GENETIC ALGORITHM BASED ON RECEDING HORIZON CONTROL FOR REAL-TIME IMPLEMENTATIONS IN DYNAMIC ENVIRONMENTS

Xiao-Bing Hu, Wen-Hua Chen

Department of Aeronautical and Automotive Engineering,
Loughborough University, UK
Email: X.Hu@lboro.ac.uk, W.Chen@lboro.ac.uk

Abstract: This paper introduces the concept of Receding Horizon Control (RHC) to Genetic Algorithm (GA) for real-time implementations in dynamic environments. The methodology of the new GA is presented with the emphases on how to effectively integrate the RHC strategy by following some RHC practices in control engineering, particularly, how to choose the length of receding horizon and how to design terminal penalty. Simulation results show that, when the RHC based GA is applied in dynamic environments, both computational efficiency and performance are improved in comparison with existing GAs.

Keywords: Receding horizon control, genetic algorithm, chromosome, fitness function, terminal penalty.

1. INTRODUCTION

Genetic algorithm (GA) is a large-scale parallel stochastic searching and optimizing method inspired by the biological mechanisms of evolution and heredity. It has now been widely used for numerical optimization, combinatorial optimization, classifier systems and many other engineering problems (Goldberg, 1989, Mitchell, 1996).

GA falls in the category of generate-and-test algorithms (Fogel, 1998). Consequently, the solution quality and convergence speed of GA strongly depend on the size of solution space of the concerned problem. Generally, as the size of solution space increases, the solution quality of GA degrades, and the convergence speed of GA slows down. These two negative changes in terms of the size of solution space often make real-time implementations of GAs unrealistic for complex problems, since online computation process has to be subject to a certain time limit. Another issue which has to be addressed is the influence of disturbances/uncertainties on the solution quality in dynamic environments.

This paper integrates the concept of RHC (Receding Horizon Control), or MPC (Model Predictive Control), into GA, delivering a new algorithm suitable for real-time implementations to a wide range of optimization problems in dynamic environments. RHC is an *N*-step-ahead online optimization strategy, and has been widely adopted in control engineering and proved very successful to control plants and processes in dynamic environments (Clarke, 1994). Recently, several researchers studied how to use GA as online optimizer for MPC in control engineering (Onnen et al, 1997, Sarimveis and Bafas, 2003). Differently, this paper focuses on how to make RHC serve GA, in order to develop a general GA-based methodology for various engineering problems, not just for control engineering, in dynamic environments. Every time the RHC based GA conducts online optimization, it is only interested in the period of current receding horizon rather than the entire dynamic process. How to choose the length of receding horizon and how to design terminal penalty are two key factors in the success of the new algorithm for real-time implementations in dynamic environments. As a result of introducing the concept of RHC, the real-time properties of GA are improved, and solution quality of GA becomes more robust under the influence of disturbances and/or uncertainties in dynamic environments.

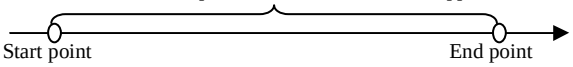
2. RHC STRATEGY

Receding horizon control has proved to be a highly effective online optimization strategy in the area of control engineering, and it exhibits many advantages against other control strategies (Clarke, 1994). 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. In this framework, decision is made by looking ahead for *N* steps in terms of a given cost/criterion, and the decision is only implemented by one step. Then the implementation result is checked, and a new decision is made by taking updated information into account and looking ahead for another *N* steps. RHC has now been widely accepted in the area of control engineering. Recently, attentions have been paid to

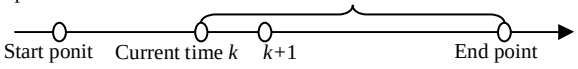
applications of RHC to areas like management and operations research (Schutter and Van den Boom, 2001, Chand et al, 2002).

Fig. 1 compares the RHC strategy with some other optimization strategies in an intuitive way. It is evident that offline optimization strategy, as shown in Fig. 1.(a), is not suitable for dynamic environments, although it is ideal for simplifying problems and analyzing algorithms theoretically. Most GAs in literature are designed and then tested mainly by using offline strategy, e.g., see Sharma et al (2004). As response to changes in dynamic environments, an online optimization routine is usually run periodically to re-calculate the decision for the period from the current moment to the end of dynamic process. We call this strategy conventional dynamic optimization (CDO), as shown in Fig. 1.(b). It is straightforward to transform an offline optimization strategy based algorithm into CDO strategy, since no modification to the algorithm is required but just initial conditions change. The real-time properties could be a problem since offline algorithm considers no time limit to the optimization process. The another problem in CDO strategy is that the performance could be very sensitive to disturbances and/or uncertainties in dynamic environments. However, to apply GAs in dynamic environments, existing literature simply follows the CDO strategy (Hu et al, 2001).

(a). **Offline optimization:** Optimize the whole dynamic process based on the predicted information in advance, and then the solution is implemented no matter what happens.



(b). **Conventional dynamic optimization (CDO):** Optimize over the period from the current time k to the end of the dynamic process, 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.



(c). **Receding horizon control (RHC):** Optimize 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.

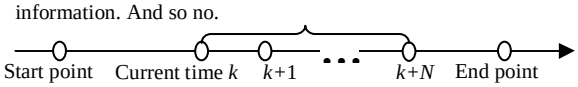


Fig. 1. Basic ideas of some optimisation strategies

As illustrated in Fig. 1.(c), thanks to the idea of receding horizon, RHC strategy provides a possible solution to the problems the CDO strategy faces. A properly chosen receding horizon can effectively filter out most unreliable information and reduce the scale of problem. The latter is especially important to any time-consuming algorithms such as GA to satisfy the time limit on online optimization process. However, how to integrate RHC strategy into GA to develop an effective and practicable method for real-time implementations in dynamic environments is more than to simply change initial and final conditions for an offline strategy based GA. To make them work in harmony, the GA based online optimiser should be designed from an RHC point of view in the first place. Therefore, unlike other literature, in this paper, there is no offline strategy based GA to be transformed into online algorithm. As will be explained in depth later, some techniques, particularly terminal penalty, which is widely used by RHC in control engineering, are adopted to design the RHC based GA proposed in this paper.

3. RHC BASED GA

Consider a dynamic process with starting time T_S and ending time T_E, denoted as $DP(T_S, T_E)$. The purpose is to make a series of decision, plan, or control, etc, during the time period from T_S to T_E, such that the behaviour of $DP(T_S, T_E)$ would be optimal in terms of a certain objective function. Suppose that the decision-making process is based on a discrete-time frame in a dynamic environment, each step (sampling interval) is t_{SI}, and the process $DP(T_S, T_E)$ is E steps long. Let k indicate the time index. k=0 corresponds to the starting time T_S, and k=E to the ending time T_E. Let $s(k)$, k=0,…,E-1, denote the decision executed in the kth step/interval, and $x(k)$, k=0,…, E, denote the internal state of $DP(T_S, T_E)$ at time instant k. The behaviour of $DP(T_S, T_E)$ in a dynamic environment is judged by an objective function

$$J_0 = J_0([s(0), s(1), \cdots, s(E-1)], [x(1), x(2), \cdots, x(E)]) . \quad (1)$$

Hereafter, we call $s(\cdot)$ as control signal, $x(\cdot)$ as system state and J_0 as performance index, in order to keep consistent with the terminologies of RHC. As shown in (1), J_0 is usually a function of control history and system state trajectory. Control history is basically related to consumed energy/cost, and system state trajectory is often expected to follow a specified reference or have a certain feature. The purpose to optimize the process $DP(T_S, T_E)$ is to achieve desirable system targets while consuming as less as possible energy/cost. In a dynamic environment, $s(k)$ needs to be online calculated in real-time at time instant k, and then implemented during the kth step/interval.

Following the CDO strategy, existing GAs for real-time implementations in dynamic environments are designed based on the following online optimization problem to calculate $s(k)$:

$$\min_{S_1(k) \in \phi_1(k)} J_1(k) , \text{ subject to } \Theta_1(k) \quad (2)$$

where $J_1(k)$ is a new performance index, $\phi_1(k)$ is the solution space, $\Theta_1(k)$ is a set of constraints, and $S_1(k)$ is a solution for the period from the current time to the end of dynamic process

$$S_1(k) = [s(k|k), s(k+1|k), \cdots, s(E-1|k)], \; E \geq k+1 \quad (3)$$

where $s(k+i|k)$, i=0,…,E-k-1, is the control signal for the (k+i)th time interval but determined at the kth time instant, different from $s(k)$, which is the control signal implemented at the kth time instant. $J_1(k)$ is usually defined based on J_0 but in terms of predicted control history and predicted system states

$$J_1(k) = J_1([s(k|k), s(k+1|k), \cdots, s(E-1|k)], [x(k+1|k), x(k+2|k), \cdots, x(E|k)]) \quad (4)$$

where $x(k+i|k)$, $i=1,\ldots,E-k$, is the $(k+i)$th time instant system state predicted at the kth time instant.

Since RHC is an N-step-ahead online optimization strategy, the online optimization problem (2) needs to be reformulated as follows: at time instant k (or the kth step), resolve

$$\min_{S_2(k)\in\phi_2(k)} J_2(k), \text{ subject to } \Theta_2(k) \quad (5)$$

where

$$S_2(k)=\begin{cases}[s(k|k),s(k+1|k),\cdots,s(k+N-1|k)], & k<E-N\\ S_1(k), & k\geq E-N\end{cases}, \quad (6)$$

is a solution for the period of receding horizon at time instant k, N is the length of receding horizon, $\phi_2(k)$ is the solution space for $S_2(k)$, $\Theta_2(k)$ is a set of constraints, and

$$J_2(k)=\begin{cases}\tilde{J}_1([s(k|k),s(k+1|k),\cdots,s(k+N-1|k)],\\ [x(k+1|k),x(k+2|k),\cdots,x(k+N|k)]) & k<E-N\\ +P(x(k+N|k))\\ J_1(k) & k\geq E-N\end{cases} \quad (7)$$

is the performance index under RHC strategy. In $J_2(k)$, $\tilde{J}_1$ is a function similar to $J_1(k)$ in (4), and $P(x(k+N|k))$ is terminal penalty, which has been widely used in control engineering for guaranteeing the stability of RHC. These variables and functions in Problem (5) will be explained and analysed in details later. Since Problem (5) is the same as Problem (2) when $k\geq E-N$, hereafter, only the case of $k<E-N$ is considered in RHC based GA.

Clearly, the concept of RHC is explicitly reflected in $S_2(k)$ and $J_2(k)$. With $S_2(k)$ and $J_2(k)$ defined for a certain problem, the methodology of how to design RHC based GA is given by the flow chart in Fig. 2. From Fig. 2, it is clear that, besides common practices in GAs such as crossover and mutation, the concept of RHC is integrated into GA during choosing N_p and N_g and designing the chromosome structure and fitness function. As will be shown later, the introduction of RHC strategy is not as simple as it looks in Fig. 2, and once RHC is properly integrated, the RHC based GA will become much more advanced than existing GAs.

Remark 1: Different from existing GAs designed based on Problem (2), the RHC based GA grounds on Problem (5). Hereafter, for the sake of identification, existing GAs are denoted as CDO_GA, and the new algorithm proposed in this paper as RHC_GA. From the definitions of $S_1(k)$ and $S_2(k)$, one can see that, in most time, $S_2(k)$ is shorter than $S_1(k)$. Suppose each control signal $s(\cdot)\in R^h$ and $k<E-N$, then $S_1(k)\in R^{(E-k)\times h}$ and $S_2(k)\in R^{N\times h}$ at time instant k, i.e., $\phi_1(k)\subseteq R^{(E-k)\times h}$ and $\phi_2(0)\subseteq R^{N\times h}$. There may be some kind of implicit relationship between $\phi_1(k)$ and $\phi_2(k)$, but $\phi_2(k)$ is generally not a subset of $\phi_1(k)$. As for $\Theta_1(k)$ and $\Theta_2(k)$, they should both include the same system dynamics, constraints for single control signal and single system state. If $\Theta_1(k)$ has some constraints for $S_1(k)$ rather than for a single control signal, then $\Theta_2(k)$ should also have similar but not exactly the same constraints for $S_2(k)$ due to the introduction of receding horizon.

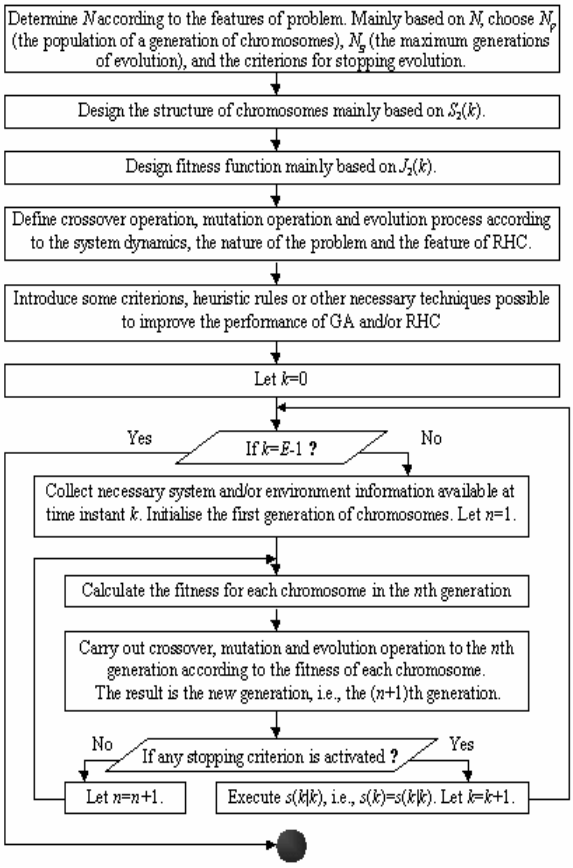


Fig. 2. Flow chart of RHC_GA

Remark 2: GA is usually very time-consuming. One motivation of introducing RHC to GA is to reduce the computational burden so that the new algorithm could be practicable for real-time implementations. Since GA is stochastic searching algorithm by nature, the computational burden theoretically depends on the size of solution space ϕ. Suppose each element in a control signal, $s_i(.)$, $i=1,\ldots,h$, is represented by d genes in a chromosome, and the information recorded in each gene can vary within a certain discrete-value set, which has w different members. Theoretically, one then has

$$Size(\phi_1(k)) = w^{(E-k-N)\times h\times d} Size(\phi_2(k)). \quad (8)$$

Practically, the computational burdens of CDO_GA and RHC_GA can be estimated in another way. Basically, to prevent the performance of GA from significantly degrading as the result of larger solution space (longer chromosomes), the population in a generation of chromosomes, N_p, and the maximum generations of evolution, N_g, should increase to a reasonable level correspondingly. Suppose they are both simply proportional to the length of chromosomes. At time instant $k<E-N$, one has

$$N_{c,1}(k) = (round((E-k)/N))^2 N_{c,2}(k). \quad (9)$$

where $N_{c,1}(k)$ and $N_{c,2}(k)$ are the numbers of all chromosomes used by CDO_GA and RHC_GA, respectively. For those operations carried out based

on chromosomes, such as calculating fitness and then sequencing, the corresponding computational burden for CDO_GA may be $\Psi_c((round((E-k)/N))^2)$ times heavier than that of RHC_GA, where Ψ_c is a function determined by the algorithms adopted for those chromosome-based operations. For those operations based on genes, such as crossover and mutation, how many times they apply to a single chromosome often depends on the length of chromosome. Simply suppose the total times these gene-based operations apply to a chromosome is proportional to the length of chromosome. Therefore, the gene-based computational burden in CDO_GA is $(round((E-k)/N))^2\Psi_g(round((E-k)/N))$ times heavier than that of RHC_GA, where Ψ_g is determined by crossover and mutation algorithms.

Remark 3: If the algorithms adopted for chromosome-based and gene-based operations are already chosen, the upper bound of computational burden in CDO_GA mainly depends on *E*, which is a system parameter and cannot be changed artificially. If a sampling interval is fixed, for those *E* resulting in an upper bound larger than a sampling interval, real-time implementation of GA is impossible. While in RHC_GA, the upper bound of computational time can be adjusted by choosing different *N*, making sure that the sampling interval is not exceeded, no matter what *E* is. Therefore, the introduction of RHC to GA can significantly reduce the computational burden, and as a result, RHC_GA is suitable for real-time implementations.

Remark 4: The terminal penalty $P(x(k+N|k))$ is used to estimate the cost when the system runs from the terminal state $x(k+N|k)$ to the end of dynamic process. How to design a proper terminal penalty $P(x(k+N|k))$ is a crucial issue for RHC_GA. This mainly depends on the nature of the dynamic process and the problem to be solved. For some systems, $T(x(k+N|k))=0$ in RHC_GA leads to no degradation of performance. For example, the sequencing problem in traffic systems or service systems, where minimizing delays is often the main concern, probably does not need terminal penalty. By the nature of sequencing problem, generally, if the delay at the early stage is small, then the total delay in the whole dynamic process is also small. Therefore, for such systems, as long as the delay over the receding horizon is minimized, the terminal penalty is not necessary. However, terminal penalty $P(x(k+N|k))$ may be crucial to many other systems, particularly to those systems with special constraints on the system states at the end of dynamic process. For example, in route planning problems, besides a certain performance index to be minimized, the destination must be reached. A successful implementation of RHC_GA in route planning problems definitely depends on a properly designed terminal penalty. If $P(x(k+N|k))=0$ for route planning problem, RHC_GA might never lead to the destination.

Remark 5: Disturbances and/or uncertainties in dynamical environments are another motivation for introducing the concept of RHC to GA. In a dynamical environment, basically, the accuracy of information decreases with time. In other words, the information for the farther future is more uncertain and therefore more unreliable. CDO_GA simply exposes its performance to all disturbances and/or uncertainties, while RHC_GA can filter out most unreliable information. It is very important to choose a receding horizon of proper length. Firstly, RHC_GA uses the receding horizon as a filtering window. Any information beyond this window will not be used for current optimization since it is usually more unreliable. If the receding horizon is too long, much unreliable information will be involved in optimization. While if too short, some necessary information will be filtered out and consequently the performance of RHC_GA becomes short-sighted. Secondly, the choice of length of receding horizon is subject to real-time properties, as discussed in Remark 3.

4. EXAMPLES

Two case studies are reported in this section in order to demonstrate how to design an effective RHC_GA for real-time implements in dynamic environments. The emphasis is on those RHC related techniques and details, particularly the terminal penalty and the length of receding horizon.

4.1 Implementation to free-flight path optimization

"Free Flight (FF)" is one of the most ambitious and promising schemes in the development and innovation of future aviation concepts and air traffic systems. How to online optimize the FF path efficiently in terms of a specified cost index in a dynamic environment is always a challenging problem. Hu et al (2001), following the CDO strategy, reported an improved GA-based approach to attack this problem. Here most techniques reported by Hu et al (2001) are adopted, but in order to introduce the concept of RHC, the structure of chromosomes has to be modified, and the fitness function has to be re-defined by integrating terminal penalty.

The structures of chromosomes for CDO_GA and RHC_GA are given in Fig. 3. The last way-point in a chromosome for CDO_GA is always the destination airport, while the last way-point in a chromosome for RHC_GA could be anywhere in the available airspace. According to Hu et al (2001), the flight time between any successive way-points (except the last two successive way-points) is a constant, i.e., a time interval. At first sight, the flight time associated with a chromosome for RHC_GA seems to be a constant, i.e., the length of receding horizon. However, after the mutation operation introduced by Hu et al (2001) to take shortcut, as illustrated in Fig. 4, the new flight time is uncertain and usually shorter than receding horizon.

Suppose the flight time t to the destination airport needs to be minimized. For CDO_GA

$$t = mT_{ti} + t_{lt} \tag{10}$$

where t is the flight time associated with a potential FF path, T_{ti}=10 minutes is the time interval, m is the number (integer) of time intervals, t_{lt} is the flight time between the last two way-points. Since the last way-point in a chromosome for RHC_GA could be anywhere in the available airspace, if (10) is applied straightforward, then RHC_GA could never find an FF path leading to the destination airport, let alone minimizing the flight time. Let $P_{last}(k)$, $P_{prev}(k)$ and $P_{D.A.}$ denote the last way-point, the second last way-point in a potential flight path, and the destination airport, respectively. To make RHC_GA work properly, a terminal penalty is necessary for modifying the fitness function

$$t = lT_{ti} + t_{ti} + P_{tw}(k), \tag{11}$$

$$P_{tw}(k) = (\beta |\theta_3| / \theta_4 + 1) dis(P_{last}(k), P_{D.A.}) / v_G, \tag{12}$$

where $P_{tw}(k)$ is the terminal penalty, "dis" is a function calculating the distance between two way-points, θ_3 and θ_4 are angles illustrated in Fig. 5, and $\beta > 0$ is a coefficient for tuning. $\theta_3 > 0$ means the heading of the last sub-trajectory in a potential flight path is over-turning. Oppositely, $\theta_3 < 0$ means under-turning. In either case, it will be penalized.

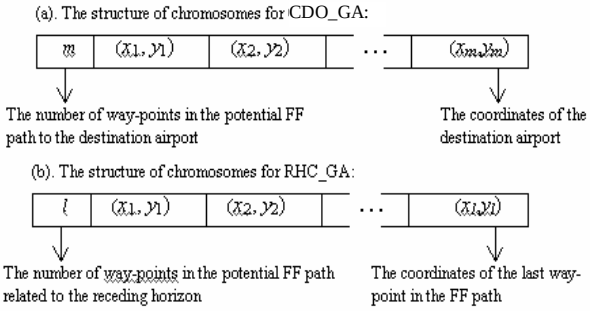


Fig. 3. Structure of chromosomes

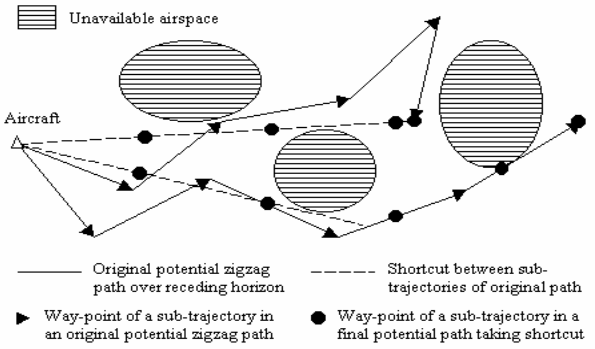


Fig. 4. Zigzag path and shortcut

In order to evaluate RHC_GA, the simulation system reported in Hu et al (2001) is adopted to set up different FF environments, and the CDO_GA in Hu et al (2001) is also used for comparative purposes. Due to limited space, only the case of DD=2000 nms (Direct Distance from the source airport to the destination airport), the most complicated case in Hu et al (2001), is considered. The comparative simulation focuses on online computational times (OCTs) and actual flight times (AFTs) from the source airport to the destination airport.

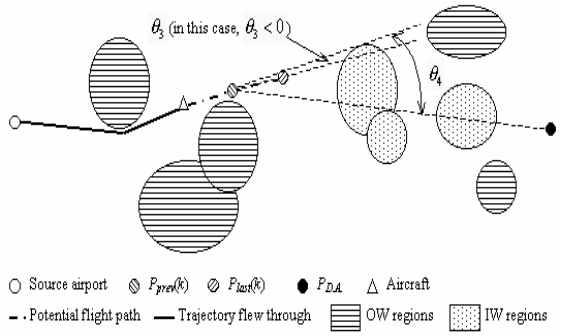


Fig. 5. Terminal penalty in Eq. (12)

Table 1 shows the influence of N, the length of the receding horizon, on the performance of RHC_GA with terminal penalty defined by (12). If N is too small, the performance is very poor, as the case of N=1 and 3. If N is too large, OCTs increase, but the performance is not necessarily improved further. Instead, the performance could degrade due to unreliable information in a dynamic environment, as shown by the case of N=9. Compared with a time interval (10-minute-long), the OCTs in Table 1 can be ignored. Therefore, in the following simulation, N=6 is adopted since it gives the least AFT.

Table 1 Influence of N on the RHC (second)

	N=1	N=3	N=6	N=9
OCT	1.26	2.27	8.88	17.74
AFT	16922	16207	15932	16274

Now, we compare RHC_GA with the CDO_GA in Hu et al (2001). Although RHC_GA is mainly proposed for dynamic process, it is still necessary to investigate its performance in a static environment. Table 2 gives the comparison results in both static case and dynamic case. From Table 2, one can see, the CDO_GA achieves the best performances, i.e., the least AFT, in static case. This is understandable, because, theoretically, in a static environment, CDO strategy should be the best in terms of performance. Table 2 also shows that the performance of RHC_GA in static case is very close to that of CDO_GA, which means the proposed RHC_GA works very well. In dynamic case, the performance of RHC_GA is better than that of CDO_GA. The reason for this has already been fully discussed in Section 3. As for OCTs, in either static case or dynamic case, RHC_GA provides reliable and promising real-time properties, while the CDO_GA seems struggling to meet the time limit of 600 seconds (a time interval). In inter-continental flights, DD is usually larger than 2000 nms, and consequently requires more OCT for CDO_GA. This implies that CDO_GA is far away from the stage of practical implementations due to its long OCT. In the case of RHC_GA, as long as N is fixed, say N=6, the OCT is always a fraction of the time limit, no matter how large DD is. This means that the RHC_GA proposed in this paper is a real solution to the FF path optimization problem.

Table 2 Comparison results (second)

	Static case		Dynamic case	
	CDO_GA	RHC_GA	CDO_GA	RHC_GA
Ave. OCT	77.54	7.30	68.92	8.88
Ave. AFT	14868	14905	16192	15932
Max. OCT	364.92	15.55	347.92	17.69
Max. AFT	14913	15052	16638	16118

4.2 Implementation to arrival scheduling and sequencing at airports

Arrival scheduling and sequencing (ASS) is one of main concerns to improve the safety, capacity and efficiency of airports. Simply speaking, ASS is the function of generating efficient landing sequences and landing times for arrivals at the airport such that the safety separation between arrival aircraft is guaranteed, the available capacity at the airport is efficiently used and airborne delays are significantly reduced. The safety separation, i.e., minimum LTI (Landing Time Interval), between a pair of successive aircraft is a function of the type and of the relative positions of the two aircraft. By shifting positions of aircraft in the original landing sequence, it is possible to reduce delays and to improve the capacity of the airport. The position-shifting based ASS problem is an NP complete problem. Basically, GA is suitable for solving this problem. By following the methodology proposed in this paper, Hu and Chen (2005) reported a RHC based GA for the ASS problem, which employed a special terminal penalty and exhibited computational efficiency and robust performance when compared with CDO_GA.

In this sub-section, we remove the special terminal penalty from the RHC_GA in Hu and Chen (2005) in order to further study its role in the ASS problem. Some simulation results are listed in Table 3, where one can see that the terminal penalty does not really matter in the ASS problem. This is understandable. By the nature of ASS problem, if the airborne delay related to the leading aircraft is small, then, generally, the delay related to the following aircraft is also small. In other words, if the airborne delay in each time interval is small, then the total delay of entire operating day is usually small. Because of this nature, one can remove the terminal penalty from the RHC_GA reported in Hu and Chen (2005). From Table 3, one can also see that, as N increases, the performance of RHC_GA improves at first, and then degrades when N is too large.

Table 3 Influence of N and terminal penalty on RHC_GA

Terminal penalty?		N=1	N=2	N=3	N=4	N=5	N=6
No	Delay (s)	142	139	136	138	148	151
	OCT (s)	1.4	2.1	2.4	3.2	4.0	4.7
Yes	Delay (s)	142	138	136	137	145	152
	OCT (s)	1.5	2.1	2.5	3.3	4.2	4.7

5. CONCLUSIONS

This paper presents a general methodology of genetic algorithm (GA) for real-time implementations in dynamic environments by integrating the concept of Receding Horizon Control (RHC). Some RHC practices in control engineering are introduced when designing this new GA, particularly how to choose receding horizon and terminal penalty. Two case studies are reported, which demonstrate how to effectively design an RHC based GA, and further show the computational efficiency and robust performance of the RHC based GA when it is applied in dynamic environments.

6. ACKNOWLEDGEMENTS

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

REFERENCES

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.

Clarke, D.W., *Advances in Model-based Predictive Control*, Oxford University Press, 1994.

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.

Fogel, D.B., Evolutionary Computing: The Fossile Record, IEEE Press, Piscataway, NJ, 1998.

Goldberg, E., Genetic Algorithms in Search Optimization & Machine Learning. Reading, MA: Addison-Wesley, 1989.

Hu, X.B., S.F. Wu and J. Jiang, "Online free-flight path optimization based on improved genetic algorithm", *Engineering Applications of Artificial Intelligence*, vol.17, no.8, pp. 897-907, 2004

Hu, X.B. and W.H. Chen, "Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling", *Engineering Applications of Artificial Intelligence*, in press, 2005.

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

Onnen, C., R. Babuska, U. Kaymak, J.M. Sousa, H.B. Verbruggen and R. Isermann, "Genetic algorithms for optimization in predictive control", *Control Engineering Practice*, vol.5, no.10, pp. 1363-1372, 1997.

Sarimveis, H and G. Bafas, "Fuzzy model predictive control of non-linear processes using genetic algorithms", *Fuzzy Sets and Systems*, vol.139, no.1, pp. 59-80, 2003.

Sharma, S.K., S.F. McLoone, and G.W. Irwin, "Genetic algorithms for simultaneous identification of local operating regimes and local controller network design", Proceedings of Control 2004, 6-9, Sep, 2004, Bath UK.