# Designing Noise-Minimal Rotorcraft Trajectories

**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

**Jim Lindsey**
Monterey Technologies, CA (USA)
jimdena@tx.rr.com

### Abstract

The ability to predict rotorcraft ground noise is important in determining and assessing environmental noise impact. The noise generated by rotorcraft can limit their usage and restrict operations, particularly near cities and populated regions. The two primary approaches commonly used to reduce rotorcraft noise are to make vehicle design modifications and to make changes in operational flight procedures. The latter have the advantage that they can often be implemented to achieve significant noise reductions at a lower cost than new design efforts. Computer modeling capabilities for developing low noise procedures have received much attention over the last 15 years. These models, when paired with an automated optimization approach, can facilitate the design of new approach trajectories for improving the environmental impact. This paper describes recent work in applying a constraint-based optimization model and local search, paired with a robust noise simulator, to solve the noise minimal trajectory optimization problem.

## Introduction

There is considerable interest by NASA and the commercial sector to develop a transportation infrastructure that is based on an increased use of rotorcraft, specifically helicopters and tilt-wing 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.

While it is possible to build civil aviation rotorcraft and tiltrotors of various sizes and capacities, the aviation industry and U.S. government officials are concerned with the impact of noise on the communities surrounding the transportation facilities. One way to address the rotorcraft noise problem is to design and test low-noise flight profiles which can be tested in simulation or through field tests.

The objective of this paper is to introduce a Trajectory Noise Optimization Problem (TNOP) for designing noise minimal rotorcraft approach trajectories. The model includes a graphical representation of the computational search space based on the state of the aircraft and the control decisions made by the pilot; a representation of constraints that identify trajectories that are 'flyable' based on pilot-elicited rules of comfort and safety of the aircraft; a noise simulator tool (Rotorcraft Noise Model, RNM) for calculating the effects of sound propagation over varying ground terrain, enabling the quantitative assessment of the overall ground noise produced by a given trajectory; two cost functions that aggregate and quantify the cumulative noise level to allow for trajectories to be compared and ordered based on the noise they produce; and an optimizing search approach using local search. The local search uses a neighborhood function based on a simple exchange of control decisions. The search is initialized using a seed solution manually crafted by a pilot based on standard approach procedures.

The remainder of this paper is organized as follows. A brief introduction to the quantification of rotorcraft noise is presented, followed by an introduction to the TNOP. We then describe the solving approach for finding noise minimal trajectories. Some preliminary results are presented and discussed.



## Background

### Introduction to 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* 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). To take exposure du-

ration 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 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.

## Rotorcraft Noise Simulation

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, which obeys an inverse square law.)
- 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, one sphere for different locations on the rotorcraft. Figure 1 shows an example sphere (actually, a hemisphere).

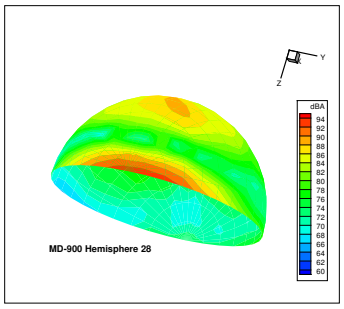


Figure 1: Example of sound hemisphere of an MD-900 helicopter.

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 in the list are repeated 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.

## Trajectory Optimization

The field of trajectory optimization has a long history, with many applications in aerospace and robotics. The basic description of the problem, which we adapt here, is 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 ge-

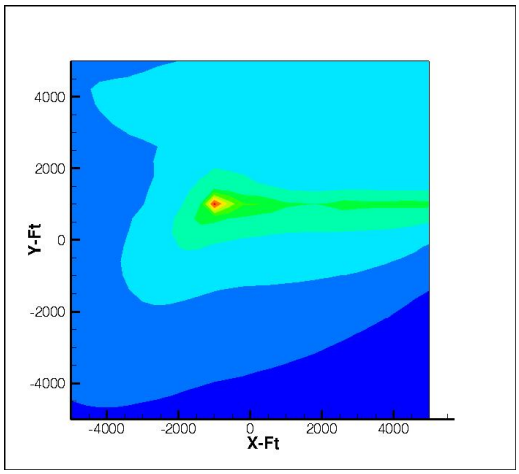


Figure 2: A Noise Contour Plot.

netic algorithms have also been applied (Xue and Atkins, 2006).

## 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.

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.

## Trajectory Optimization Problem Formulation

The *trajectory noise optimization problem* (TNOP) is the problem of designing flight approach trajectories for rotorcraft that minimize the cumulative ground noise exposure from the vehicle. 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 (Figure 3). 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. More formally, each *node*

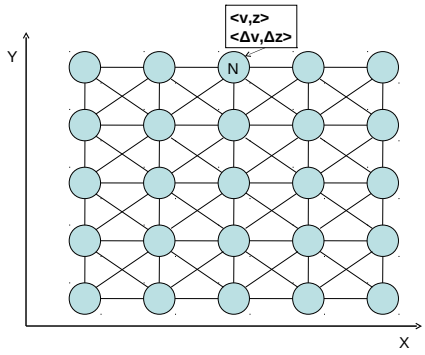


Figure 3: Two dimensional grid of the X-Y space.

$n$ at position $x_n, y_n$ is associated with two vector variables:

- the state vector: $s_n = \langle x_n, y_n, z_n, v_n \rangle$, where $z_n$ is altitude and $v_n$ is speed (velocity). Since $x_n$ and $y_n$ are fixed we will omit them and write $\langle v_n, z_n \rangle$. Intuitively, they represent the altitude and speed of the rotorcraft when flying over position $(x_n, y_n)$;
- the control vector: $d_n = \langle \Delta v_n, \Delta z_n \rangle$, where $\Delta v_n$ represents a decrease in speed, and $\Delta z$ represents a decrease in altitude. Intuitively the control vector represents the changes in speed and altitude that will be applied to the rotorcraft starting from node $n$.

As shown in Figure 3, all pairs of horizontally, vertically or diagonally adjacent nodes are connected by an edge. We assume that each edge has a value representing a distance between the nodes and that the rotorcraft can only move along

edges. Moreover, we do not model turns directly and we assume that the rotorcraft can turn instantaneuosly at any node, involving no action.

In general, we can define a distance $D$ between arbitrary pairs of nodes along a path, as the sum of the distances of the edges in the path. An edge implicitly represents two directed arcs (one for each direction).

We define a $trajectory$ as path on the grid plus the assignment to all the state vectors and control vectors of the nodes traversed by the path. Given a trajectory $t$ through node $n$, with vectors $s_n$ and $d_n$, we denote the assignment to the vecors of $n$ in $t$ as $val(t, s_n) = \langle z_{t,n}, v_{t,n} \rangle$ and $val(t, d_n) = \langle \Delta z_{t,n}, \Delta v_{t,n} \rangle$. A trajectory $t$ is said to be *consistent* if, for every pair of consecutive nodes $p$ and $n$, where $p$ precedes $n$, we have that the value of $val(t, s_n)$ is the result of applying the control actions $val(t, d_p)$ in state $val(t, s_p)$. More precisely,

$$v_{t,n} = v_{t,p} + \Delta v_{t,p} \ , \ z_{t,n} = z_{t,p} + \Delta z_{t,p}.$$

We will be searching for flyable trajectories that minimize noise. Conditions that make a trajectory suitable to fly are, however, usually expressed in terms of constraints over the decent angle and deceleration. In particular, any part of a trajectory should be characterized by an angle of decent $\alpha \in [0^o, 10^o]$ and a deceleration $a \in [0g, 0.1g]$ (or $a \in [40ft/min^2, 201ft/min^2]$). Such restrictions induce constraints on the domain of $\Delta v$ and ($\Delta z$ as follows. Given a pair of nodes $n_i, n_j$ and a path between them of distance $D$ we have:

- the deceleration constraint: $Dom_{D,v_i}(\Delta v_i) = \{\delta_v \| \exists a \in [0, 0.1], \delta_v = \sqrt{v_i^2 + 2a \times D} - v_i\}$, where $a$ is expressed in gs.
- the angle-of-decent constraint: $Dom(\Delta z_i) = \{\delta_z \| \exists \alpha \in [0^o, 10^o], tan(\alpha) = \frac{\delta_z}{D}\}$.

A trajectory is said to be *flyable* if it satisfies all the deceleration and angle-of-decent constraints along its path.

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. One example of a standard approach is a box pattern, as the one shown in Figure 4.

A box pattern can be represented by a sequence of 6 nodes $N_0 \dots N_5$ where turns take place at nodes $N_1$, $N_2$ and $N_3$, $N_0$ is the start of the approach and $N_5$ is the landing point. Given a box pattern, say $(N_0, \dots, N_5)$, our goal is to find an assignment, say $(val(s_1), \dots val(s_5), val(d_0), \dots val(d_5))$, to the state and control vectors of the nodes not fixed by the initial and final conditions, such that the noise simulated by RNM on the corresponding trajectory minimal.

It is thus important to define a way to evaluate the overall noise of a trajectory. In order to do so, we porpose the following two heuristic functions.

**Binning Heuristic function** 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, $\langle r_1, r_2, \dots, r_n \rangle$ 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 $b(t) = \langle b_1(t), b_2(t), \dots, b_n(t) \rangle$ where $b_i(t) = |S_i(t)|$. The bin-score of solution $t$ is

$$Bin(t) = \Sigma_{i=1\dots n} w_i b_i(t)$$

where $w_i$ is the weight associated to the $i$-th bin, $w_i > w_{i+1}$ and $\Sigma_{i=1,\dots,n} w_i = 1$. 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 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.

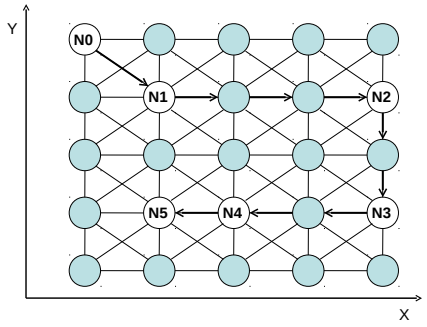


Figure 4: A "box"-like approach pattern.

**Significant Improvement Heuristic function** Let $s$ denote a reference solution and $t$ another solution. Then the significant improvement score of $t$ w.r.t. $s$ is

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

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 Box-TNOP

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

**Box-TNOP-HC**(Trajectory $\sigma_{seed}$, 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 $\sigma_{best}$

**Neighbor**(Trajectory $\sigma$)
do
  $n_v = random(\sigma_{\text{-}\{0,5\}})$ // randomly pick a node
  $n_z = random(\sigma_{\text{-}\{0,5\}})$ // randomly pick a node
while $(n_v, n_z)_{state} = done$
    $p = predecessor(n_v)$
    $\sigma_v = Vswap(p, n_v, \sigma)$
    $p = predecessor(n_z)$
    $\sigma_z = Zswap(p, n_z, \sigma_v)$
$(n_v, n_z)_{state} = done$
return $\sigma_z$

Figure 5: Greedy Local Search Algorithm

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.

Figure 5 describes the pseudocode of our algorithm, which we call *Box-TNOP-HC*. The inputs to the algorithm are

- a seed solution $\sigma_{seed}$;
- a scoring function $score$ that can be either $Bin$ or $SI$;
- 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 $\sigma_{seed}$. 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 $\sigma_{best}$. During the execution we keep track of the current solution, the neighborhood of which we are exploring, denoted by $\sigma_{cur}$, and the best flyable solution found so far, denoted with $\sigma_{best}$. Both such solutions are initially assigned the seed solution. Then, the algorithm starts exploring the neighborhood of $\sigma_{cur}$. 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 $\sigma_{cur}$ and starts scanning its neighborhood. Whenever no better solution is found, a random move in the neighborhood is taken.

## Neighborhood Function

So far our procedure is a standard hill-climbing local search where stagnation in local optima is avoided by random moves in the neighborhood. The key aspect is, of course, on how the neighborhood is defined. The intuition behind the definition of neighborhood we use is that of considering trajectories generated by swapping the decrease in speed and the decrease in altitude of two nodes in the current trajectory. More formally, let us consider a solution $\sigma$ as represented in Figure 6. We recall that for each node $n$ of the box trajectory we have the following assignments to the state and control vector components: $z_{\sigma,n}, v_{\sigma,n}, \Delta v_{\sigma,n}, \Delta z_{\sigma,n}$ .

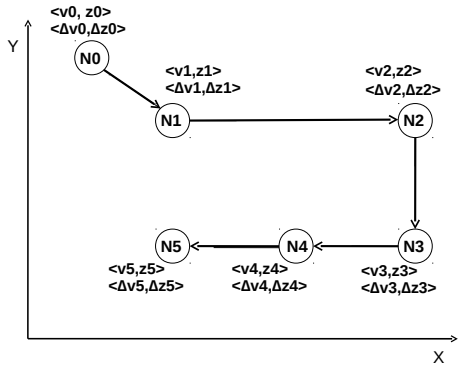


Figure 6: A "box"-like approach trajectory.

Let us consider two consecutive nodes $p$ and $n$ of $\sigma$, where $p$ precedes $n$. We define the new trajectory $\sigma' = Vswap(p, n, \sigma)$, as the trajectory that coincides with $\sigma$ except for the following values:

$$\Delta v_{\sigma',p} = \Delta v_{\sigma,n}$$

and

$$v_{\sigma',n} = v_{\sigma,p} + \Delta v_{\sigma,n}\ \Delta v_{\sigma',n} = \Delta v_{\sigma,p}.$$

Note that $v_{\sigma',p} = v_{\sigma,p}$. In other words, $\sigma'$ is obtained by swapping the speed decrease of $p$ and $n$ and by propagating the effect of this from $p$ to $n$. Similarly, we can define $\sigma'' = Zswap(p, n, \sigma)$ as the trajectory obtained from $\sigma$ by swapping the $\Delta z$ values of $p$ and $n$, and propagating the effect of this from $p$ to $n$.

**Theorem 1** *Assume we are given a consistent trajectory $\sigma$ and two consecutive nodes, $p$ and $n$, where $p$ precedes $n$ in $\sigma$. Then trajectory $\sigma' = Vswap(p, n, \sigma)$ is consistent.*

**Proof:** First we note that the only changes regard speed, and thus, $\sigma'$ is consistent as far as altitude since it coincides with $\sigma$. Let $\sigma'_{<p}$, resp. $\sigma'_{>n}$, be the sub-trajectory obtained from $\sigma'$ considering only the nodes of $\sigma'$ preceding $p$, resp. following $n$. Since, $v_{\sigma',p} = v_{\sigma,p}$, $\sigma_{<p}$ is consistent.

In $\sigma'$ the values of speed and decrease in speed at $p$ are as follows: $v_{\sigma',p} = v_{\sigma,p}$, $\Delta v_{\sigma',p} = \Delta v_{\sigma,n}$. From the consistency constraint, we get that the speed at $n$ in $\sigma'$ should be $v_{\sigma',n} = v_{\sigma',p} + \Delta v_{\sigma',p}$. Using the definition of $Vswap$ and substituting, we get $v_{\sigma',p} + \Delta v_{\sigma',p} = v_{\sigma,p} + \Delta v_{\sigma,n}$, which is the value assigned to the speed of $n$ in $\sigma'$ by $Vswap$. This allows us to conclude that the consistency constraint for speed between $p$ and $n$ is satisfied in $\sigma'$. In order to conclude we only need to prove that $\sigma'_{>n}$ is consistent. We do so by showing that $v_{\sigma,n} + \Delta v_{\sigma,n} = v_{\sigma',n} + \Delta v_{\sigma',n}$. In fact,

$$\begin{aligned} v_{\sigma,n} + \Delta v_{\sigma,n} = & \\ & = v_{\sigma,p} + \Delta v_{\sigma,p} + \Delta v_{\sigma,n} \\ & = v_{\sigma',p} + \Delta v_{\sigma',n} + \Delta v_{\sigma',p} \\ & = v_{\sigma',n} + \Delta v_{\sigma',n}. \end{aligned}$$

□

An analogous result can be given for $Zswap$.

Given a trajectory $\sigma$, we define its neighborhood as the set

$$neighborhood(\sigma) = \{\sigma' | \sigma' = Zswap(p', n', Vswap(p, n, \sigma)), \exists p', n', p, n\}$$

In words, the neighbors of a trajectory are all the trajectories obtained from it by a swap of the decrease of speeds of two consecutive nodes followed by a swap of decrease of altitude of two (not necessarily different) consecutive nodes. Thus, each neighbor can be identified with the pair of nodes $(n, n')$.

Procedure $Neighbor$ shown in Figure 5 takes in input a trajectory $\sigma$ and computes a new trajectory in its neighborhood. As it can be seen from the pseudocode, it first picks two random nodes, $n_v$ and $n_z$, in the set containing all nodes except $N_0$ and $N_5$, denoted with $\sigma_{\text{-}\{0,5\}}$) . This is done by applying function $random$, until a pair of nodes corresponding to an unexplored neighbor is formed. Since function $Neigbor$ is called from *Box-TNOP-HC* only when the neighborhood is not empty, this while loop will always terminate. Next, trajectory $\sigma_v$, obtained by applying $Vswap$ to $n_v$ and the node preceding it, is computed. Finally, $\sigma$'s neighbor, $\sigma_z$ is computed by applying $Zswap$ to $n_z$ and its predecessor in $\sigma_v$. The procedure ends by labeling $\sigma_z$ as an explored neighbor. This is implemented by setting the state of pair $(n_v, n_z)$ to $done$.

## Experiments

In this section we present some preliminary experimental results on the run-time behavior of our system. First, in understanding the experimental design, it is important to emphasize the distinction between the number of grid points in the problem and the number of nodes. A grid point represents a data point for our cost functions, a point on the ground at which a noise prediction is made by the simulator. A grid point may correspond to a node (if they share the same x-y values) but not necessarily (one may choose to define a node that is outside of the grid if he or she does not care to measure the noise at that point). Intuitively, the number of nodes is chosen to correspond roughly to the number of distinct control decisions a pilot typically makes during approach or take-off. 6 is a reasonable number of decisions, and so for our problem sets to this point our trajectories consist of 6 nodes (we will vary this number in future experiments).

The *data resolution* of the problem is the number of distinct data points used to evaluate trajectories, and is also something the designer can vary. Clearly, there is a computational cost incurred by a high resolution, both during the simulation, and in the post-processing within the cost functions. Also not surprisingly, this burden is felt more intensely in the simulation than in the post-processing, where more complex math is performed.

The number of data points can be derived from the formula: $DP = (\frac{UL+LL}{Dist} + 1) \times (\frac{UL+LL}{Dist} + 1)$, where (assuming a square grid) $UL$ and $LL$ represent the lower and upper bounds along the y axis, and $Dist$ represents the distance between nodes (here, expressed in feet). All the experiments recorded have $UL + LL = 10000$ feet.

We have conducted experiments with values of $Dist$ between 100 ($DP = 10201$) and 1000 ($DP = 121$). By definition, with higher resolution there is the ability to discriminate better among the costs of different solutions, but again at a cost of more processing. With the lowest resolution, we found that many runs did not produce a large difference in cost among solutions. On the other hand, we have yet to be able to conduct a large local search (over 100 samples) on the highest resolution.

Therefore, for the experiments recorded here we have considered a grid structure of 441 ($DIST = 500$) nodes and we have used the trajectory shown in Figure 7 as the seed solution. The values assigned in the seed solution have been manually chosen in order to represent a default trajectory that would be flown by a pilot.

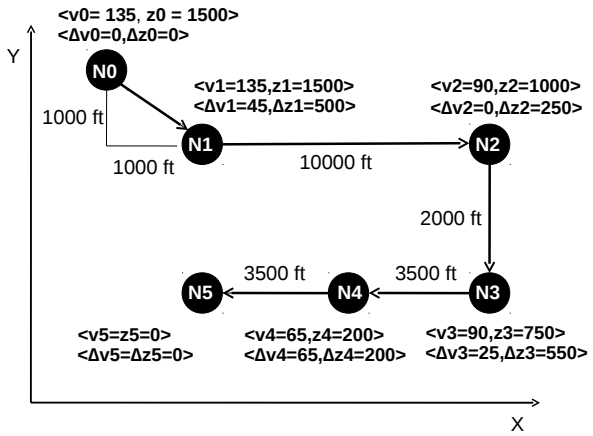


Figure 7: A Seed solution.

We have run our algorithm 20 times with a threshold of 100 steps using both scoring functions. We used 4 bins with

vector of ranges $\langle [115, +\infty], [100, 114], [85, 99], [-\infty, 84] \rangle$ and weights $\langle 0.5, 0.2, 0.2, 0.1 \rangle$. Figures 8 shows the runtime behavior for both heuristic functions. Our experiments suggest that both functions converge quickly (fewer than 25 samples) to a locally optimal or near-optimal value. This suggests a strategy for local search in which we use fewer samples for each run, but employ restarts or other methods for wide exploration. We are currently conducting these experiments. We are also exploring the use of different seed solutions based on alternative standard paths designed by the pilot in our experiments. We are also designing experiments that will use the best result of lower resolution search as a seed for a more focused higher resolution search.

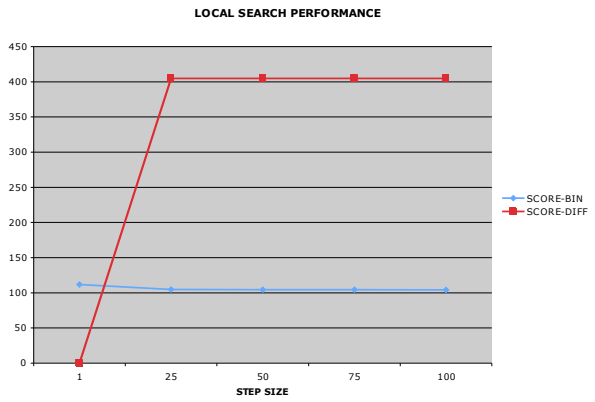


Figure 8: Runtime behavior of *Box-TNOP-HC when* $score = Bin$ *(blue diamond) and* $score = SI$ *(red square).*

## Conclusions and Future Work

This paper describes a constraint-based optimization model for the problem of minimizing the noise of approach trajectories for rotorcrafts. We show how such a model is a suitable representation when a local search based approach is used to find better solutions. The presented framework is appealing since it allows to incorporate constraints representing the knowledge provided by experts (such as pilots) and provides a structured and straightforward way to embed simulation results within search. Experimental results, while preliminary, look promising and suggest this line of research as having the potential for providing significant improvements concerning rotorcraft noise minimization.

An extensive experimental scenario is the first item on our agenda. We also plan to design new heuristic functions to be used within the hill-climbing schema as well as to investigate the possibility of systematic search. In the near future we plan to incorporate turns, either directly into search or as modifications of the solutions found by our system. Other, less imminent, issues regard incorporating information on different sensitivity levels for different areas around heliports and the embedding of our system with on-board guidance tools.

## Acknowledgements

We would like to thank Eric Greenwood (NASA Langley Research Center) for the extremely helpful insight he has provided on matters regarding acoustics and RNM. Thanks also to David Smith for helpful feedback and suggestions for designing the neighborhood function.

## References

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

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 *Proceedings of the 62nd Annual Forum of the American Helicopter Society*, pages 1–17.

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.

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.

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.