Proceedings of the Twenty-Fourth Innovative Appications of Artificial Intelligence Conference

Local Search for Designing Noise-Minimal Rotorcraft Approach Trajectories

Robert A. Morris
NASA Ames Research Center, USA

K. Brent Venable
University of Padova, Italy

Marco Pegoraro
University of Padova, Italy

James Lindsey
Monterey Technologies, USA

Abstract

NASA and the international community are investing in the development of a commercial transportation infrastructure that includes the increased use of rotorcraft, specifically helicopters and civil tilt rotors. However, there is significant concern over the impact of noise on the communities surrounding the transportation facilities. One way to address the rotorcraft noise problem is by exploiting powerful search techniques coming from artificial intelligence coupled with simulation and field tests to design low-noise flight profiles which can be tested in simulation or through field tests. This paper investigates the use of simulation based on predictive physical models to facilitate the search for low-noise trajectories using *local search* combined with a robust noise simulator.

Introduction

There is considerable interest in developing a commercial transportation infrastructure that is based on an increased use of rotorcraft, specifically helicopters and aircraft such as a 40-passenger civil tilt rotor. Rotorcraft have a number of advantages over fixed wing aircraft, primarily in not requiring direct access to the primary fixed wing runways. As such they can operate at an airport without directly interfering with major air carrier and commuter aircraft operations. There is significant concern over the impact of noise on the communities surrounding the transportation facilities. One way to address the rotorcraft noise problem is to automatically design flight profiles which can be evaluated with respect to noise in simulation or through field tests.

The problem of designing low noise flight profiles can be viewed as a trajectory optimization problem (LaValle 2006). Informally, a trajectory optimization problem consists of a set of *states*, a vector of *control decisions*, a start and goal state, a cost function, and a set of constraints. A state represents locations (i.e. points in a 3D space), velocity and heading. A control decision is a vector representing change in velocity, altitude, heading, and in turn radius. The problem, which we adapt here, can, thus, be stated informally as follows: *given a set of states and control actions, find a sequence of actions (trajectory) that minimizes a cost function subject to a set of dynamic constraints, and constraints on start- or end-states.* Approximate approaches to optimization such as *local search* (Hoos and Stutzle 2004) have been shown to be particularly effective in finding solutions of good quality fast and in high-dimensional spaces. This paper reports on the use of local search in trajectory optimization. We assess the performance and effectiveness of local search using different aggregate evaluation functions.

Background

Noise simulation. Sound can be broken down into frequencies. The ear is more sensitive to mid- and high frequency sounds, so we find noise in these ranges more annoying. The so-called *A-weighting* function (Conner, Burley, and Smith 2006) approximates the sensitivity of the human ear and helps to assess the relative loudness of various sounds. Sound levels vary with time. To take exposure duration into account, the most common measure is the *Sound Exposure Level (SEL)*. *SEL* 'summarizes' the variable energy level of an event with arbitrary duration by mapping it to an event of one second duration with the same overall energy and a constant energy level. *SEL* provides a comprehensive way to describe noise events for use in modeling and comparing noise environments. Computer noise models base their computations on *SEL* values.

Helicopter noise sources include the main rotor, the tail rotor, the engine(s), and the drive systems. The most noticeable acoustical property of helicopters is referred to as BVI (Blade Vortex Interaction) noise. This impulsive noise occurs during high-speed forward flight as a result of blade thickness and compressible flow on the advancing blade. This causes the blades airloads to fluctuate rapidly and results in impulsive noise with shock waves that can propagate forward. At lower airspeeds, and typically during a descent, BVI can occur when a blade intersects its own vortex system or that of another blade (Cox et al. 2009).

The Rotorcraft Noise Model (RNM) (Conner, Burley, and Smith 2006) is a simulation program that predicts how the sound of a rotorcraft will propagate through the atmosphere and accumulate on the ground. RNM is capable of calculating cumulative noise exposures such as A-weighted SEL. Given a flight trajectory and other parameters describing the rotorcraft and the environment, RNM simulation produces predictive noise data in various formats. Of interest here is the generation of *ground noise contour plots*: a set of values

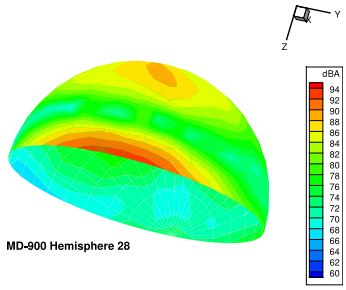

Figure 1: A sound hemisphere of an MD-900 helicopter.

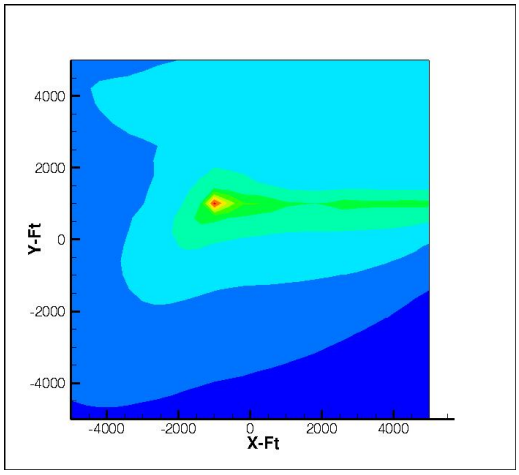

Figure 2: A Noise Contour Plot.

representing ground noise exposure using A-weighted SEL over a designated grid of x-y points around the evaluated trajectory. Figure 2 shows an example of such a plot, where each color corresponds to a dB level (redder and lighter colors noisier). These plots provide the data used to compute the aggregate cost functions used during search.

RNM combines a model of sound propagation through the atmosphere with a database of noise data either experimentally or analytically generated. The database is comprised of a set of *sound spheres*. Points on the sphere are described in terms of a radius from the source and two spherical angles. A sphere is associated with one noise source and one flight condition (flight path angle, nacelle angle (for tilt-rotors) and airspeed). Figure 1 shows an example sound hemisphere.

Local search. Local search (Hoos and Stutzle 2004) is one of the fundamental paradigms for solving computationally hard combinatorial problems. Given a problem instance, the basic idea underlying local search is to start from an initial search position in the space of all possible assignments (typically a randomly or heuristically generated assignment, which may be infeasible, sub-optimal or incomplete), and to improve iteratively this assignment by means of minor modifications. At each *search step* we move to a new assignment selected from a *local neighborhood*, chosen via a heuristic evaluation function. The evaluation function typically maps the current candidate solution to a real number and it is such that its global minima correspond to solutions of the given problem instance. The algorithm moves a the neighbor with the smallest value of the evaluation function. This process is iterated until a *termination criterion* is satisfied. The termination criterion is usually the fact that a solution is found or that a predetermined number of steps is reached. To ensure that the search process does not stagnate, most local search methods make use of random moves: at every step, with a certain probability a random move is performed rather than the usual move to the best neighbor.

Reasons for preferring local search for trajectory optimization include (1) *Anytime performance*, since, on average, local search behaves well in practice, yielding low-order polynomial running times (Hoos and Stutzle 2004); (2) *flexibility and ease of implementation*, as deployment-related deadlines suggest the use of techniques which are easy to implement; and (3) *simulator compatibility*, since, running RNM is heavy from a computational point of view. This last issue, in particular, means that the repetitive evaluation of partial trajectories, required by complete incremental solving paradigms (e.g. Branch and Bound), may be unacceptably time consuming.

In this paper we use a specific variant of local search known as hill-climbing. In hill-climbing search (Selman and Gomes 2006), we select any local change that improves the current value of the objective function.

The Trajectory Noise Optimization Problem

We focus on approach trajectories because that is virtually where all the community noise problems arise and the problem for take-off trajectories is very similar. We will focus on A-weighted SEL as our noise exposure metric for the reasons explained in the background section. RNM simulation provides a black box scoring function for candidate trajectories. Specifically, RNM produces an output file that assigns predicted noise for a set of ground points arranged in a two-dimensional grid on the X-Y plane. The grid size is defined in terms of the values of the corner nodes and the distance between nodes.

Upon this grid our model superimposes an organization of *nodes* associated with the state of the aircraft and the control decisions being made by the pilot. We introduce state variables X, Y, Z, V, H and associated domains for, respectively, location (X, Y), altitude (Z), airspeed (V), and heading (H). We use normal conventions for heading, whereby 0 is north, 90 is east, 180, south, and 270 west. Given a state variable Q we write q, to refer to domain elements of the variable. A *state* of the system is a 5-tuple $s = \langle x, y, z, v, h \rangle$.

Similarly, we introduce decision variables $\Delta V, \Delta Z, \Delta H, \Delta R$ for change in velocity, change in altitude, change in heading, and change in turn radius, also with associated domains, and we write Δv to denote a value in the domain of ΔV, etc. Change in heading involves addition modulo 360. Change in radius involves one action to initiate the change (e.g. $\Delta R = 180$ to start a 180 degree radius turn) and a complementary action to come out of the turn (e.g. $\Delta R = -180$ to restore straight flight). A decision

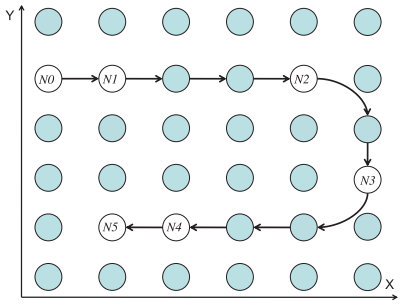

Figure 3: A "box"-like approach pattern.

vector (or simply decision) is a tuple d of values for each decision variable.

A *node* is a pair $\langle s, d\rangle$ of a state and decision, representing the state of the rotorcraft when the pilot or automated system begins to apply decision d. Given node $N_i = \langle s_i, d_i\rangle$, we will denote with $\langle x_i, y_i, z_i, v_i, h_i\rangle$ and $\langle \Delta v_i, \Delta z_i, \Delta h_i, \Delta r_i\rangle$ its components.

A path (trajectory) is a sequence of k nodes. Between two adjacent nodes N_j, N_{j+1} there is an edge labeled with the distance flown $dist_j$ (in feet), between the locations corresponding to the nodes. For a turn, it measures the portion of the circumference of the circle flown. A *consistent* path is one in which, for all $j = 1 \ldots k-1$, node N_{j+1} is the result of applying d_j at s_j for the entire length $dist_j$. We express this as a transition function $T : N \rightarrow N$, where N is the set of nodes.

We assume we are given two nodes designated as start and finish, with fixed state and control vectors and that a solution is any consistent flyable trajectory between them. To control the size of this space we initially start by limiting the paths to those that would be considered 'standard' by pilots. One example of a standard approach is a box pattern, as the one shown in Figure 3. This trajectory is represented by 6 nodes, $N_0 \ldots N_5$, where two 90^o turns start, respectively, at N_2 and N_3. The goal is to find an assignment $(s_0, \ldots, s_5, d_0, \ldots, d_5)$ to the state and control vectors of the nodes not fixed by initial and final conditions, such that the noise simulated by RNM on the corresponding trajectory is minimal.

All the trajectories considered in this work are either hand crafted by pilots and or randomly generated by a program designed to generate only standard trajectories.

Flyability Constraints

Conditions that make a trajectory suitable to fly are usually expressed in terms of constraints over the glide slope angle and deceleration. In particular, any part of a trajectory should be characterized by an angle of descent $\gamma \in [0^o, 12^o]$ and a deceleration $a \in [0g, 0.1g]$ (or $a \in [40ft/sec^2, 201ft/sec^2]$). Such restrictions induce constraints on the change of velocity and altitude as follows. Given a pair of nodes N_i, N_j and a path between them of distance $dist_{ij}$ we have:

- the deceleration constraint (dec): $\Delta v_i \in \{\delta_v \| \exists a \in [0, 0.1], \delta_v = \sqrt{v_i^2 + 2a \times dist_{ij}} - v_i\}$, where a is expressed in gs.
- the angle-of-decent constraint (aod): $\Delta z_i \in \{\delta_z \| \exists \gamma \in [0^o, 12^o], tan(\gamma) = \frac{\delta_z}{dist_{ij}}\}$.

In addition, there are a minimal velocity and altitude (v_{mini}, z_{mini}) that a rotorcraft must have when starting the final part of the approach (that is at the, so called, landing decision point). Such values are a function of the distance of the landing decision point from the landing site. A trajectory is said to be *flyable* if it satisfies all the deceleration and angle-of-decent constraints along its path, and does not violate the bounds defined by v_{mini} and z_{mini}.

A *Trajectory Noise Optimization Problem* (TNOP) is a tuple $\langle S, D, s_0, s_f, aod, dec, v_{mini}, z_{mini}\rangle$, where S is a set of states, D s set of decisions, s_0, s_f are initial and final states, aod, dec are deceleration and descent angle constraints, and v_{mini} and z_{mini} are as just defined. A feasible solution to a TNOP is a path $P = N_0, N_1, \ldots, N_k$ where $N_0 = \langle s_0, d_0\rangle$, $N_k = \langle s_f, 0_d\rangle$, where 0_d represents the decision of leaving everything unchanged, and for all $j = 2 \ldots k$, $N_j = T(N_{j-1})$, and where P satisfies the flyability constraints.

Cost Functions

We introduce two natural ways of 'aggregating' RNM contour noise data into scalar valued functions. One cost function identifies ranges of values that correspond to various levels 'high', 'medium' and 'low' noise, and creates 'bins' that store the number of grid noise data points in that range. Each bin is assigned a weight indicating its importance in determining solution quality, and the trajectory is evaluated as the weighted sum of the bin values.

Formally, we define a *Binning Heuristic function* (Bin) as follows. Given in input a solution t, RNM computes the A-weighted SEL value for each of the grid points. Let us denote with $SEL(t, x, y)$ such a value for the grid point (x, y) given trajectory t. We define a sequence of decreasing ranges, $\langle r_1, r_2, \ldots, r_n\rangle$ partitioning the SEL values of the grid points. Given a trajectory t let us denote by $S_i(t) = \{(x, y) | SEL(t, x, y) \in r_i\}$. We define the following vector $b(t) = \langle b_1(t), b_2(t), \ldots, b_n(t)\rangle$ where $b_i(t) = |S_i(t)|$. The bin-score of solution t is $Bin(t) = \Sigma_{i=1\ldots n} w_i b_i(t)$ where w_i is the weight associated to the i-th bin, $w_i > w_{i+1}$ and $\Sigma_{i=1,\ldots,n} w_i = 1$. Depending on how the weights are set this function can be made to favor either trajectories that assign lower levels of noise to larger regions or trajectories that avoid, even small, extremely noisy regions. The goal will be that of minimizing the Bin value.

The other cost function is based on ordering two candidate solutions based on a notion of 'significant difference' in their predicted noise values. One noise data point is significantly different from another if the human ear can detect a change in the noise. Counting the number of significantly different pair of noise values for the same point between two solutions, we can generate a partial ordering of the candidates.

Formally, we define a *Significant Improvement Heuristic function* $(Diff)$ as follows. Let s denote a reference solution and t another solution. Then the significant improve-

ment score of t w.r.t. s is

$$Diff(s,t) = |\{(x,y) \| SEL(t,x,y) - SEL(s,x,y) \geq 1.5dB\}| - |\{(x,y) \| SEL(s,x,y) - SEL(t,x,y) \geq 1.5dB\}|.$$

In other words, this heuristic function considers a reference solution (that, in our case will be seed solution of the local search), and then scores all other solutions counting the number of grid points where they produce a noise that is at least 1.5dB lower than the one produced by s at the same point. A 1.5dB threshold has been identified to be the smallest improvement that can be perceived by a human. The intuition behind this heuristic function is that of promoting solutions that improve significantly in the largest number of grid points. Given this heuristic function the goal is to minimize its value.

Local Search for solving TNOPs

The technique we propose here to solve the optimization problem described in the previous section is a hill-climbing local search approach. Figure 4 describes the pseudocode of our algorithm, which we call *Box-TNOP-HC*.

Box-TNOP-HC(Trajectory σ_{seed}, function $score$, integer $threshold$)
$\sigma_{cur} = \sigma_{seed}$ // current trajectory
$\sigma_{best} = \sigma_{seed}$ // best incumbent trajectory
$step = 1$
do
  σ_0 = Neighbor(σ_{cur})
  $neighborhood(\sigma_{cur}) = neighborhood(\sigma_{cur}) \setminus \{\sigma_0\}$
  while $neighborhood(\sigma_{cur}) \neq \emptyset$ and $score(\sigma_0) \geq score(\sigma_{cur})$
    σ_0 = Neighbor(σ_{cur})
    $neighborhood(\sigma_{cur}) = neighborhood(\sigma_{cur}) \setminus \{\sigma_0\}$
  $\sigma_{cur} = \sigma_0$
  if $flyable(\sigma_{cur})$ and $score(\sigma_{cur}) < score(\sigma_{best})$
    $\sigma_{best} = \sigma_{cur}$
  $step++$
while $step \leq threshold$
return σ_{best}

Neighbor(Trajectory σ)
1 $n = random(\sigma)$ // randomly pick a node
2 $p = partner(n)$ // randomly choose partner for transfer
3 $select\ c \in \{\Delta v, \Delta z\}$ // randomly choose control variable
4 $v_c = val(c,p,n)$ // find an allowable value to transfer
5 $\sigma_n = transfer(n,p,v_c,\sigma)$ // add to n and subtract from p
6 $(n,p,c,v_c) = used$ // mark quadruple as used
return σ_n // return the neighbor

Figure 4: Algorithm Box-TNOP-HC.

The inputs to the algorithm are a seed solution σ_{seed}; a scoring function $score$; and a positive integer $threshold$, representing the number of search steps after which the execution must terminate. The output of *Box-TNOP-HC* is a solution denoted by σ_{best}. During the execution we keep track of the current solution, the neighborhood of which we are exploring, denoted by σ_{cur}, and the best flyable solution found so far, denoted with σ_{best}. Both such solutions are initially assigned the seed solution. Then, the algorithm starts exploring the neighborhood of σ_{cur}. As soon as it finds a solution that is better than the current one, it checks if it is flyable and if so it saves as the best incumbent. *Box-TNOP-HC* then updates σ_{cur} and starts scanning its neighborhood. Whenever no better solution is found, a random move in the neighborhood is taken.

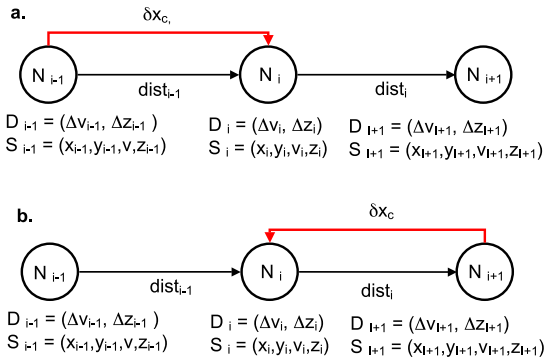

Figure 5: Transferring values between adjacent nodes.

Neighborhood Function

The neighbor of a trajectory s is the result of applying one of two operators that alter the change of speed or altitude (ΔV, ΔZ) at two adjacent nodes of s. Figure 5 illustrates the general case where a node has two adjacent nodes with which to swap values.

More specifically, a node N_i is chosen at random to be the recipient of the transferred value. A node adjacent to N_i (i.e., N_{i-1} or N_{i+1}, called the *partner*) and a control variable, ΔV or ΔZ, are also chosen randomly. An amount $0 < \delta x_c \leq \Delta x_{i'}$ is then computed and *transferred* to N_i; that is, δx_c is added to the appropriate control variable in N_i and subtracted from the value of the partner. Note that given a trajectory with L nodes, $N_1, \ldots N_L$, no transference is possible for the final node, N_L. The first and $L-1$st nodes have only one partner; the rest have two.

The transfer value v_c to transfer must be chosen in a way to preserve the feasibility of the new trajectory. Intuitively, there are two considerations: first, if too much value is transferred to a node, the trajectory will force the pilot to ether descend or decelerate too quickly during the segment beginning at N_i, violating the limit constraints on these values. Second, if too much control is passed *backward* from N_{i+1} to N_i, then more deceleration is applied sooner, and if too much is transferred earlier the helicopter might end up flying too low or too slow at N_{i+1}. This test involves the lower bound values v_{mini} and z_{mini} defined earlier for the state at the partner node.

Moreover, since there is an infinite number of choices of values for v_c and changes in the sound levels occur only for large enough transfers, we have decided to consider only a fixed set of such values corresponding to relevant percentages (i.e. 25%, 50%, 75%, and 100%) .

Finally, the effects of the transference of control is propagated to the states of the relevant nodes. Specifically, if control is transferred forward to N_i, then the state of N_i is

changed; if control is transferred backward to N_i, then the state of N_{i+1} changes.

Experiments

The goal of the experiments reported in this section are to assess the effectiveness and limitations of local search in solving the TNOP. We were interested in exploiting the different cost functions, as well as the variable data resolution capabilities of the simulator, in refining and improving local search. Specifically, we introduce 6 local search variants:

1. **Bin**: local search using Bin cost function with 7 bins corresponding to the SEL ranges $[125, -], [115, 124], [105, 114], [95, 104], [85, 94], [75, 84], [0, 74]$ with weights respectively $(0.2, 0.2, 0.2, 0.1, 0.1, 0.1, 0.1)$.
2. **Diff**: local search using Diff cost function.
3. **DiffBin**: two- phase local search: first, running Diff, then running Bin starting from the best solution found in the first phase;
4. **BinDiff**: same as previous, but using scoring functions in reversed order;
5. **HBin**: two-phase variable resolution search: first, running Bin with coarse resolution (1000 feet per grid element), followed by search using BIN with finer (200) resolution.
6. **HDiff**: same as the previous, but with Diff.

We conducted a number of experiments using the 6 different approaches defined above. Each of these variants was run 5 times, starting with a different randomly generated seed each time. The random generator generated paths that conformed to the 'box' pattern of Figure 3 with, possibly, a different location of each node except for the final node. The threshold value was set to 100 for Bin and Diff, 90 and 50 for the lower and higher resolution search, respectively, of HBin and HDiff, and 50 for each phase of BinDiff and DiffBin. The resolution (R) chosen for the reported experiments was 200 (i.e. 200 feet between each element in the grid), except for the coarse stages of $Hbin$ and $HDiff$, where, as noted, the resolution size was 1000.

Table 1 shows some results of our experiments. The first column contains the average over the 5 runs of the scores of the best solution. Recall that, given the way the cost functions are defined, lower is always better and that for $Diff$ the final score can be a negative integer while for Bin the score will be a positive natural number. The second column shows the absolute best score found in the 5 runs and the last column shows the average required time.

The relevant comparisons to make are among the two groups of rows segmented in the table returning, respectively, a Bin score and a $Diff$ score. Note first that applying $Diff$ first to the local search improves the outcomes on both average score and best (comparison of first and second rows), and the difference in time suggests that convergence to a better local optimum happens quicker. Using two different resolutions for Bin improves the best score, but not the average, suggesting that the hybrid resolution approach does allow for more exploration of the solution space. The reason for the increase in average time is most probably due to the RNM overhead caused by the resolution change.

In the case of the $Diff$ cost function, the results of multi-stage search are more promising. Dividing the work with Bin search providing the seed for $Diff$ search (i.e. $BinDiff$) shows better average score and best score, with lower time cost. The $HDiff$ approach, although again using more time, found the overall best score. These results, thus, indicate that improvements can be made with a multi-stage local search approach.

Table 1: Quality Comparison of 6 Approaches to Local Search Optimization, plus Random Walk Results.

R=200	Average	Best	Time(sec)
Bin^{100}	70	66	1060
$\text{DiffBin}^{50,50}$	67.2	57.2	813.8
$\text{HBin}^{90.75}$	73	58.7	1104.9
RW(Bin)^{100}	75.2	74.6	962.3
Diff^{100}	-723	-1097	1412.4
$\text{BinDiff}^{50,50}$	-932.6	-1313	1014.7
$\text{HDiff}^{90,75}$	-880	-1387	1566.5
RW(Diff)^{100}	-695	-729	960.4

To evaluate the effectiveness of local search, we ran a simple Random Walk (RW) procedure, whereby 100 flyable trajectories were randomly generated, scored, and the overall best preserved. The last two rows of the two groups of Figure 1 show these results. While RW is competitive with local search, the modest extra time required for developing local search results in better exploration of the solution space and higher quality trajectories. Moreover, it should be noted that the hybrid approach does always better than RW either faster or with an extremely limited increase in time.

If we consider an on-board use of the algorithm where the critical resource is time, a tradeoff between data resolution, which allows for better discrimination of noise quality, with search depth, which allows for more candidate trajectories to be evaluated emerges as an issue to be taken into account. In our experimental setting we converged on resolution =200 and threshold=100 as a reasonable tradeoff. The question then arises: could we have obtained significant improvement with a slight adjustment to threshold, or can we be reasonably certain that a 'plateau' has been reached in solution quality? Answering this question requires generating data that reveal aspects of the 'topology' of the solution space. In Figure 6 we show the runtime behavior for the Random Walk runs discussed earlier (using Bin; the behavior for Diff is similar). A dramatic improvement early on followed by smaller, gradual improvements as more of the search space is examined can be observed. This pattern, suggests that we could reduce the threshold (say, to 50) and increase the number of random restarts (say from 5 to 10). Or we could increase the resolution (say to 100) and again lower the threshold, increasing the discriminating power of the cost function while offsetting the cost of running RNM through reducing the depth of search. A similar trend has been observed also for the other functions which outperform

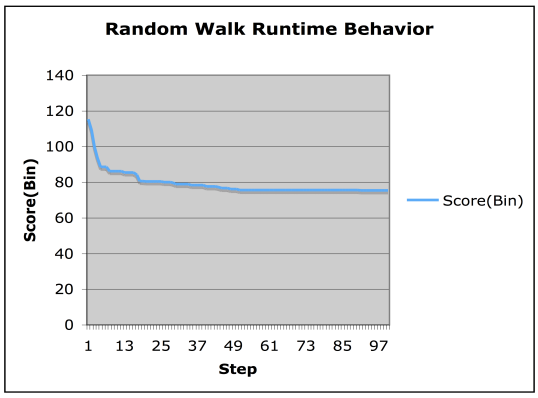

Figure 6: Runtime behavior of RW with Bin function.

RW also in terms of the reduced thresholds.

Related and Future Work

The field of trajectory optimization has a long history. In addition to noise, trajectories have been optimized with respect to time, fuel, path length and obstacle avoidance. Methods of solving trajectory optimization problems range from numerical methods (Betts 1998) to non-linear programming problems (Goplan et al. 2003) or dynamic programming (Hagelauer and Mora-Camino 1998). In addition, path planning methods from robot motion planning has been used (P. Cheng and LaValle 2001). Randomized optimization methods such as simulated annealing and genetic algorithms have also been applied in the work by Xue and Atkins (Xue and Atkins 2006). The latter bears the most similarity to the work described here, but has a number of important differences. There, the search space in modeled with a k-ary tree approach where each branch represents a change in the value of a parameter (e.g. path angle and acceleration) and the branching factor is restricted to at most k. We, instead, consider box-shaped trajectories, inspired by standard flying practices, which have a more restricted shape but yet cannot be modeled in the framework used in (Xue and Atkins 2006). Moreover, the noise produced by a trajectory is evaluated in (Xue and Atkins 2006) using a verified noise database, whereas we use RNM as an evaluation tool. Finally, the local search techniques employed are different, as we use a standard hill-climbing procedure whereas in (Xue and Atkins 2006) simulated annealing was used.

In this paper we have extended and revised the work presented in (Morris, Venable, and Lindsay 2012b; 2012a) in several ways. The box structure of the trajectories has been extended with turns and the box can now lie in any part of the grid only constrained by the landing site. The neighborhood function is fundamentally different as it allows to explore more than one transfer of deceleration and decent between two nodes and a random flyable (rather than a manually generated) trajectory is used as a seed. A new random walk algorithm exploring only flyable trajectories has been implemented and tested. The experimental results show a significant improvement both in terms of solution quality and performance[1] and we thus foresee our new approach as promising since the previous one has already been field tested with a positive outcome.

We are currently exploring trajectories that do not conform to the narrowly defined box pattern defined here. The resulting increase in dimensionality has led to the need for more robust modeling and search. We are now mapping TNOP into a 3D path planning problem to which we are applying sampling-based search methods. Future reports will document the results of this work.

We believe this line of research in also very promising in terms of advancing knowledge on the AI side. For example, this work has emphasized several important issues regarding the combination of local search techniques with cost functions derived from the output of sophisticated software such as a simulator.

References

Betts, J. T. 1998. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control and Dynamics* 21(2):193–207.

Conner, D. A.; Burley, C. L.; and Smith, C. D. 2006. Flight acoustic testing and data acquisition for the rotor noise model (rnm). In *Proc. of the 62nd Annual Forum of the American Helicopter Society*, 1–17.

Cox, C.; Schaaf, P.; Syms, R. A.; Tramontana, P.; Orozco, J.; Bennet, R.; Brieger, J.; and Jacobs, E. 2009. Fly neighborly guide. Technical report, Helicopter Association International.

Goplan, G.; Xue, M.; Atkins, E.; and Schmitz, F. H. 2003. Longitudinal-plane simultaneous non-interfering approach trajectory design for noise minimization. In *Proc. of the 59th AHS International Forum and Technology Display*, 1–18.

Hagelauer, P., and Mora-Camino, F. 1998. A soft dynamic programming approach for on-line aircraft 4d-trajectory optimization. *European Journal of Operational Research* 107:87–95.

Hoos, H. H., and Stutzle, T. 2004. *Stochastic Local Search: Foundations and Applications*. Elsevier - Morgan Kaufmann.

LaValle, S. 2006. *Planning Algorithms*. Cambridge University Press.

Morris, R.; Venable, K.; and Lindsay, J. 2012a. Automated design of noise-minimal, safe rotorcraft trajectories. In *Proc. of the 68th American Helicopter Society Annual Forum & Technology Display*.

Morris, R.; Venable, K.; and Lindsay, J. 2012b. Simulation to support local search in trajectory optimization planning. In *IEEE 2012 Aerospace Conference*.

P. Cheng, S. Z., and LaValle, S. M. 2001. rrt-based trajectory design for autonomous automobiles and spacecraft. *Archives of Control Sciences* 11(3-4):167–194.

Selman, B., and Gomes, C. 2006. Hill-climbing search. In *Encyclopedia of Cognitive Science*. John Wiley & Sons.

Xue, M., and Atkins, E. M. 2006. Terminal area trajectory optimization using simulated annealing. In *44th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: AIAA.

[1] Results omitted due to lack of space.