Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

Solving Dynamic Single-Runway Aircraft Landing Problems With Extremal Optimisation

Irene Moser and Tim Hendtlass

Abstract—**A dynamic implementation of the single-runway aircraft landing problem was chosen for experiments designed to investigate the adaptive capabilities of Extremal Optimisation. As part of the problem space is unimodal, we developed a deterministic algorithm which optimises the time lines of the permutations found by the EO solver. To assess our results, we experimented on known problem instances for which benchmark solutions exist. The nature and difficulty of the instances used were assessed to discuss the quality of results obtained by the solver. Compared to the benchmark results available, our approach was highly competitive.**

I. Introduction

EXTREMAL Optimisation (EO) is a recent addition to the available range of stochastic solvers. In its current form, it was first described in [7]. While similar to a Genetic Algorithm, it only uses a single solution, consisting of multiple components, which are assigned individual fitness values. The solution mutates through the search space using a power law distribution to determine its next move from a neighbourhood of candidate solutions ordered by fitness. Best solutions, when found, are recorded for reporting, but do not influence the development of the working solution.

The dynamic aircraft landing problem was chosen as an example to further explore EO's capabilities to solve dynamic problems. Existing problems from the OR library in [2] were to be used as known benchmarks with comparable implementations and results. After some initial examination of the problem instances it became clear that to choose the most promising instances for our examples, some analysis of the available data would be necessary.

II. Aircraft Landing

The aircraft landing problem used in these experiments has been described in detail in [4], its dynamic implementation in [3]. Aircraft appear at a given appearance time A, which is when they report to the ATC for landing. All P aircraft have a range (earliest E to latest L landing time) within which they may land and a target T between E and L as an optimal landing time. Penalties apply for landing aircraft before or after the target time. Aircraft landing after other aircraft have to wait until a given, type-dependent separation time has elapsed. The optimisation goal is a) to find a feasible solution that lands all aircraft within their landing ranges (between the earliest and latest landing times) and b) to find the feasible solution which minimises the penalties for early/late (compared to target) landings. The objective function to optimise as formalised in [4] is given in equation (1).

$$\min \sum_{i=1}^{P} \left(g_i \alpha_i + h_i \beta_i \right) \tag{1}$$

where P is the number of aircraft
g is the penalty for landing early
h is the penalty for landing late
α is the amount of time the aircraft is early
β is the amount of time the aircraft is late.

In the dynamic case, the solving process follows a time line measured as the CPU time used by the program. The aircraft are made available to the *solver* at their given appearance times and remain in the *active window* of the solver until the landing time found by the solver is too close to current time to make any further change (i.e. within *freeze time*). In the dynamic case, the solver solves a sequence of smaller problem instances instead of the overall problem of all aircraft in the static instance.

III. Known Approaches

There are only two known dynamic approaches to the ALP, one based on the instances in [2], one working on actual data. Beasley et al. [3] apply two variations of a GA as well as a deterministic linear programming implementation to the time windows as implemented in this work. The instances airland9 - 13 were first devised for these experiments.

Ciesielski et al. [9] use two different GA adaptations to solve the problem dynamically, applying a sliding time window of three minutes each. For the experiments, two actual (past) schedules with 28 (29) aircraft over 37 (38) minutes from Sydney airport, incorporating the use of both its runways, have been used.

There are a few approaches which solve the instances from [2] in a static context. Ernst et al. [1] introduce a simplex-based approach for the scheduling part of the problem. The sequentially ordered aircraft are divided into trees where only the root node of each tree is allowed to land on target, similar to the scheduling model used in this work. The permutation part of the problem is solved using a GA,

Manuscript received November 3, 2006

I. Moser is a PhD student in the Faculty of Information and Communication Technology at Swinburne University, Melbourne, Australia (e-mail: imoser@ict.swin.edu.au).

T. Hendtlass is a Professor in the Faculty of Information and Communication Technology at Swinburne University, Melbourne, Australia (e-mail: thendtlass@swin.edu.au).

encoding the individuals as a permutation of all aircraft. The authors compare this approach with a Simplex, which is more time-consuming but has the advantage of providing accurate results.

An ACS (Ant Colony System) approach was proposed by Randall [14]. As ACS is a constructive approach, solutions are built from random sequences observing landing ranges and separation times. The pheromone model divides the landing ranges into four different parts. Experiments on airland1 – airland5 and airland8 (listed as airland6) show that the approach does not solve the problems to the same quality as [4].

Pinol et al. [13] propose to use two techniques they call Scatter Search and Bionomic Algorithms on the problems from [2], which are in essence GAs with special features. Both algorithms use a local search technique to optimise the new individual's time line. The benchmark solver used is a local search not discussed in detail, which finds the optimum reliably but at the cost of a runtime exponential in the number of aircraft.

Wen [15] uses Beasley’s integer linear programming formulation [4] to create a set partitioning problem in which runways are represented as columns and aircraft as rows. The column generation follows the Branch-and-Price method which is designed to simplify linear programs with large numbers of variables. A separate deterministic scheduler is used to optimise the time line of a given permutation. The approach is reported to outperform the linear program from Beasley et al. [4]

IV. Methodology

A. Implementation

The aircraft landing optimisation process consists of two parts, the search for the best permutation and the optimisation of the landing times given a permutation. The latter part of the solving process, as has been observed in [4], [3], [13] and [15] can be solved by a deterministic algorithm. Therefore we developed a deterministic hill climbing algorithm for optimising the landing times. The stochastic EO solver’s task is to find the optimal permutation of the aircraft in the active window.

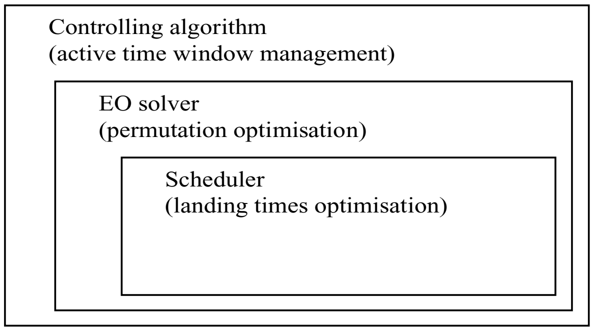
Controlling algorithm
(active time window management)
EO solver
(permutation optimisation)
Scheduler
(landing times optimisation)

Fig. 1 Parts of which our algorithm is composed.

A separate algorithm is needed to manage the active window and the time line. The *controlling algorithm* for the dynamic process has been implemented as follows:

1. At the start of the program, the CPU time is stored as the start time of the algorithm.
2. After each iteration of the algorithm, the start time is subtracted from the current CPU time. The result is compared to the appearance times of aircraft that are yet to appear and to the landing times of planes in the currently active solver window. If the landing time of an aircraft is within freeze time, it is removed from the active solver window.
3. If aircraft have appeared, they are added in order of their target times and the preliminary schedule of all active aircraft are adapted according to the minimum separation times.
4. The active aircraft are optimised as described in section B.
5. The algorithm stops when all aircraft have landed or when the CPU time has exceeded the latest landing time of the last aircraft, whichever comes first.

B. Extremal Optimisation

Unlike Genetic Algorithms, which work on a population of solutions, EO improves a single solution using mutation. A solution consists of multiple components which are assigned individual fitness values. The initial implementation proposed by Boettcher and Percus [8] only accepted the component with the worst fitness to be mutated and replaced by a random component. This method proved inefficient in that it was prone to entrapment in local optima.

The same authors later [7] introduced a power-law-based probabilistic choice of component for mutation. The solution components are ranked according to their fitness values, using a rank of 1 for the best quality component and K for the worst. A candidate component is chosen for mutation and confirmed with a probability of $k^{-\tau}$ where k is the fitness rank and τ is a small value between 1 and 10.

As these approaches solved problems with binary component representations, once the component had been chosen according to the rank-based distribution scheme, there was no doubt as to the type of change. The ALP is modelled as a permutation problem in this work. Therefore, instead of ranking the components of the existing solution by their fitness, a number of candidate mutations – each a copy of the current solution with a small change – is chosen and ranked by quality. This is referred to as the *neighbourhood*.

The performance of the EO algorithm depends on the type of change made to the current solution to define the neighbourhood, and on the only free parameter τ. It has been shown by Boettcher and Percus in [6] and other works, that the optimal value for τ is determined by $1+[\ln(K)]^{-1}$, where K is the number of neighbours to choose from. Some experiments with the current problem showed that good results will be achieved in the range 1.6-1.8, which coincides with the assumption that the active time window is, on average, a small number ≈ 5. As the value varies slightly for each problem instance, and the difference in quality is not large, we selected 1.6 over the whole series of experiments.

Similarly to the approach to solving a Travelling Salesperson Problem [7], where 2-opt moves were applied, our neighbourhood is defined as a single position swap, i.e. pairs of adjacent aircraft change places. Hence the length of the candidate sequence equals the length of the active time window. The candidates are then submitted to the scheduling algorithm to find their optimal cost which determines the rank k of each candidate.

Refining step 4 of the controlling algorithm described in section A, the steps of the EO solver are as follows:

1. Receive target-ordered sequence of active aircraft of length K and use it as a working solution.
2. Build a neighbourhood of K candidate solutions by swapping one pair of aircraft at a time.
3. Submit candidates to scheduler for optimisation of landing times and establish cost of optimised schedule.
4. Rank candidate solutions ascending by cost.
5. Choose one candidate according to $k^{-\tau}$ and adopt as new working solution unconditionally.
6. If the working solution is better than the existing schedule (either the preliminary one set by the controlling algorithm or subsequent improvements made in this step), adopt the landing times of the working solution.
7. Let the controlling algorithm check for update of active window. If changes have occurred, resume from 1. If not, resume from 2.

Note that, as the current solution continues to evolve, best solutions - when they are found - are simply set aside for reporting. The current working solution may be many steps from any recorded best solution.

C. Scheduler

Our implementation is based on the observation that the permutation at hand can be split up into sequences determined by equation (2), which can then be scheduled separately.

$$T_m + \sum_{i=m+1}^{n} \max\{S_{ij} \mid m \le j < i\} \ge T_n \qquad (2)$$

where $n > m$; $m \ge 0$; $n \le P$;
T is the target time of an aircraft,
P is the total number of aircraft and
S_{ij} is the minimum separation time between aircraft *i* and each preceding aircraft *j*

Note that equation (2) enforces that for each plane, the separation time to all its predecessors, not only the immediately preceding plane, must hold. If equation (2) holds for all aircraft between *m* and *n*, we regard the sequence between *m* and *n* as an interdependent chain of planes that must be optimised in combination.

The landing times of the aircraft in such an interdependent chain can be regarded as linked by the minimum separation times. They can be changed only in unison. To find out whether moving the sequence to an earlier landing is beneficial, we add one penalty weight (as if landing each plane one time unit earlier) for each aircraft to the score. Each aircraft which is currently scheduled with a landing time after target, adds a negative *h* to the score (moving it to an earlier landing would reduce the cost) and each aircraft whose landing time is scheduled on or before target, adds a positive *g* (as this aircraft would add to the cost if landed earlier).

The mirror image of this test is carried out for a move to a later landing time. Obviously, only one of these moves can have a negative result and thus reduce the cost. If neither move is beneficial, the optimum has already been achieved. If there is a benefit to either move, the possible length of the shift has to be explored. In our example with the left shift, possible limits are

a) the scheduled landing time of the last plane of an earlier chain and the separation times between the planes;
b) the earliest possible landing time of each craft in the sequence;
c) the target times of the planes currently landing later than their target. This is because they no longer add a negative late penalty to the score once the target time is reached.

If the possible shift range is zero and the boundary is the earlier chain, the benefits of uniting the chains and shifting them together has to be explored.

Similarly, if the benefit of both shifts is zero, we have to explore if one or more of the first aircraft in a chain can be shifted to an earlier landing with benefit while leaving the latter planes in place, or if one or more of the later planes can be shifted to a later landing with benefit if the earlier planes are left at their current landing times. This would indicate that the chain has to be split.

The exact steps of the scheduling algorithm, used at step 3 of the EO algorithm, are as follows:

1. Pre-schedule all active aircraft by assigning each aircraft whose landing time t_j is not influenced by the separation time from a preceding aircraft to its target time T_j. If the separation times from previous aircraft prevent assigning the target time, assign the first possible landing time (minimum separation time).
2. Divide all aircraft in the active window into *x* separate chains $s_1..s_x$ of aircraft whose landing times influence each other according to equation (2).

TABLE I
PROPERTIES OF PROBLEM INSTANCES

Problem instance	Number of aircraft	Run time per 10 aircraft*	Cost for target time order**	Active window length	Max (mean) sequence***	Number of moves****	Longest move*****
airland1	10	744	700	9	7 (5)	0 (0)	0
airland2	15	543	1520	10	7 (5)	2 (1.3)	2
airland3	20	485	1730	11	7 (3)	1 (0.5)	1
airland4	20	420	2540	11	15 (15)	0 (0)	0
airland5	20	465	5940	10	15 (15)	3 (1.5)	4
airland6	30	1089	N/A	6	27 (15)	Unknown	Unknown
airland7	44	1148	N/A	4	36 (11)	Unknown	Unknown
airland8	50	226	2470	18	6 (2.3)	3 (0.6)	3
airland9	100	1412	10690	11	17 (2.8)	13 (1.3)	5
airland10	150	1383	28868	15	25 (3.5)	27 (1.8)	8
airland11	200	1290	23973	12	14 (3.8)	32 (1.6)	10
airland12	250	1225	29091	13	23 (3.4)	36(1.4)	10
airland13	500	1122	69935	17	41 (4.5)	78 (1.5)	7

*Proportion of the time line (appearance of earliest – end of landing range of latest plane) available to 10 successive craft on average
**Cost of solution when optimal schedule is calculated for airplanes ordered by target times
***Maximum number of planes having no more distance between them than the minimum separation times. Average sequence length in brackets.
****How many aircraft are not between their nearest neighbours in target order, counted as the smallest number necessary to restore original order; i.e. 1,2,6,3,4 count as one (for 6) not 2 (for 3 and 4); in brackets number of moves per 10 aircraft
*****Number of aircraft with bigger (smaller) target landing times which are between this craft and its previous (successive) neighbour in target order.
Airland6 and airland7 cannot be solved in a dynamic context for lack of sufficient active window length (1 aircraft for most time windows). Their target-ordered sequence cannot be optimised as landing time decisions have to be made without knowledge of subsequent aircraft.

3. While there are chains s_i whose landing times can be further optimised, for each s_i check the benefit of
   i. shifting all aircraft in s_i to an earlier landing time by means of adding an early penalty *g* for all aircraft with $t_j \le T_j$ and a negative late penalty -*h* for each aircraft landing later than the target time, i.e. $t_j > T_j$
   ii. deferring the landing of all aircraft in s_i by adding *h* for each aircraft with $t_j \ge T_j$ and -*g* for all aircraft with $t_j < T_j$.
4. If none of above checks returns a negative cost (an improvement), go to 7. If there is a gain in either (both is not possible), find the maximal displacement
   i. for a shift to earlier landing; the distance to the minimum of the following:
      a) the earliest landing according to the minimum separation times from planes in the preceding sequences $\{s_k \mid k < i\}$, if they exist;
      b) the earliest landing according to the minimum separation times from already landed aircraft, if they exist;
      c) $\min\{t_j - E_j \mid \forall t_j \in s_i; t_j \le T_j\}$, where E_j is the earliest landing time of aircraft j;
      d) $\min\{t_j - T_j \mid \forall t_j \in s_i; t_j > T_j\}$
   ii. for a shift to a later landing; the distance to the minimum of the following:
      a) the minimum distance to the landing times of the planes of the following sequences $\{s_l \mid l > i\}$, if they exist;
      b) $\min\{L_j - t_j \mid \forall t_j \in s_i; t_j \ge T_j\}$, where L_j is the latest landing time of aircraft j
      c) $\min\{T_j - t_j \mid \forall t_j \in s_i; t_j < T_j\}$
5. If either of the shifts (4.i or 4.ii) is beneficial and the maximum possible displacement is nonzero, shift the landing times by the maximum displacement and go to 3. If one of the shifts is beneficial, but the maximum displacement is 0, go to 6.
6. Check if the distance to the aircraft of the previous (following) sequence is 0. If yes, join the chains and go to 3.
7. Check if it is beneficial to split a sequence by either
   i. shifting the later planes in a sequence to a later landing time while leaving one or more of the first in place;
   ii. shifting the earlier planes of a sequence to an earlier landing time while keeping the current landing times of one or more later planes in the sequence.

8. If none of these checks is successful, set the "optimised" flag of the sequence to true and continue with the next sequence until all sequences are optimised.

V. Experiments

To select suitable instances from the available datasets, we analysed the best solutions obtained from preliminary experiments, which yielded the best-known costs for most of the smaller problems. The results of the analysis are listed in Table I.

It is apparent that solving the first 7 instances would not improve our knowledge of the solver's capabilities considerably. Problems 6 and 7 are, due to the scarce number of aircraft in the active window (more than half of the time windows has only one aircraft), not suited for combinatorial optimisation in the dynamic case. Problems 1 and 4 do not require any changes to the target time ordering and make the EO solver redundant.

Problems 2, 3 and 5 are possible candidates, but since they only require minor moves (changing the position of very few craft by one or two positions) and there is no real sequence of numerous active windows of interesting length, we decided to experiment only on instances 8 – 13.

Airland8 has a rather short timeline of 226 seconds per 10 aircraft (Table I), an adequate challenge in the given environment. Problems 9 – 13 average between 1122 and 1412 seconds for 10 aircraft. To speed up the timeline and enhance the challenge, we ran them at 3 and 5 times the speed. Due to insignificant differences in the results, we only list the results of the fastest runs.

As in the benchmark, we observed the minimum separation time between all pairs of aircraft, not only adjacent planes.

VI. Benchmarks

The benchmarks used have been described in [3] and previous works of the same authors. DALP-H1 and DALP-H2 are hybrid approaches based on a stochastic algorithm and a deterministic schedule optimiser implemented as a primal-dual simplex algorithm in [1] and [11] which splits up interdependent sequences into a tree structure.

The stochastic parts of both DALP-H1 and DALP-H2 were implemented as Genetic Algorithms with a constructive preprocessing stage. Both rely on target-time ordering for their primary solution structure. For DALP-H1, the basic algorithm from [1] was adapted to the dynamic case by adding new aircraft by target times to the present aircraft which are ordered by the landing times the solver has found before.

DALP-H2 was first described in [5] as using a genetic algorithm with real-valued normalised encoding of the landing time as a position on the landing range. Features that adapt it to the dynamic environment have been added in [3] to ascertain that the range of the active time window is not exceeded. The extension made in [3] adds good individuals from runs with DALP-H1 as well as individuals obtained by orderings according to earliest/latest/target times.

VII. Results

We compare our best results, obtained from the experiments using EO on the chosen instances, with the results from [3] in Table II.

TABLE II
Results of Experiments, Comparison with Existing Benchmark

Problem Instance	DALP-OPT	DALP-H1	DALP-H2	EO Result
airland8	**2000**	2915	2710	2320
airland9	7848	13555	12554	**6110**
airland10	17726	31945	31034	**16386**
airland11	19327	27417	23963	**14559**
airland12	25049	34246	31440	**19342**
airland13	58393	78008	58392	**44708**

The results are the minima found over the trials, best results in bold.
DALP-OPT, DALP-H1 and DALP-H2 are algorithms for which results have been listed for the single runway case of these instances in [2].
All results have been rounded to the nearest integer.

More detailed results of the trials are shown in Table III. Each experiment was repeated 100 times. As definite best results exist only for the first 8 problem instances, we list our results as percentages of the results of DALP-OPT, the deterministic solver used as a benchmark for the dynamic case in [3], mentioned in Table II.

TABLE III
Results of Experiments with EO on Selected Problem Instances

Problem Instance	Min	Avg	Standard dev.
airland8	116	117	1.4
airland9	77	77	0
airland10	92	96	1.5
airland11	75	75	0
airland12	77	77	0
airland13	76	77	0.2

The best, mean and standard deviation of 100 trials are given as a percentage of the best solution of "DALP-OPT" in Table II.

Whether the DALP-OPT results are optimal is an interesting question. A cross-check with EO on the smaller problems, for which known optimal static results exist, proved that it is possible to solve these problems to a quality that is closer (airland4, airland5, airland8) or equal (airland1, airland2, airland3) to the static best-known result even in a dynamic environment. However, given the dynamic nature of the problem optimisation results of previous time windows influence the optimisation of subsequent active windows.

VIII. Discussion

The test results show that the algorithm solves the problems very quickly and to a very good result. Prolonged run times do not produce better results. There is, however, very little variation in the results for most of the problems. This may either indicate that the solver is reduced to finding a certain local minimum, or corroborate the theory that the problem is not very demanding.

$$\sum_{i=1}^{Z} z_i! \qquad (3)$$

where $Z \le P$ is the number of sequences of interdependent aircraft and $z_1 = n_1 - m_1 ... z_Z = n_Z - m_Z$.

The complexity of the permutation problem, even in the static case, is limited by equation (3). If the complete sequence does not fit into a time window, the complexity is further restricted (and, as an aside, it becomes virtually impossible to obtain results of static case quality in the dynamic case). To shed some light on this question, we listed the active window length and the lengths of interdependent sequences in Table I. For each interdependent sequence, we explored whether all its aircraft were in the active solver window at once. This was the case for all sequences in instances airland1 – airland3. In all the other problems, the coincidence of the interdependent sequences of aircraft with the active windows was at most 50%.

As Table I shows, the average number of moves that will create the optimal solution starting from a target-time-ordered permutation is around 2 per 10 aircraft at most for all the problem instances in the library. The vast majority of these moves is made by swapping the positions of two neighbouring aircraft (counted as a single move). Airland10 and airland13 are the only "interesting" instances; the best solution we found has longer average moves as well as small areas where many moves are intertwined and the underlying target-time ordering is barely visible.

The range of shifts has also been reduced for practical reasons, as first implemented in [10] and more recently in [1]. It is still a reliable measure, since target time ordering has been used as a starting point for permutations by the stochastic solvers both in [3] and the work presented here. It seems plausible that an incremental algorithm with single step moves can outperform a Genetic Algorithm with crossover and mutation operators, as they may prove "overkill".

The values obtained by our solver are therefore very consistent with our analysis. The best results found by EO (listed in Table III) outperform the benchmark results by a factor of up to 2.2.

It seem surprising that our stochastic algorithm has been able to outperform the deterministic benchmark solver DALP-OPT. As we are working in a dynamic environment, deterministic solvers can be expected to produce the optimal result within the local time window. However, this may lead to a suboptimal global result.

IX. Further Work

The current implementation can easily be adapted to solve a problem with multiple runways. In the case of the given problem instances, this does not seem to pose a great challenge to our solver. To experiment further with the capabilities of a stochastic solver, the density of aircraft would have to be adjusted to the number of runways to produce an adequately dense and interesting problem.

Future experiments will be directed at the question of whether the EO neighbourhood would improve by including moves with variable lengths (covering more than one position). In combination with more complex problem instances, this variation can be expected to produce interesting results.

As the optimal value of τ depends on the length of the candidate list, and the candidate list length fluctuates with the length of the active solver window, experimenting with an adaptible value for τ is also one of our priorities.

As there is only one existing benchmarch result available to these instances, and it is likely that this type of permutation problem would lend itself as an ideal problem to be solved by Ant Colony Optimisation, trials with ACO on these problem instances, and more difficult instances are likely to produce excellent results.

References

[1] H. Balakrishnan and B. Chandran, "Scheduling Aircraft Landings under Constrained Position Shifting," AIAA Guidance, Navigation and Control Conference and Exhibit, 2006.

[2] J. Beasley, OR-library, "Distributing Test Problems by Electronic Mail," Journal of the Operational Research Society, pp. 1069-1072, 4/1990.

[3] J. Beasley, M. Krishnamoorthy, Y. Sharaiha and D. Abramson, "Displacement Problem and Dynamically Scheduling Aircraft Landings", Journal of the Operational Research Society, pp. 54-64, 55/2004.

[4] J. Beasley, M. Krishnamoorthy, Y. Sharaiha and D. Abramson, "Scheduling Aircraft Landings - the Static Case", Transportation Science, pp. 180-197, 34/2000.

[5] J. Beasley, J. Sonander and P. Havelock, "Scheduling aircraft landings at London Heathrow using a population heuristic," Journal of the Operational Research Society, pp. 483–493, 52/2001.

[6] S. Boettcher and A. G. Percus, "Extremal optimization: an evolutionary local-search algorithm" In: Proceedings of the 8th INFORMS Computing Society Conference, 2003.

[7] S. Boettcher and A. Percus, "Nature's Way of Optimizing," Artificial Intelligence, Vol. 119, Number 1, 2000, pp. 275-286.

[8] S. Boettcher and A. Percus, "Optimization with Extremal Dynamics," Physical Review Letters, Vol. 86, 2001, pp. 5211-5214

[9] V. Ciesielski, and P. Scerri, "An anytime algorithm for scheduling of aircraft landing times using genetic algorithms", Australian Journal of Intelligent Information Processing Systems, vol. 4, pp. 206-213, 1997.

[10] R.G. Dear, "The dynamic scheduling of aircraft in the near terminal area, " Report R76-9, Flight Transportation Laboratory, MIT, 1976.

[11] A. T. Ernst and M. Krishnamoorthy, "Algorithms for Scheduling Aircraft Landings," AGIFORS 2001.

[12] A.T. Ernst, M. Krishnamoorthy and R.H. Storer, "Heuristic and Exact Algorithms for Scheduling Aircraft Landings," Networks pp. 229-241, 34/1999.

[13] H. Pinol and J. Beasley, "Scatter search and bionomic algorithms for the aircraft landing problem", European Journal of Operational Research, vol. 171, 2006, pp. 439-462.

[14] M. Randall, "Scheduling Aircraft Landings with Ant Colony Optimisation", Proceedings of the International Conference Artificial Intelligence and Soft Computing, pp. 129-133, 2002.

[15] M. Wen, "Algorithms of Scheduling Aircraft Landing Problem", Master thesis, Department of Informatics and Mathematical Modelling, Technical University of Denmark, 2005.