k

  .
$$N_p = 30 + 10(round(\max(0, M_{AC}(k) - 10)/5)), \qquad (9)$$

$$N_g = 20 + 10(round(\max(0, M_{AC}(k) - 10)/5)). \qquad (10)$$

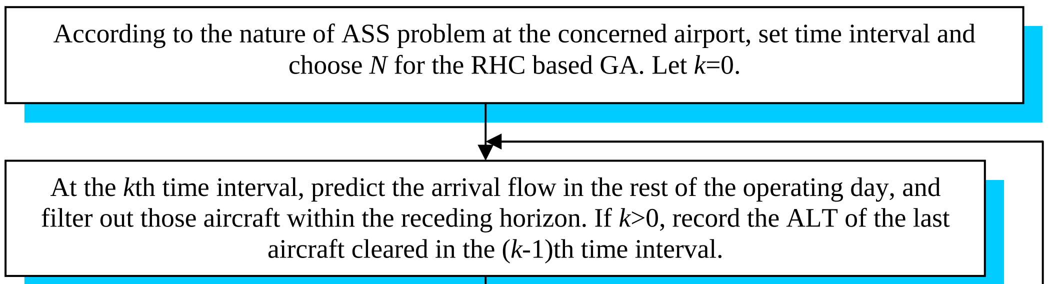
  From Eq.(9) and Eq.(10), one can see that more aircraft are included in the online optimization process, more chromosomes and more generations are needed to guarantee the solution quality. The parameters in Eq.(9) and Eq.(10) are chosen and tested through extensive simulation. In the case of $M_{AC}(k) \leq 30$, they result in satisfactory solution quality and tolerable computational burden.

3.2.5 Flow chart of the RHC based GA

With the above technical preparations, our RHC based GA for the ASS problem can be eventually developed by simply following the framework of common RHC algorithms, as illustrated by the flow chart in Figure 6.

The successful design of the RHC based GA partially depends on a proper choice of the length of receding horizon, N. If N is too small, most useful information could be missed out, and therefore, the RHC algorithm could be short-sighted and exhibit poor performance. On the other hand, if N is too large, the computational burden will become very heavy, and in addition, much more unreliable information in the future is used and could degrade the solution quality of the algorithm.

To assess whether a RHC based GA is properly designed as well as to fairly compare the new algorithm with other relevant literature, the last step in flow chart sums up the actual airborne delay occurring in the entire operating day.


According to the nature of ASS problem at the concerned airport, set time interval and choose N for the RHC based GA. Let k=0.
At the kth time interval, predict the arrival flow in the rest of the operating day, and filter out those aircraft within the receding horizon. If k>0, record the ALT of the last aircraft cleared in the (k-1)th time interval.

4. Simulation results

To fully investigate the performance of the RHC based GA proposed in this paper, which is hereafter denoted as RHC_GA, we compare it with two other algorithms based on conventional dynamic optimization (CDO) strategy: a conventional GA, denoted as CDO_GA, and the algorithm reported in [10], denoted as CDO_TSP. The algorithm in [10] is developed by modeling the ASS problem as a traveling salesman problem (TSP). Since the CDO_GA is based on the conventional dynamic optimization strategy, its chromosomes are usually longer than those in the RHC_GA, its fitness function is defined by Eq. (3) and (4), and wherever it is applicable, $M_{AC}(k)$ used in the RHC_GA should be replaced by $N_{AC}(k)$. All following simulation is conducted with a Pentium II (300MHz) based PC.

To illustrate how these three algorithms work, Table 2 gives the results of a realistic large scale problem with 30 aircraft. The data of the original arrival flow is borrowed directly from [10]. In order to compare with the data given in [10], no disturbances/uncertainties are considered at this stage. In Table 2, in the first three columns we list the aircraft serial number in the original sequence, categories and PLTs; in the second three columns, the actual sequence of landing aircraft, with their categories and ALTs, issued by the CDO_TSP; in the third three columns, the actual sequence of landing aircraft, with their categories and ALTs, issued by the CDO_GA; in the last three columns, actual the sequence of landing aircraft, with their categories and ALTs, issued by the RHC_GA with a receding horizon of 4 steps long. In the simulation, the operating interval T_{OI} is set as 5 minutes. In Table 2, the landing sequence generated by the RHC_GA is better than that of the CDO_GA, and similar to that of the CDO_TSP. However, due to the nature of GA (a stochastic optimization method), no general conclusion regarding the performance of these three algorithms can be made from just Table 2, until more simulation studies are conducted.

In [10], besides the original arrival flow used in Table 2, there are another two original arrival flows: an uncongested case and a congested case. We also apply the CDO_GA and the RHC_GA to them. To distinguish from the case in Table 2, we consider the case in Table 2 as a normal case. In either of these three cases, the total number of aircraft is 30, but they have different degrees of congestion. The detailed data of the original arrival flows can be found in [10]. We then introduce 20% uncertainties (the PLTs of 20% of the aircraft are not correct) into these three original arrival flows to simulate a dynamic ATC environment. In each case, the uncertainties are generated randomly for 50 times, and the data of uncertainties are saved in 50 files. For the CDO_GA and RHC_GA, 10 simulation runs with the same data of uncertainties saved in each of 50 files are conducted. In total, simulation is conducted for 500 times in each case. Since the CDO_TSP is not a stochastic optimization method (refer to [10]), only one simulation run is conducted with the data in each of 50 files, and therefore in total 50 simulation runs are conducted in each case. Due to limited space, we only list some key data of our simulation results in Table 3. From the data in Table 3, one can see that, regarding the average delay of each aircraft, the solution quality of the CDO_GA is the worst, while the RHC_GA achieves even better performance than the CDO_TSP. Although the CDO_GA and the CDO_TSP consider all aircraft in the original arrival flow, clearly they do not achieve the global optimal solution. There are two reasons for this. Firstly, because they both belong to the family of generate-and-test methods (for more details about CDO_TSP, see [10]), the quality of either CDO_GA or CDO_TSP is sensitive to the size of the solution space. Secondly, there are uncertainties in the predicted arrival flows. For the RHC_GA, the size of solution space for each online optimization routine is mainly determined by the length of receding horizon. By choosing a proper receding horizon, it is relatively easier to maintain the solution quality over the receding horizon at a certain level as well as to reduce the influence of uncertainties. By the nature of the ASS problem, small delay at each step usually results in a small total delay in the operating day. Therefore, the RHC_GA achieves the best performance. Although the CDO_TSP is much more effective than the CDO_GA, from the relevant data in Table 3 (e.g., the landing orders for aircraft 15-17), one can see that the CDO_TSP sometimes makes unnecessary position-shiftings, which also explains why the RHC_GA is better even than the CDO_TSP.

As for computational burden, thanks to the receding horizon strategy, the average computational time by an online optimization routine under the RHC_GA is the least. As the original arrival flow becomes more congested, RHC_GA requires more time for each online optimization routine, because more aircraft have PLTs within the receding horizon. The computational time consumed by CDO_GA or CDO_TSP is not sensitive to the degree of congestion of the original arrival flow, since all aircraft are taken into account at each step. According to the data of arrival flows in CDO_TSP, the operating day is shorter than 2 hours. From the computational time listed in Table 3, one can image the advantages of the RHC_GA against either CDO_GA or CDO_TSP regarding computational efficiency when the operating day becomes 8 hours long or even longer.

Table 4 shows the influence of the length of receding horizon on the performance and the computational burden of the RHC_GA. From the data in Table 4, one can see that, in general, as *N* increases, the performance of the RHC_GA improves at first, and then degrades when *N* is too large. This is understandable. Due to the natures of the GA optimizer and of the ASS problem, the performance should degrade with the length of receding horizon. But a too short receding horizon could result in short-sighted performance, e.g., when *N*=1. Clearly, in this simulation, the receding horizon should be within 2 to 4 to achieve the best performance. Online computational burden is no doubt increasing as *N* goes up.

Table 2. Results of CDO_GA and RHC_GA in a single test

PAF			Scheduled by CDO_TSP				Scheduled by CDO_GA				Scheduled by RHC_GA			
AC No.	Cat.	PLT (s)	AC No.	Cat.	ALT (s)	Delay (s)	AC No.	Cat.	ALT (s)	Delay (s)	AC No.	Cat.	ALT (s)	Delay (s)
1	1	0	1	1	0	0	1	1	0	0	1	1	0	0
2	1	79	2	1	96	17	2	1	96	17	2	1	96	17
3	1	144	3	1	192	48	3	1	192	48	3	1	192	48
4	2	204	5	1	288	24	5	1	288	24	5	1	288	24
5	1	264	6	1	384	64	6	1	384	64	6	1	384	64
6	1	320	7	2	584	380	4	2	584	380	4	2	584	380
7	2	528	4	2	664	136	7	2	664	136	7	2	664	136
8	1	635	9	2	744	14	9	2	744	14	9	2	744	14
9	2	730	10	2	824	58	8	1	816	181	10	2	824	58
10	2	766	11	1	896	261	11	1	912	122	8	1	896	261
11	1	790	12	1	992	202	12	1	1008	88	11	1	992	202
12	1	920	8	1	1088	168	10	2	1208	442	12	1	1088	168
13	3	1046	17	2	1288	55	15	2	1288	152	15	2	1288	152
14	4	1106	16	2	1368	202	16	2	1368	202	16	2	1368	202
15	2	1136	15	2	1448	312	14	4	1478	372	17	2	1448	215
16	2	1166	14	4	1558	452	13	3	1548	502	13	3	1518	472
17	2	1233	13	3	1628	582	17	2	1648	415	14	4	1648	542
18	1	1642	20	3	1770	0	18	1	1720	78	18	1	1720	78
19	1	1715	18	1	1842	200	19	1	1816	101	19	1	1816	101
20	3	1770	19	1	1938	223	21	1	2074	0	20	3	1997	227
21	1	2074	21	1	2074	0	20	3	2255	485	21	1	2074	0
22	1	2168	22	1	2170	2	23	4	2385	126	22	1	2170	2
23	4	2259	23	4	2398	139	24	2	2465	38	23	4	2398	139
24	2	2427	24	2	2478	51	26	2	2679	0	24	2	2478	51
25	1	2481	25	1	2550	69	27	3	2883	0	25	1	2550	69
26	2	2679	26	2	2750	71	22	1	2955	787	26	2	2750	71
27	3	2883	27	3	2883	0	25	1	3051	570	27	3	2883	0
28	2	2982	28	2	2983	1	29	1	3147	101	28	2	2983	1
29	1	3046	29	1	3055	9	30	1	3243	152	29	1	3055	9
30	1	3091	30	1	3151	60	28	2	3443	461	30	1	3151	60

Table 3. Some key data of simulation results

(Average results)	Normal case		Uncongested case		Congested case	
	Delay (s)	Comp. time (s)	Delay (s)	Comp. time (s)	Delay (s)	Comp. time (s)
CDO_TSP	142.1176	4.7032	60.7072	4.6137	436.5118	4.7482
CDO_GA	158.5079	7.7441	67.9746	7.5059	466.0831	7.6825
RHC_GA	137.4880	3.3051	58.7167	2.1524	435.6017	4.3651

Table 4 The influence of the length of receding horizon on the RHC_GA

(Average results)		Normal case		Uncongested case		Congested case	
		Delay (s)	Comp. time (s)	Delay (s)	Comp. time (s)	Delay (s)	Comp. time (s)
Length of receding horizon *N*	1	141.9092	1.5114	65.2424	1.1536	438.9676	1.7516
	2	137.8345	2.0611	57.5146	1.4865	435.7356	2.8245
	3	136.4492	2.4928	57.8706	1.8085	437.1915	3.2649
	4	137.4880	3.3051	58.7167	2.1524	435.6017	4.3651
	5	144.7620	4.1705	61.0630	2.9405	442.5289	4.7476
	6	152.3775	4.7062	60.4931	3.6436	450.5692	5.3862

5. Conclusions

Airports, especially busy hub airports, prove to be the bottlenecks in the air traffic control system. How to carry out arrival scheduling and sequencing effectively and efficiently is one of main concerns in improving the safety, capacity and efficiency of airports. This paper proposes a novel Receding Horizon Control (RHC) based Genetic Algorithm (GA) to help solve the problem of Arrival Scheduling and Sequencing (ASS) in a dynamic environment. The emphasis is put on the methodology of how to integrate the RHC strategy into GA, and special attention is paid to receding horizon and terminal penalty. The potential advantages of the proposed RHC based GA for the ASS problem, with regard to airborne delay and computational burden, are investigated. In general, it is found that for the ASS problem, the RHC based GA achieves much better performance than a pure GA, while requiring much less computational time.

6. Acknowledgements

This work is supported by an ORS Award from the Overseas Research Students Awards Scheme, Universities UK.

Reference

[1] Pelegrin, M., “Towards Global Optimization for Air Traffic Management”, AGARD-AG-321, 1994.

[2] Gregory, C.C., H. Erzberger and F. Neuman, “Delay exchanges in arrival sequencing and scheduling”, *Journal of Aircraft*, vol.36, no.5, pp. 785-791, 1999.

[3] Gregory, C.C., H. Erzberger and F. Neuman, “Fast-time study of aircraft-influenced arrival sequencing and scheduling”, *Journal of Guidance, Control and Dynamics*, vol.23, no.3, pp. 526-531, 2000.

[4] Andreatta, G. and G. Romanin-Jacur, “Aircraft flow management under congestion”, *Trans. Science*, vol.21, pp. 249-253, 1987.

[5] Bianco, L. and MM. Bielli, “System aspects and optimization models in ATC planning” in: Bianco, L. and A.R. Odoni (Eds.), Large Scale Computation and Information Processing in Air Traffic Control. Springer Verlag, pp. 47-99, 1993.

[6] Dear, R.G., “The dynamic scheduling of aircraft in the near terminal area”, FLT R76.9, Flight Transportation Laboratory, M.I.T., Cambridge, 1976.

[7] Psaraftis, H.N., “A dynamic programming approach to the aircraft sequencing problem”, FTL R78-4, Flight Transportation Laboratory, M.I.T., Cambridge, 1978.

[8] Psaraftis, H.N., “A dynamic programming approach for sequencing identical groups of jobs”, *Opns. Res.*, vol.28, pp. 1347-1359, 1980.

[9] Venkatakrishnan, C.S., A. Barnett and A.M. Odoni, “Landings at Logan Airport: Describing and increasing airport capacity”, *Transp. Sci.*, vol.27, pp. 211-227, 1993.

[10] Bianco, L., P. Dell’Olmo and S. Giordani, “Scheduling models and algorithms for TMA traffic management” in: Bianco, L., P. Dell’Olmo and A.R. Odoni (Eds.), Modelling and Simulation in Air Traffic Management. Springer Verlag, pp.139-167, 1997.

[11] Bianco, L., G. Rinaldi and A. Sassano, “Scheduling task with sequence-dependent processing times”, *Naval Res. Log.*, vol.35, pp. 177-184, 1988

[12] CLARKE, D.W., *Advances in Model-based Predictive Control*, Oxford University Press, 1994.

[13] De Schutter, B. and T. van den Boom, “Model predictive control for max-plus-linear discrete event systems”, *Automatica*, vol.37, no.7, pp. 1049-1056, 2001.

[14] Van Den Boom, T. and B. De Schutter, “Properties of MPC for max-plus-linear systems”, *European Journal of Control*, vol.8, pp. 453-462, 2002.

[15] Chand, S., V.N. Hsu and S. Sethi, “Forecast, solution, and rolling horizons in operations management problems: a classified bibliography”, *Manufacturing & Service Operations Management*, vol.4, no.1, pp.25-43, 2002.

[16] Chen, W.H., T. van den Boom and B. De Schutter, “Discussion on: ‘Properties of MPC for max-plus-linear systems’ by T. van den Boom and B. De Schutter”, *European Journal of Control*, vol.8, pp. 463-464, 2002.

[17] Mitchell, M., An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.

[18] Banzhaf, W., P. Nordin, R.E. Keller, F.D. Francone, Genetic Programming-An Introduction, Morgan Kaufmann, San Francisco, CA, 1998.

[19] Eiben, A.E., M. Schoenauer, “Evolutionary computing”, Information Processing Letters, Vol.82, pp.1-6, 2002.