Simulation to Support Local Search in Trajectory Optimization Planning

Robert A. Morris
NASA Ames Research Center, CA (USA)
robert.a.morris@nasa.gov

K. Brent Venable
University of Padova, Italy
kvenable@math.unipd.it

James Lindsey
Monterey Technologies, CA)
NASA Ames Research Center(USA)
james.e.lindsey@nasa.gov

***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 a class of automated search algorithms called *local search*. A novel feature of this approach is the ability to incorporate constraints directly into the problem formulation that addresses passenger safety and comfort.**

TABLE OF CONTENTS

INTRODUCTION

The problem motivating this work is the design of low-noise approach trajectories for rotorcraft in order to reduce surrounding community noise. This is an important component in developing a 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]. The search space for such a problem is the space of all possible trajectories, which is typically characterized by a high dimensional model describing constraints on the position, dynamics, and kinematics of the rotorcraft. Approximate approaches such as *local search* have been shown to be particularly effective in finding solutions of good quality fast and in high-dimensional spaces. Given a combinatorial optimization problem, the basic idea underlying local search [Hoos and Stutzle, 2004] is to start from an initial search position in the space of all possible assignments, and to improve iteratively this assignment by means of minor modifications. Solutions are compared through an evaluation function, which typically maps the current candidate solution to a real number. In this paper the evaluation function is based on the aggregation of information obtained by running a simulator of ground noise.

This paper reports on experimental results using this approach. The goals of these experiments include assessing the performance and effectiveness of local search using different aggregate evaluation functions. We are aided here by the fact that the simulator has a tunable parameter that allows predictions to range from 'coarse' to 'fine-grained', with exponential reductions in the speed of the simulation between high- and low-resolution simulation. We leverage this feature by tuning it dynamically during search, thus allowing, in effect, to sharpen or coarsen our optimization results on the fly.

The remainder of this paper is organized as follows. In the background section we give preliminary notions on trajectory optimization, noise simulation and local search. We then move to the definition of our setting describing the configuration space, the flyability constraints and the cost functions. Next, we describe the local search approach and the definition of the neighborhood function. We conclude by discussing our experimental setting and results.

BACKGROUND

This section provides the necessary technical background that will be used to formulate the approach presented in this paper.

Trajectory Optimization

The field of trajectory optimization has a long history, with many applications in aerospace and robotics. 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, is thus 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.

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].

In particular, the aforementioned work is the most closely related to the work presented here, since it applies a local search technique for noise minimization. There are, however, several relevant differences. In [Xue and Atkins, 2006], 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 a more sophisticated simulation tool (RNM). Also the local search techniques employed are different, as we use a standard hill-climbing procedure and they use simulated annealing where, each step, the current solution is replaced by a random solution with a probability depending both on the difference between the corresponding function values and also on a global parameter T (temperature), that is gradually decreased during the process.

Noise and how it is measured

Noise is unwanted sound. Sound is variation in air pressure detectable by the human ear in the form of vibration of the ear drum. The decibel is a ratio that compares the sound pressure of the sound source of interest (e.g., the rotorcraft overflight) to a reference pressure (the quietest sound we can hear). Humans can detect sound pressure over a wide range, 10^{-9} to 10^{-3} pounds per square inch (psi). Because the range of sound pressures is very large, we use logarithms to simplify the expression to a smaller range, and express the resulting value in decibels (dB).

Sound can be broken down into frequencies (low, medium, high). 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* [Conner et al., 2006] approximates the sensitivity of the human ear and helps to assess the relative loudness of various sounds.

Sound levels vary with time, which is important if we are interested in the noise associated with a certain event of interest (e.g. an approaching rotorcraft). The simplest way to describe a discrete noise event is with its maximum sound level, or *Lmax* [Conner et al., 2006]. *Lmax* only accounts for noise amplitude, and does not discriminate between noise exposures of short duration or long duration. 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.

The US Federal Aviation Administration (FAA) considers a 1.5 dB the minimum significant change where cumulative exposure is above 65 Day-Night Average Sound Level (DNL). Any abatement strategy that promises over 5 dB change in noise level is considered definitely beneficial. As we show later, we will use this value in assessing and comparing noise cost functions for trajectories.

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 the modulation of sound by the relatively slow-turning main rotor. The resulting sound can become impulsive in character and is referred to as BVI (Blade Vortex Interaction Noise). 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. These fluctuations result in impulsive noise with shock waves that can propagate forward. At lower airspeeds, and typically during a descent, rotor impulsive noise can occur when a blade intersects its own vortex system or that of another blade. This type of noise is BVI noise. When this happens, the blade experiences locally high velocities and rapid angle-of-attack changes. This tends to produce a sound that is loud and very annoying in character [Cox et al., 2009], [Greenwood and Schmitz, 2010].

Rotorcraft Noise Model Simulation Tool

The Rotorcraft Noise Model (RNM)[Conner et al., 2006] is a simulation program that predicts how the sound of a rotorcraft will propagate through the atmosphere and accumulate at observer (receiver) locations. RNM is capable of calculating cumulative noise exposures such as A-weighted SEL. The input to RNM consists of

- a set of computational parameters, including identity of rotorcraft, and the dimensions and resolution of a grid that will display output noise (discussed further below);
- a specification of points of interest; and
- a specification of the flight trajectory, including position, velocity and orientation.

RNM contains a model of how sound propagates through the atmosphere. In general, the noise that propagates from a source to a receiver at a given distance from the source can be expressed as a sum of the following factors:

- the sound level at the source;
- the geometrical spherical spreading loss (since energy is a conserved quantity, the energy per unit area (intensity) of an expanding spherical pressure wave decreases as $1/r^2$. This is called spherical spreading loss.)
- loss due to atmospheric absorption effects, based on theoretical predictions and experimental data;
- ground reflection and attenuation (including the effects of terrain); and
- effects due to wind.

RNM allows for there to be multiple sources of noise from the same rotorcraft.

Noise data either experimentally or analytically generated from models is stored in the form of a *sound sphere*. 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). There may be more than one sphere for the same flight condition; for example, spheres for different locations on the rotorcraft. Figure 1 shows an example sound hemisphere.

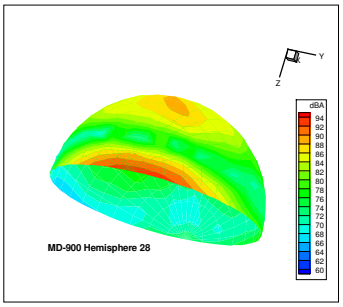

Figure 1. Example of sound hemisphere of an MD-900 helicopter.

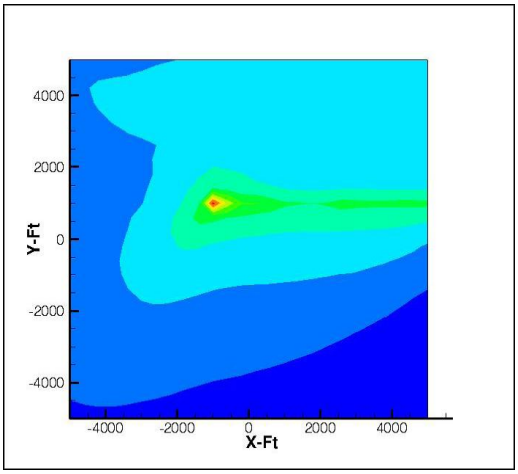

Figure 2. A Noise Contour Plot.

There are three main computational components of the RNM simulation:

- *Input Module*: Linear interpolation over the input trajectory as a pre-processing step. Input data are interpolated (if required) to a default of 2 second spacing.
- *Source Database Lookup and Selection*: Selecting and interpolating over the sound spheres to determine the best representative of the noise generating for a given location and flight condition in the input trajectory; and
- *Source to receiver propagation*: Accumulating and storing the sound for a given receiver.

The second and third components are executed for each trajectory point, sound source, flight operation and receiver location.

RNM simulation produces predictive noise data in various formats. Of interest to our work, is the generation of *ground noise contour plots*: a set of values representing ground noise exposure using A-weighted SEL or other metric 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 local search, as discussed below.

Local Search

Local search [Hoos and Stutzle, 2004], [Aarts and Lenstra, 1997] 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 to 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, although other variants may stop the search after a predefined amount of time. Different local search methods vary in the definition of the neighborhood and of the evaluation function, as well as in the way in which situations are handled when no improvement is possible. 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.

The reasons for preferring local search include:

1. *Anytime performance:* On average, local search behaves well in practice, yielding low-order polynomial running times [Aarts and Lenstra, 1997]. Since the trajectory space is large, it is difficult *a priori* to characterize globally preferred solutions. Consequently, we are interested in a system that can examine large parts of the search space quickly.
2. *Flexibility and ease of implementation:* deployment-related deadlines suggest the use of techniques which are easy to implement.
3. *Simulator Compatibility:* running RNM is heavy from a computational point of view. This means that the repetitive evaluation of partial trajectories, required by complete incremental solving paradigms (e.g. Branch and Bound), may be unacceptably time consuming. Local search, on the other hand, only requires the evaluation of complete solutions.

In hill-climbing search [Selman and Gomes, 2006], we select any local change that improves the current value of the objective function. Greedy local search is a form of hill-climbing search where we select the local move that leads to the largest improvement of the objective function. Traditionally, one would terminate hill-climbing and greedy search methods when no local move could further improve the objective function. Upon termination, the search would have reached a local, but not necessarily global, optimum of the objective function. In recent years, it has been found, perhaps somewhat surprisingly, that simply allowing the local search to continue, by accepting 'sideway' or even 'downhill' moves, i.e. local moves to states with, respectively, the same or worse objective values, one can often eventually still reach a global optimum.

THE TRAJECTORY NOISE OPTIMIZATION PROBLEM (TNOP)

The formulation of the trajectory optimization problem used here is based on techniques developed for motion planning in continuous state spaces. These techniques include defining geometric models for representing location; defining a configuration space of possible transformations of the 'agent' in motion; and methods for discretizing the state space by partitioning the space into regions to enable solutions using graphical methods. The following sections summarize the components used in the problem formulation.

Configuration Space Definition

We focus on approach trajectories (and the nearly identical problem of take-off) because that is where all the community noise problems arise. We will focus on A-weighted SEL as our noise exposure metric. 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 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 $\langle N_j, dist_j, N_{j+1} \rangle$ 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.

In our setting we are given two nodes designated as start and finish, with fixed state and control vectors, and 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 (see Figure 3). One example of a standard approach is a box pattern, as the one shown in Figure 4. Such a pattern can

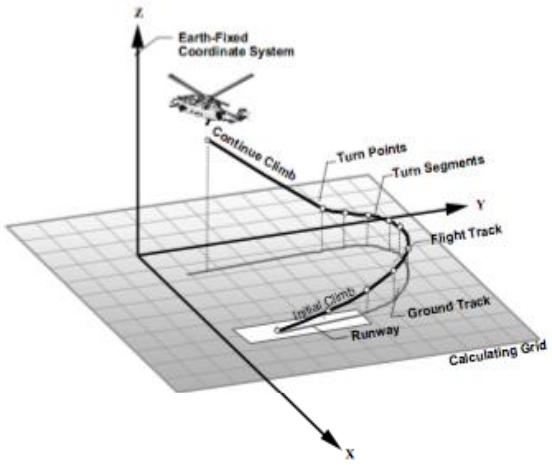

Figure 3. A standard ascent trajectory.

be represented by a sequence of 6 nodes $N_0 \ldots N_5$, where two 90^o turns start, respectively, at N_2 and N_3. Given a box pattern, say $(N_0, \ldots, N_5)$, our goal is to find an assignment say $(s_0, \ldots, s_5, d_0, \ldots, d_5)$ 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.

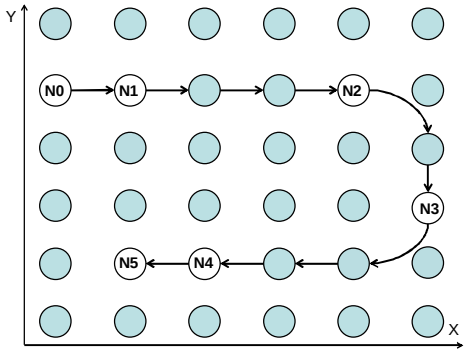

Figure 4. A "box"-like approach pattern.

Flyability Constraints

As mentioned above, we will be searching for flyable box trajectories that minimize noise. Flyable trajectories are ones that maintain aircraft safety and comfort of the passengers. Although these criteria (especially comfort) are somewhat subjective, we have taken advantage of the pilot expertise of one the authors to define the appropriate 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 (acc): $\Delta z_i \in \{\delta_z \| \exists \gamma \in [0^o, 12^o], tan(\gamma) = \frac{\delta_z}{dist_{ij}}\}$.

In addition, there is a minimal velocity and altitude $(vmini, zmini)$ that a rotorcraft must maintain during descent; these are a function of the distance of the craft 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 $vmini$ and $zmini$.

Formally then, a *Trajectory Noise Optimization Problem* (TNOP) is a tuple $\langle S, D, s_0, s_f, acc, dec, vmini, zmini \rangle$, where S is a set of states, D s set of decisions, s_0, s_f are initial and final states, acc, dec are deceleration and descent angle constraints, and $vmini$ and $zmini$ 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

To use RNM noise contour plot data to evaluate the predicted ground noise of candidate trajectories, we introduce two natural ways of 'aggregating' the 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 $\Sigma_{i=1,\ldots,n} w_i = 1$. The intuition behind this function is that of evaluating a solution by how it partitions the grid points into regions of different levels of noise. Thus a solution that assigns lower levels of noise to larger regions of the grid is to be preferred. Weights are used to model this and to further penalize the presence of, even small, extremely noisy regions. Given this heuristic function the goal is to minimize its 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 improvement score of t w.r.t. s is

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

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. As noted earlier, the 1.5dB threshold has been chosen since it is 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 maximize its value.

LOCAL SEARCH FOR THE TNOP

The technique we propose here to solve the optimization problem described in the previous section is a hill-climbing local search approach.

Figure 5 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
  $\sigma_0 = neighbor(\sigma_{cur})$
  $neighborhood(\sigma_{cur}) = neighborhood(\sigma_{cur}) \setminus \{\sigma_0\}$
  while $neighborhood(\sigma_{cur}) \neq \emptyset$ and $score(\sigma_0) \leq score(\sigma_{cur})$
    $\sigma_0 = neighbor(\sigma_{cur})$
    $neighborhood(\sigma_{cur}) = neighborhood(\sigma_{cur}) \setminus \{\sigma_0\}$
  $\sigma_{cur} = \sigma_0$
  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 $p = partner(n)$ // choose adjacent node, either forward or backward
3 $select\ c \in \{\Delta v, \Delta z\}$ // change rate of deceleration or descent
4 $v_c = val(c, p, n)$ // find a an allowable value to transfer
5 $\sigma_n = transfer(n, p, v_c, \sigma)$ // add the value to p and subtract from n
6 $(n, p, c) = used$ // mark triple as used
return σ_n // return the neighbor

Figure 5. Algorithm Box-TNOP-HC.

The inputs to the algorithm are

- a seed solution σ_{seed};
- a scoring function $score$;
- a positive integer $threshold$, representing the number of search steps after which the execution must terminate.

We note that the box trajectory is implicitly represented in σ_{seed}. Moreover, since in our case there is no way to test if an optimal solution as been found, the algorithm will always run for $threshold$ number of steps.

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.

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 at two adjacent nodes of s. Figure 6 illustrates the general case where a node has two adjacent nodes with which to swap values.

To perform the operation that generates a neighbor, a node N_i is chosen at random to be the recipient of the transferred value. A partner N_i (i.e., N_{i-1} or N_{i+1}) and a control variable, ΔV or ΔZ, is chosen (deterministically in a pre-assigned order) from where to transfer value to N_i. 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 N_i's control variable. Note that given a trajectory with L nodes, $N_1, \ldots N_L$, no

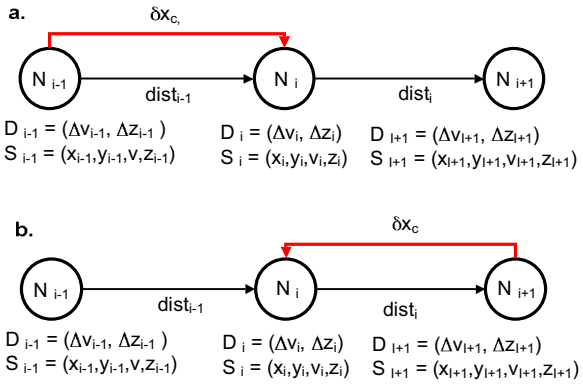

Figure 6. Transferring values between adjacent nodes.

transference is possible for the final node, N_L. The first and $L-1$st nodes have only one partner; the rest of the $L-3$ nodes have two partners.

The value v_c to transfer between a node and its partner 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. This test must be applied whether the partner in the transfer is the forward or backward neighbor of N_i. 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 $vmini$ and $zmini$ defined earlier for the state at the partner node.

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 goals of the experiments we have performed are:

- to assess the runtime behavior of cost functions Bin and $Diff$;
- to determine whether the two apparently incomparable noise cost functions can be combined to produce solutions that were better than solutions obtained from each applied alone;
- to find ways to exploit the ability to tune the data resolution in the simulator to converge on good solutions faster.

Specifically, we compared 6 different local search variants:

1. **Bin**: local search using Bin cost function with 7 bins corresponding to the following SEL ranges $([125, -], [115, 124], [105, 114], [95, 104], [85, 94], [75, 84], [0, 74])$ with weights respectively of $(0.2, 0.2, 0.2, 0.1, 0.1, 0.1, 0.1)$.
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, meaning a larger grid distance between nodes and then running BIN with finer resolution.
6. **HDiff**: same as the previous, but with Diff.

Results

We conducted a number of experiments using the 6 different approaches defined above. Each of these variants was run 5 times, starting with the same initial seed. The threshold value was set to 100 for Bin and Diff, 25 and 50 for each phase of HBin and HDiff and 50 for each phase of BinDiff and DiffBin. We tested two different thresholds for the algorithms using two different resolutions in order to take into consideration also the additional time required for each iteration at high resolution. The default resolution we used is 500ft of grid distance between nodes yielding a grid of size 261. In the approaches using two resolutions the lowest one was set to 1000 ft and the grid was 75.

For each run we have recorded:

- the best solution found and its score;
- the best incumbent score at each iteration step;
- the total amount of time taken to complete all iterations.

Table 1 shows some results of our experiments. In particular, the first column contains the average over the 5 runs of the scores of the best solution. We recall that, given the way the cost functions are defined, lower is always better and that for $Diff$ the final score will be ≤ 0 while for Bin it will be a positive integer. The second column shows the absolute best score found in the 5 runs and the last column shows the average required time.

First, let us compare Bin, DiffBin, and HBin, the three variants to using Bin. The results, although preliminary, indicate that the quality of the solution is not improved when Bin is used with more than one resolution. However, the combination with Diff does produce a good quality result in less than half the time of just running Bin.

In the case of comparing the Diff-based searches (Diff, BinDIff, HDiff), a similar trend can be observed, where the gap between the quality produced by Diff alone and its hybrids is markedly better.

Table 1. Quality Comparison of 6 Approaches to Local Search Optimization.

	Average(5)	Best	Time(sec)
Bin	21.67	21.21	2014
Diff	-128	-196	2012
BinDiff	450.2	930	2998
DiffBin	125.8	110.6	5777
$HBin^{25}$	22.28	21.39	1704
$HDiff^{25}$	-59	-126	1786
$HBin^{50}$	21.6	20.8	5322
$HDiff^{20}$	641.2	1373	5317
$HDiff^{50}$	-53	-77	5999

In Figures 7 and 8 we show the runtime behavior for the single-phase variants Bin and Diff. As it can be seen, Bin converges fast but it is much less discriminating. We speculate that this can be caused by the limited number of bins used and the flattening effect of the weighted sum. Diff, instead, converges slower but discriminates more among solutions. The results of this variant are very encouraging since, as shown in Table 1, the best solutions found are significantly more quiet than the seed solution on almost half of the grid.

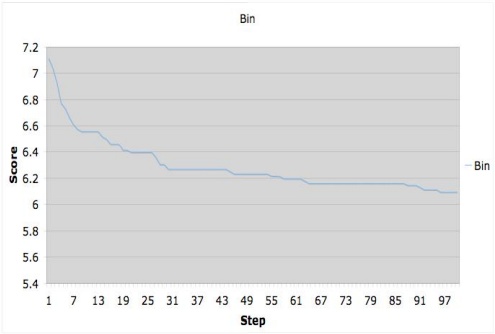

Figure 7. Runtime behavior of Bin

SUMMARY AND FUTURE WORK

This paper has explored the idea of plan optimization through robust simulation. The effective use of simulation in many cases assumes the ability to 'tune' the resolution of the simulation to provide enough advice without hindering the ability to conduct search. We illustrated these ideas on the problem of minimizing rotorcraft noise during approach and landing. We also explored the design of cost functions from

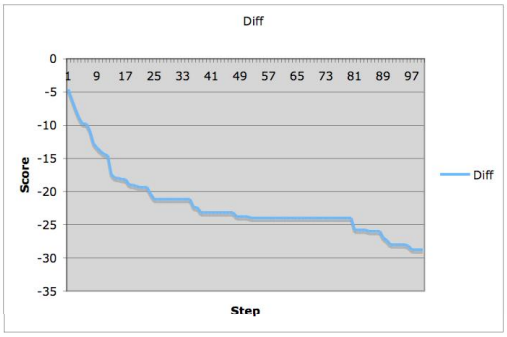

Figure 8. Runtime behavior of Diff.

data acquired through simulation. The experimental results have identified the advantages of employing a heuristic cost function based on identifying regions of significant improvement. Moreover, the results suggest that it is not obvious that the creative use of multiple cost functions and varying data resolution can yield improvements in the performance of local search solvers. Further testing, however, is required to confirm these preliminary results.

Although the data collected from the experiments support the approach of interleaving optimization planning with simulation, the results are somewhat limited, due primarily to the rigid framing of the problem, in particular, in not allowing the decision space to include changes along the position x, y state elements. Consequently, we're in the process of expanding the model to allow changes in position. This change will enable modeling of decisions affecting turns, i.e., turn angle and radius. These changes will increase the dimensionality of the search space, and we anticipate the need for using more sophisticated path planning algorithms, such as those based on probabilistic methods, to explore the trajectory space.

REFERENCES

[Aarts and Lenstra, 1997] Aarts, E. and Lenstra, J. K. (1997). *Local Search in Combinatorial Optimization*. Princeton University Press.

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

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

[Cox et al., 2009] 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 et al., 2003] Goplan, G., Xue, M., Atkins, E., and Schmitz, F. H. (2003). Longitudinal-plane simultaneous non-interfering approach trajectory design for noise minimization. In *Proceedings of the 59th AHS International Forum and Technology Display*, pages 1–18.

[Greenwood and Schmitz, 2010] Greenwood, E. and

Schmitz, F. (May 11-13, 2010). A parameter identification method for helicopter noise source identification and physics-based semi-empirical modeling. In *American Helicopter Society 66th Annual Forum*, Phoenix, AZ.

[Hagelauer and Mora-Camino, 1998] 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 and Stutzle, 2004] Hoos, H. H. and Stutzle, T. (2004). *Stochastic Local Search: Foundations and Applications*. Elsevier - Morgan Kaufmann.

[LaValle, 2006] LaValle, S. (2006). *Planning Algorithms*. Cambridge University Press.

[P. Cheng and LaValle, 2001] 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 and Gomes, 2006] Selman, B. and Gomes, C. (2006). Hill-climbing search. In *Encyclopedia of Cognitive Science*. John Wiley & Sons.

[Xue and Atkins, 2006] 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.