The Knowledge Engineering Review, Vol. 32, e20, 1–14.
doi:10.1017/S0269888917000121

Autonomous Unmanned Aerial Vehicle (UAV) landing in windy conditions with MAP-Elites

SIERRA A. ADIBI[1], SCOTT FORER[2], JEREMY FRIES[2] and LOGAN YLINIEMI[2]

[1]*William E. Boeing Department of Aeronautics & Astronautics, University of Washington, Box 352400, Seattle, WA 98195, USA; e-mail: sierra.adibi@gmail.com*;
[2]*Department of Mechanical Engineering, University of Nevada, 1664 N. Virginia St., Reno, NV 89557, USA; e-mail: sforer580@gmail.com, friesjeremy@gmail.com, logan@unr.edu*

Abstract

With the recent increase in the use of Unmanned Aerial Vehicles (UAVs) comes a surge of inexperienced aviators who may not have the requisite skills to react appropriately if weather conditions quickly change while their aircraft are in flight. This creates a dangerous situation, in which the pilot cannot safely land the vehicle. In this work we examine the use of the MAP-Elites algorithm to search for sets of weights for use in an artificial neural network. This neural network directly controls the thrust and pitching torque of a simulated 3-degree of freedom (2 linear, 1 rotational) fixed-wing UAV, with the goal of obtaining a smooth landing profile. We then examine the use of the same algorithm in high-wind conditions, with gusts up to 30 knots.

Our results show that MAP-Elites is an effective method for searching for control policies, and by evolving two separate controllers and switching which controller is active when the UAV is near-ground level, we can produce a wider variety of phenotypic behaviors. The best controllers achieved landing at a vertical speed of <1 m s^{-1} and at an angle of approach of $<1°$ degree.

1 Introduction

In recent years, Unmanned Aerial Vehicles (UAVs) have seen a surge in popularity in a wide range of applications, from military to recreational, due largely to their expanding capabilities. With this rise in popularity comes a drastic increase in the number of aircraft piloted by inexperienced operators and a higher rate of incidents involving unmanned craft (Oncu & Yildiz, 2014).

In order to mitigate some of the risk brought on by the projected 1.6 million UAVs sold to hobbyists in 2015, the Federal Aviation Administration (FAA) implemented a series of regulations designed to promote safety in the US' airspace (FAA, 2015). Despite these efforts, licenses are not required for hobbyists operating small UAVs. When the tactical understanding of flight mechanics that comes with pilot training is absent, further safety precautions are necessary.

Human error is well understood to be a contributing factor in the majority of aviation accidents (Li & Harris, 2006; Shappell *et al*., 2007), and for inexperienced pilots, adverse weather conditions significantly increase the risk of incident (G & SP, 2007). In particular, landing a fixed-wing aircraft in high-wind conditions can cause a significant number of problems for a pilot who is not familiar with the appropriate procedures (FAA, 2008). For those UAV hobbyists seeking the longer range and higher speeds offered by fixed-wing aircraft, these difficulties can translate into distinct risks.

In this work we use a model of this scenario as a challenging testbed for examining the use of the MAP-Elites algorithm (Mouret & Clune, 2015) to search for successful control policies for autonomous landing, even in high-wind situations. Our model consists of a 3-degree of freedom (DOF) physics-based flight simulator

(2 linear DOF, x and z, and 1 rotational DOF about the centroid of the wing of the UAV, ϕ) over the Euclidean plane. With further study, this could be developed into a system to help mitigate the risks from inexperienced pilots in a difficult landing scenario.

The major contributions of this work are to:

- Investigate the use of the MAP-Elites algorithm in a highly dynamic UAV control environment including gusting wind.
- Develop a method for improving the phenotypic diversity discovered by the MAP-Elites algorithm through near-ground control switching (NGCS).
- Provide a set of recommendations for choosing phenotypes for MAP-Elites in highly dynamic problems.

The rest of this paper is organized as follows: Section 2 describes the necessary background on artificial neural networks (ANN), MAP-Elites, and flight mechanics. Section 3 provides the details of the physics-based flight simulator used. Section 4 describes the simulator verification process. Section 5 describes the experimental parameters for the flight simulator and MAP-Elites in this work. Section 6 presents our experimental results for UAV control with MAP-Elites in no- and high-wind situations, with and without NGCS. Finally, Section 7 concludes the work and addresses lines of future research.

2 Background

In this paper, we propose the use of an ANN in conjunction with the MAP-Elites search algorithm to develop robust controllers for fixed-wing landing in a variety of conditions. This section includes the necessary background on ANNs (Section 2.1), MAP-Elites (Section 2.2), and flight mechanics (Section 2.3) and situates our work within the literature (Section 2.4).

2.1 Artificial neural networks

An ANN is a powerful function approximator, which has been used in tasks as varied as weather forecasting (Mellit & Pavan, 2010), medical diagnosis (Baxt, 1991), and dynamic control (Lewis *et al.*, 1998; Yliniemi *et al.*, 2014b). Neural networks have also been successful in many direct control tasks (Jorgensen & Schley, 1995; Yliniemi *et al.*, 2014a). An ANN is customized for a particular task through a search for 'weights', which dictate the output of an ANN, given an input.

In this work, we use a single-hidden-layer, fully-connected, feed-forward neural network. We normalize the state variables input into the network by using their upper and lower limits so that each state variable remains on the same scale. The neural network, using the normalized state inputs, calculates the normalized control outputs, which are then scaled based on the desired bounds for thrust and torque.

By using a search algorithm, appropriate weights can be found to increase the ANN's performance on a measure of fitness. With a sufficient number of hidden nodes, an ANN is capable of approximating any function (Hornik *et al.*, 1989) if the appropriate weights can be found through a search method.

2.2 MAP-Elites

MAP-Elites is a search algorithm which has the basic functionality of 'illuminating' the search space along low-dimensional phenotypes—observable traits of a solution—which can be specified by the system designer (Mouret & Clune, 2015). For an effective search, these phenotypes do not need to have any specific features, except that they are of low dimension. MAP-Elites has been successfully used in the past for: re-training robots to recover performance after damage to limbs (Cully *et al.*, 2015), manipulating objects (Ecarlat *et al.*, 2015), soft robotic arm control, pattern recognition, evolving artificial life (Mouret & Clune, 2015), and image generation (Nguyen *et al.*, 2015).

MAP-Elites is population-based and maintains individuals based on their fitness, $\mathcal{P}$, and phenotype, $\mathbf{b}$; Figure 1 shows a simplification of the algorithm. The MAP, $\mathbb{M}$, is described by outer limits on each phenotypic dimension and a resolution along each dimension. This forms a number of *bins*, which are

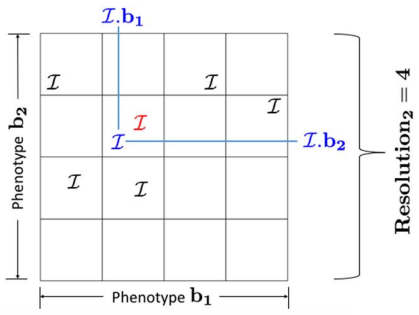

Figure 1 A simple representation of the MAP-Elites algorithm. At most one individual can be maintained in each bin. After simulation, the blue individual is placed in the same bin as the red individual, due to its phenotype, **b**. The individual with the higher fitness will be maintained. The other will be discarded

differentiated based on one or more phenotypes. Each bin may only contain a single individual $\mathcal{I}$ which bears a phenotype within a certain range. When multiple individuals exist with similar phenotypes, the bin stores only the more fit individual. This process offers protection to individuals which generate unique phenotypes, as there is likely less competition in these bins. The system designer can then examine how the fitness surface changes across a phenotype space, which consists of directly observable behaviors. MAP-Elites is related to an evolutionary algorithm in that a single bin that can support n individuals is equivalent to an evolutionary algorithm with a carrying capacity of n individuals.

Algorithm 1 describes how MAP-Elites generates solutions; the process occurs in three stages: 'creation', 'fill', and 'mutate'. The creation stage initializes all of the bins, each of which can hold a single individual (in this case, a set of weights to be given to the neural network for control) within a certain range of phenotypes.

Algorithm 1: `Map-Elites` algorithm. The map ($\mathbb{M}$) is populated with individuals ($\mathcal{I}$) based on their phenotype ($\mathbf{b}$) and fitness ($\mathcal{P}$).

Input: N_F, N_M
Output: $\mathbb{M}$

$\mathbb{M} \leftarrow$ `InitializeMap()`
for $iter = 1 \rightarrow N_F$ **do**
    `// Fill map loop`
    $\mathcal{I} \leftarrow$ `BuildIndividual()`
    $\mathcal{I}.\{\mathcal{P}, \mathbf{b}\} \leftarrow$ `Simulation`$(\mathcal{I})$
    $\mathbb{M} \leftarrow$ `Place`$(\mathcal{I})$
for $iter = 1 \rightarrow N_M$ **do**
    `// Mutate map loop`
    $\mathbf{b}_{iter} \leftarrow$ `RandomBinPhenotype()`
    $\mathcal{I} \leftarrow$ `GetIndividual`$(\mathbb{M}, \mathbf{b}_{iter})$
    $\mathcal{I}' \leftarrow$ `CopyIndividual`$(\mathcal{I})$
    $\mathcal{I}' \leftarrow$ `MutateIndividual`$(\mathcal{I}')$
    $\mathcal{I}'.\{\mathcal{P}', \mathbf{b}'\} \leftarrow$ `Simulation`$(\mathcal{I}')$
    $\mathbb{M} \leftarrow$ `Place`$(\mathcal{I}')$
return *Map of solutions* $\mathbb{M}$

The fill stage consists of generating random individuals, simulating those individuals, and placing them in the appropriate bin within the map. In the case of two individuals belonging to the phenotype range of the same bin, the more fit individual survives, while the other is discarded.

In contrast, during the mutate stage, one of the individuals within the map is randomly copied, mutated, and simulated. This mutation occurs by changing the individual's genotype (weights of the neural network). Such a mutation will typically result in a change in phenotype evaluation, so the resulting

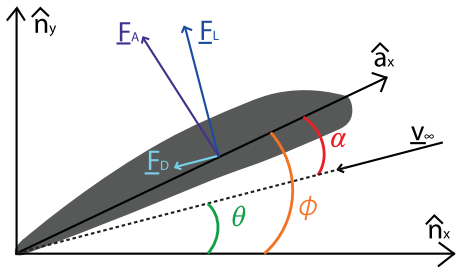

Figure 2 The aerodynamic force vectors which act the airfoil

individual may be placed in a different bin than the parent individual. In this way the map can continue to be filled during the mutate stage. This stage can continue until a stopping condition is met; in this work we choose a preset number of iterations and examine the final individuals after this process is complete.

The total number of individuals that can be maintained is equal to the number of bins, but in cases of lower phenotypic diversity in the population, fewer individuals may be maintained, as fewer bins are accessed. A major benefit of the MAP-Elites algorithm is that it not only preserves individuals with unique behaviors (because they may exist in a low-competition bin), but also that it encourages a spread of behaviors across the phenotype space. This may allow a system designer to better describe the shape of the fitness surface across phenotype dimensions.

2.3 Flight mechanics

In this section, we discuss the principles of aerodynamics utilized in the flight simulator (Anderson, 2010). The following theory is used in conjunction with computational data for a NACA 2412 airfoil, as calculated by the XFOIL airfoil analysis software (Drela, 2013).

The aerodynamic forces of lift (F_L) and drag (F_D) on an airfoil are a function of the axial shear stresses (A) and normal pressure (N), as well as the angle between the airfoil chord and the velocity vector, known as the angle of attack (α). Figure 2 depicts the directions that F_L and F_D act on the airfoil, as well as the total aerodynamic force, F_A; F_L is always directed normal to the direction of the free stream velocity, V_∞, and F_D is always in the direction of the V_∞. In the figure, ϕ represents the pitch of the airfoil with respect to the horizontal, and θ is the angle between V_∞ and the horizontal. Equations (1) and (2) describe the method of obtaining F_L and F_D:

$$F_L = N\cos\alpha - A\sin\alpha \tag{1}$$

$$F_D = N\sin\alpha + A\cos\alpha \tag{2}$$

These values describe the behavior of an airfoil under a specific set of conditions, and they are often used in their dimensionless forms, known as the coefficients of lift (C_L) and drag (C_D). They are calculated by dividing the forces by the planform area of the wing (s_{ref}) and the free stream dynamic pressure (q_∞).

Calculations of s_{ref}, q_∞, C_L, and C_D are as follows:

$$s_{\text{ref}} = c \cdot \ell \tag{3}$$

$$q_\infty = \frac{1}{2}\rho_\infty V_\infty^2 \tag{4}$$

$$C_L = \frac{F_L}{q_\infty s_{\text{ref}}} \tag{5}$$

$$C_D = \frac{F_D}{q_\infty s_{\text{ref}}} \tag{6}$$

In the preceding equations, c is the average chord length of the wing and ρ_∞ the free stream air density. The force coefficients are thus dependent on the size and geometry of the wing, as well as the angle of

attack, while remaining independent of the free stream air density and speed. With previously calculated values for C_L and C_D for varying α, F_L and F_D can be calculated using the following Equations (7) and (8):

$$F_L = \frac{1}{2} \cdot C_L \cdot V_\infty^2 \cdot \rho_\infty \cdot s_{\text{ref}} \tag{7}$$

$$F_D = \frac{1}{2} \cdot C_D \cdot V_\infty^2 \cdot \rho_\infty \cdot s_{\text{ref}} \tag{8}$$

It is an important note that V_∞ denotes the speed of the air in reference to the craft, which is often different from the speed of the craft with respect to the ground. This difference may be caused by the presence of wind, as is the case in our simulator.

Fixed-wing aircraft typically use propellers to create thrust, which acts parallel to the craft's body, denoted as the $\hat{a}_x$ direction. In addition to controlling pitch, or rotation about the $\hat{a}_y$, through use of the elevator, a craft can also control its rotation about the $\hat{a}_x$, known as roll, and rotation about the $\hat{a}_z$, known as yaw. Figure 3 depicts the rigid frame used to describe the aircraft body. In this work, we use a simulator with 3 DOF: 2 linear (x and z) and 1 rotational (pitch about $\hat{a}_y$). For an aircraft to increase the amount of lift it can generate at a given speed, it can rotate about the $\hat{a}_y$ axis to change α; Figure 4 displays the values of C_L and C_D used for a variety of α in the simulator (Drela, 2013).

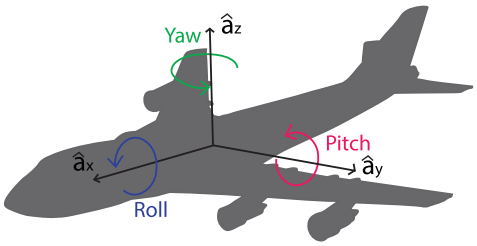

Figure 3 The rigid frame used to describe the aircraft

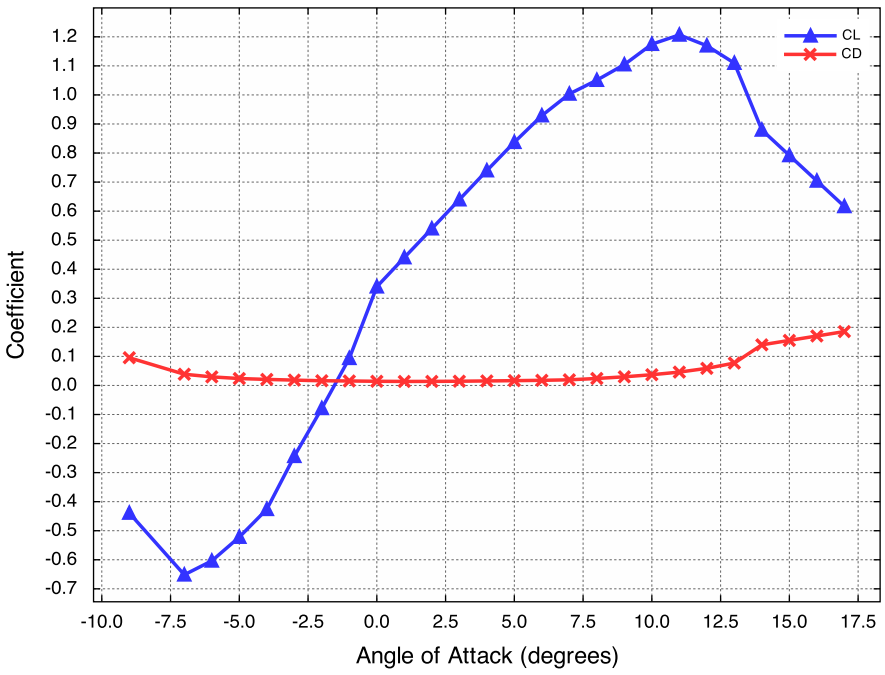

Figure 4 Plot of C_L and C_D for varying α on a NACA 2412 airfoil; `GetCoefficients(`α`)` returns these values

2.4 Related work

In this work, we study the use of the MAP-Elites search algorithm (Mouret & Clune, 2015) to develop successful weights for neural networks to act as control policies for a UAV in high-wind conditions. Autonomous flight and landing of UAVs has been a topic of interest for multiple decades (Foster & Neuman, 1970; Kim *et al.*, 2003; Green & Oh, 2006; Yliniemi *et al.*, 2014b). As such, here we only provide a small sample of related works. For a more comprehensive view of autonomous UAV control, we refer the reader to Gautam *et al.*'s (2014) work; for MAP-Elites, the work by Mouret and Clune (2015).

Most of the work on autonomous UAV control has consisted of model-based control schemes attempting to follow a pre-defined flight path. These control schemes can be either linear, linear with regime-switching, or nonlinear (Alexis *et al.*, 2011; Saeed *et al.*, 2015). In contrast, in this work we do not need a system model and instead use a search algorithm to develop weights for a neural network for model-free control.

Shepherd showed that an evolved neural network can outperform even a well-tuned linear controller with proportional, integral, and derivative gains (PID controller) for a quadrotor UAV; the craft was able to recover from disturbances of up to 180° (being turned upside-down), while a PID controller was only capable of recovering from disturbances of <60° (Shepherd & Tumer, 2010). In contrast, in this work we are performing a landing task with a fixed-wing UAV and using a different search mechanism.

Previous work on MAP-Elites has also been used to search for neural network weights for various tasks (Cully *et al.*, 2015; Ecarlat *et al.*, 2015; Mouret & Clune, 2015). In this work we also search for neural network weights; however, the task in this work has unique dynamics compared to previous MAP-Elites studies and a strong sequential decision-making component.

3 Flight simulator design

To model UAV flight behavior, a 3 DOF flight simulator was designed for 2 linear and 1 rotational DOF. This 3 DOF simulator does not capture all dynamics (as a vehicle in flight has 6 DOF), but this more complex representation is beyond the scope of this work. The simulator approximates the effects of aerodynamic forces, while maintaining realistic aircraft behavior. It operates by receiving thrust and pitch controls, calculating the aerodynamic and gravitational forces on the aircraft, combining the forces, and calculating the system's physical response. All pitching moments caused by aerodynamic forces are neglected due to the use of trim (Bauer, 1995) and C_L and C_D for varying α of the aircraft are based on that of a NACA 2412 airfoil, as predetermined using XFOIL (Drela, 2013).

Algorithm 2: `SimulateTimeStep` communicates with the ANN to get the controls for the time step, then returns the new state, as calculated by the forces acting on the aircraft.

Input: t, S_t
Output: S_{t+t_s}
$\{F_C, M_C\} \leftarrow$ `GetControls`(S_t)
$\{V_\infty, \theta\} \leftarrow$ `GetAirSpeed`$(\dot{r}_x, \dot{r}_z)$
$\alpha \leftarrow$ `GetAttackAngle`(θ, S_t)
$\{C_L, C_D\} \leftarrow$ `GetCoefficients`(α)
$\{F_L, F_D\} \leftarrow$ `CalcAeroForces`(C_L, C_D, V_∞)
$\vec{F}_R \leftarrow$`GetVectorComponents`$(S_t, \theta, F_L, F_D, F_C)$
$S_{t+t_s} \leftarrow$ `DynamicsCalc`$(S_t, \vec{F}_R, M_C)$
return S_{t+t_s}

Algorithm 2, `SimulateTimeStep`, describes the main function of the simulator. While the time, t, is less than the maximum run time, $t_{\max}$, the simulator provides the current state, S_t, to the ANN through the function `GetControls` and receives the force and moment applied, F_C and M_C, respectively. The simulator then calculates V_∞ and θ from the aircraft's ground speed in the 2 linear DOF, $\dot{r}_x$ and $\dot{r}_z$, and the wind generator. The average peak wind speed was ~15 m s^{-1}, chosen to model the type of high winds that could shake a tree and would be difficult for a person to walk in (Lewis, 2011); a typical wind profile for a 30 seconds run can be seen in

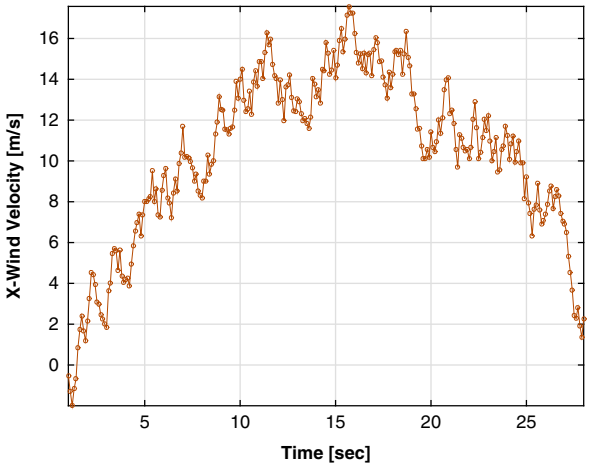

Figure 5 One sample of the simulator's randomly generated wind; in this work we center the distribution around a sine function with amplitude 15 m s^{-1}

Figure 5. Using θ and S_t, α, C_L, and C_D are calculated. After determining the aerodynamic forces on the aircraft, all of the forces are summed. The resultant force vector, $\vec{F}_R$, is put into the function `DynamicsCalc` along with S_t and M_C; thereby the new state, S_{t+t_s}, is determined.

Algorithm 3: `DynamicsCalc` determines the new state of the aircraft using Newtonian physics and trapezoidal integration.

Input: S_t, $\vec{F}_R$, M_C
Output: S_{t+t_s}

for *each linear DOF, i,* **do**
  $\ddot{r}_{i,t+t_s} \leftarrow \frac{F_{R,i}}{m}$
  $\dot{r}_{i,t+t_s} \leftarrow \dot{r}_{i,t} + \frac{1}{2} \cdot t_s \cdot (\ddot{r}_{i,t} + \ddot{r}_{i,t+t_s})$
  $r_{i,t+t_s} \leftarrow r_{i,t} + \frac{1}{2} \cdot t_s \cdot (\dot{r}_{i,t} + \dot{r}_{i,t+t_s})$
for *each rotational DOF, j,* **do**
  $\ddot{\phi}_{j,t+t_s} \leftarrow \frac{M_{C,j}}{I_j}$
  $\dot{\phi}_{j,t+t_s} \leftarrow \dot{\phi}_{j,t} + \frac{1}{2} \cdot t_s \cdot (\ddot{\phi}_{j,t} + \ddot{\phi}_{j,t+t_s})$
  $\phi_{j,t+t_s} \leftarrow \phi_{j,t} + \frac{1}{2} \cdot t_s \cdot (\dot{\phi}_{j,t} + \dot{\phi}_{j,t+t_s})$
$S_{t+t_s} \leftarrow \{\vec{\ddot{r}}_{t+t_s}, \vec{\dot{r}}_{t+t_s}, \vec{r}_{t+t_s}, \vec{\ddot{\phi}}_{t+t_s}, \vec{\dot{\phi}}_{t+t_s}, \vec{\phi}_{t+t_s}\}$
return S_{t+t_s}

The function `DynamicsCalc`, is described in detail in Algorithm 3. The algorithm was designed for scalability and therefore contains two loops: one for calculating linear system responses and another for calculating rotational system responses. The first loop runs through an iteration for each linear DOF, i; Newton's first law of motion is used to calculate the acceleration, $\ddot{r}_{i,t+t_s}$, then the aircraft's velocity, $\dot{r}_{i,t+t_s}$, and position, $r_{i,t+t_s}$, are calculated using trapezoidal integration.

The second loop then performs an iteration for each rotational DOF, j. It calculates the aircraft's angular acceleration, $\ddot{\phi}_{j,t+t_s}$, and performs trapezoidal integration to calculate the aircraft's angular velocity, $\dot{\phi}_{j,t+t_s}$, and pitch, $\phi_{j,t+t_s}$. Finally, all of the state variables form S_{t+t_s}, and the new state is returned.

4 Simulator verification

In order to verify that the flight simulator effectively models fixed-wing flight, we conducted preliminary experiments to test the functionality of the aerodynamic forces before introducing the neural network and MAP-Elites algorithm: a takeoff test and a stable flight test.

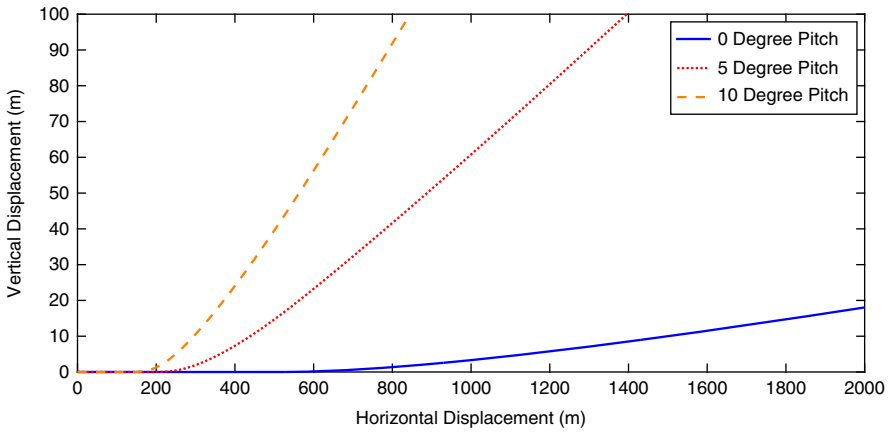

Figure 6 x- vs. z-position during the takeoff simulator verification test for 0, 5, and 10° constant ϕ

To ensure that the simulator properly calculates aerodynamic forces, we modeled a fixed-wing takeoff, and compared against the well-understood behavior of a fixed-wing craft. With $\phi = 0$, a constant thrust of 100 N was applied to the aircraft for 60 seconds. After approximately 14 seconds, at a speed of 70 m s^{-1}, the aircraft lifted off, and continued to climb for the remainder of the simulation. The simulation was then repeated with ϕ increased to 5°, then 10° with respect to the horizontal. A plot of the horizontal and vertical displacements of the three simulations is shown in Figure 6. For the three simulations, as ϕ increases, the total horizontal displacement required for takeoff to occur decreases, which agrees with aerodynamic theory. To verify lift for both positive and negative α, ϕ was set to −10° with respect to the horizontal, and the same test was performed, this time allowing the thrust to act for a total of 8 minutes. Due to the negative coefficient of lift associated with this negative angle of attack, the aircraft never lifts off, thus validating the aerodynamic lift forces of the simulator for both positive and negative α.

For the stable flight test, a force balance was first conducted on the aircraft to determine its horizontal equilibrium speed and equilibrium thrust at $\phi = 0$. Through algebraic manipulation, $V_\infty = 69.3$ m s^{-1} and $F_C = 8.3$ N were calculated for steady flight. These values were then used in the simulator for an aircraft already in the air, and the result was a bounded system that approached $v_z = 0$ with a maximum error of <0.01 mm s^{-1}.

5 Experimental parameters

Our experimental runs used the following parameters: the UAV has a mass of 20 kg and a wing area of 0.2 m^2, corresponding to a wingspan of 1 m and chord length of 20 cm. It has a rotational inertia of 20 kg m^2. We initialize the UAV in low-level flight; it starts 25 m above the ground, with a 60 m s^{-1} x-velocity and 0 m s^{-1} z-velocity. To encourage landing in the early stages, we initialize the UAV pitched slightly downward, at ~ 3° below horizontal. The aircraft has full mobility through the space: it can climb without bound, and we observed some cases of the aircraft completing a full loop. The UAV is allowed up to 60 seconds of flight time, though the simulation is cut-off when the UAV touches down or when it reaches 200 m above ground level.

Our neural network used for control consisted of seven state inputs, five hidden units, and two control outputs. The state variables form the vector $\{t, \dot{r}_x, \dot{r}_z, r_z, \dot{\phi}, \sin(\phi), \cos(\phi)\}$ and are normalized using the ranges $\{t \in -10:70, \dot{r}_x \in 40:90, \dot{r}_z \in -10:10, r_z \in -5:30, \dot{\phi} \in -0.2:0.2, \sin(\phi) \in -0.5:0.5, \cos(\phi) \in 0:1\}$. We used five hidden units, because this allowed the search algorithm to discover successful policies while keeping the overall computation time low. The two control outputs are $\{F_C, M_C\}$, the force of thrust and the moment about the center of the wing. This force and moment is used for the dynamics of the time step. The UAV can provide thrust limited in the range 0:50 N, and a moment in the range 0:5 Nm.

In our implementation of MAP-Elites, we represented each individual $(\mathcal{I})$ as a set of weights for the neural network. We chose to use two phenotype dimensions with a resolution of 10 equally spaced bins

along each dimension. This results in 100 empty bins at the start of each experimental trial. We used 50 filling iterations and 1000 mutation iterations.

We examined a wide range of possible phenotypes, but found that the most effective phenotypes were those which averaged values across a moderate number of timesteps. For example, the final kinetic energy was a very sensitive phenotype; very small genotype (neural network weight) changes caused very large changes in the phenotype space. In contrast, using the average kinetic energy over the last 2–3 seconds (20–30 timesteps) proved to be more stable. In our results reported in this paper, we used the following two phenotypes:

- *Phenotype 1*: Average x-position over the last 2 seconds the UAV is in the air, with limits of 200:900 m.
- *Phenotype 2*: Average z-position over the last 2 seconds the UAV is in the air, with limits of 0:4 m.

We arrived at these phenotypic choices after a number of trials using other phenotypes including time-in-air, average x- or z-velocity over the last 2 seconds, final orientation, and average angle of attack over the last 2 seconds. We found that the final phenotypes we selected offer an interesting spread of behaviors for the following reasons: (1) preserving solutions which vary along the x-position before they land preserves solutions with longer-range approach behaviors which vary from conservative to aggressive, and (2) preserving solutions which vary in average z-positions before they land preserves solutions with close-to-ground behaviors which range from conservative to aggressive. We found that averaging over a short period of time before the UAV landed offered a more stable phenotype than simply choosing the final instantaneous velocity or position.

To calculate fitness of an individual, we use the angle at which the aircraft approaches the ground ('glide angle') and the factor i_L. The glide angle is calculated using the x- and z-velocities over the last 3 seconds in flight; this allows for a more consistent fitness calculation than using the final instantaneous glide angle. The i_L indicator returns a value of 0 if the UAV has landed and a very large negative number if it has not. The fitness calculation is shown below:

$$\mathcal{P} = \sum_{t\,=\,(\text{end}-3)\,:\,\text{end}} -\left|\text{a tan}\left(\frac{\dot{r}_{z,t}}{\dot{r}_{x,t}}\right)\right| + i_L \tag{9}$$

During our experiments, we noticed that the number of bins being filled were significantly lower than was expected and remained so no matter how we adjusted the outer limits of the MAP. We found that this was due to a co-variation between our phenotype variables. In order to achieve a wider variety of phenotype behaviors, as well as to allow the UAV to drastically change its behavior when near the ground, we developed a NGCS scheme. In this scheme an individual is described not by one set of weights for the neural network, but two separate sets of weights. The UAV uses one set of weights for the approach, and when it is <5 m above ground level and has a negative velocity in the z-direction, the second set of weights are used. This allows for the UAV to more easily learn the nonlinear behaviors that are necessary for a smooth landing; human pilots use 3° as a rule of thumb for the glide angle before 'flaring' when they are close to the ground. This flaring action consists of increasing their angle of attack in order to create more lift for a softer landing (Watson *et al.*, 1983).

We performed separate trials with and without wind effects. In the trials with wind, we created a random distribution around a sine function with amplitude 15 m s^{-1} (Figure 5).

6 Results

We conducted experiments by using MAP-Elites to search for controllers that provided a smooth landing in four cases: (1) no wind, no NGCS, (2) with wind, no NGCS, (3) no wind, with NGCS, and (4) with wind, with NGCS. Specifically, we show:

- A typical whole population of final controllers for each case (Section 6.1; Figures 7–10).
- Mean and standard deviation (μ, σ) of controller performance across 30 statistical runs (Section 6.2; Table 1).

- The effect of the choice of phenotype on final performance and recommendations for phenotypes in dynamic domains (Section 6.3; Figure 11).

In each of the figures, a black path with triangles denotes that the UAV did not land within the allotted 60 seconds, a red path with asterisks denotes a hard landing with glide angle >3°, and a blue path with circles denotes a soft landing with glide angle <3°. The plots reflect the UAV's x–z flight path.

6.1 Typical final population

The additional complexity of behaviors that can be learned by using the NGCS technique can best be seen by the flight profiles themselves. In Figures 7–10, we show the flight profiles of all members of the final population of a typical run for each method.

Figure 7 shows the final map population flights in the no NGCS, no wind case. The controllers that are developed can be easily described as 'pull up, at various rates'. Seven of the generated controllers do not actually land over the 60 seconds of flight time, and instead climb until they reach the height bound, ending the simulation. Even though they are penalized heavily for not landing, they are protected as they have a sufficiently different phenotype from the others.

Figure 8 shows the final map population flights in the no NGCS, windy case. These controllers are also easily described as 'pull up, at various rates'. In this case, the controllers are more proficient at making sure that their final z-location is at ground level, though this results in the undesirable behavior of 'climb, stall,

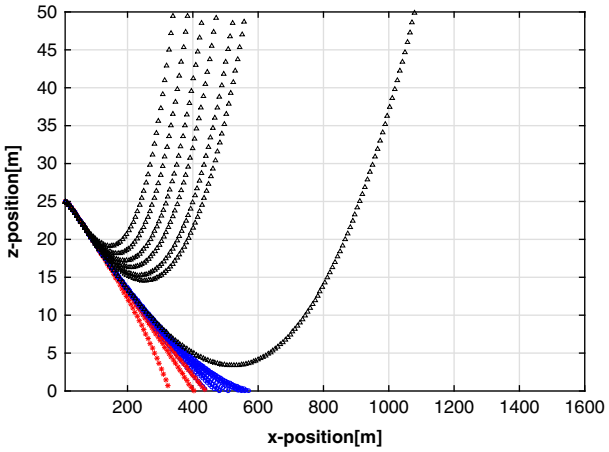

Figure 7 Case 1: Final flight profiles for a trial with no near-ground control switching, with no wind show the largest number of crafts that do not land

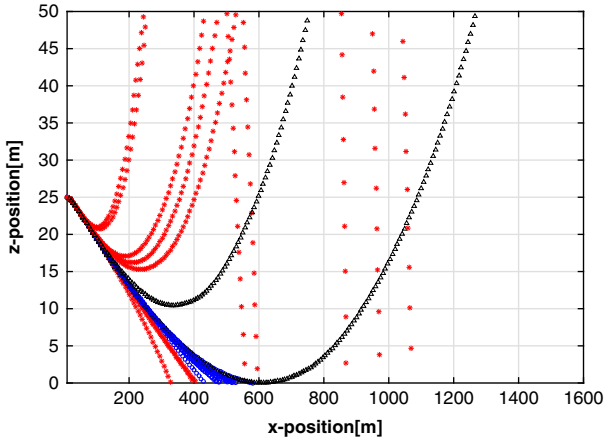

Figure 8 Case 2: Final flight profiles for a trial with no near-ground control switching, with wind

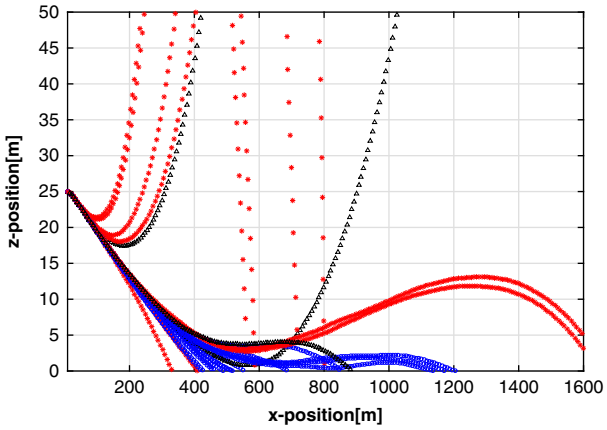

Figure 9 Case 3: Final flight profiles for a trial with near-ground control switching with no wind

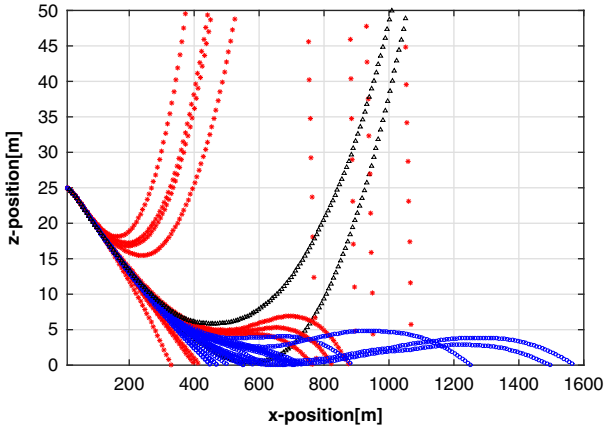

Figure 10 Case 4: Final flight profiles for a trial with near-ground control switching with wind show the largest number of crafts with soft landings

fall'. In an implementation, a system designer would use their knowledge to choose the best controller to implement, so the inclusion of these controllers in the final population is not a detriment. However, the stochastic nature of the wind showcases the primary weakness of this method: one controller gets as low as 0.05 m, but then continues to pull up and climbs without bound.

Figure 9 shows the final map population flights in the NGCS, no wind case. The phenotype protections still result in some controllers with the 'climb, stall, fall' profile; however, those that get close enough to the ground that NGCS takes effect have a much more sophisticated behavior than those without NGCS. These runs will level off, and sometimes briefly climb, but generally not without bound.

Figure 10 shows the final map population flights in the NGCS, windy case. These solutions are qualitatively similar to Figure 9, showing that the NGCS method in particular is good at rejecting the stochastic effects of the high winds.

6.2 Case profiles

We additionally performed 30 statistical trials for each wind, NGCS combination, as well as 100 flights with random control inputs. Table 1 shows the number of solutions generated and the mean and standard deviation $\{\mu, \sigma\}$ for the glide angle, landing z-velocity, and landing x-velocity. The values reported represent the values for controllers that landed with less than a 40° glide angle (to prevent 'climb, stall, fall' outliers from having a high impact on the σ calculations). In the random case, 60 of the 100 controllers landed, while 40 climbed without bound.

Table 1 Median number of solutions in the final map, mean, and standard deviation (μ, σ) of the landing glide angle, landing z-velocity, and landing x-velocity, for each wind/near-ground control switching (NGCS) combination and 100 randomly controlled trials

	I	II	III	IV	Rand
Wind	No	Yes	No	Yes	No
NGCS	No	No	Yes	Yes	No
Median no. of solutions	12	13	25	23	N/A
Glide angle μ (°)	2.34	2.30	2.28	2.34	13.56
Glide angle σ (°)	1.60	1.51	1.30	1.29	10.23
Landing z-velocity μ (m s^{-1})	2.83	2.82	2.92	2.99	16.04
Landing z-velocity σ (m s^{-1})	1.92	1.81	1.65	1.64	12.10
Landing x-velocity μ (m s^{-1})	70.89	70.74	73.59	73.26	65.60
Landing x-velocity σ (m s^{-1})	2.46	1.10	3.36	3.09	5.60

6.3 Recommendations for MAP-Elites in dynamic problems

Before concluding this work, we collate our experiences with MAP-Elites in a problem with a strong dynamics component, in order to provide guidelines for future researchers expanding the capabilities of MAP-Elites and related search algorithms in such problems. Our experiences have supported the following observations:

- *Phenotype stability*: When a small change in genotype leads to an extreme change in phenotype, we found that we did not achieve as high of a final performance. In a dynamic problem, averaging a few points around the time of interest—in this case the landing time—led to an increase in stability for the genotype–phenotype mapping. This leads to MAP-Elites performing more consistently across independent trials.
- *Phenotype coupling*: When phenotypes are coupled, the co-variation can prevent certain phenotype spaces in the map from being filled, or can preserve very low-fitness solutions for a long period of time.
- *Phenotype limits*: In this work, if a phenotype was outside our chosen limits for the map, it would be considered a part of the nearest bin. This had the benefit of not requiring any special handling for individuals with phenotypes outside of the limits, but also led to the preservation of some solutions that were well outside of the expected behaviors.
- *Strong nonlinearities*: If a desirable control scheme has strong nonlinearities or discontinuities, but the conditions in which these changes apply are easily described, defining an individual as a set of neural network weights can at least allow a greater spread in the phenotype space, if not increased average performance.
- *Selection for mutation*: Choosing the individual to mutate based on a uniform random over indexes will equally weight each individual; choosing based on a uniform random over phenotypes will weight those in rare portions of the phenotype space or those at the edges of clusters much more highly than those individuals that are surrounded in the phenotype space. We found a clear difference in selection probability, but not a clear difference in performance when testing both of these strategies. A fitness-biased selection (which we did not address in this work) will provide a more-greedy approach, which might not be beneficial (Lehman & Stanley, 2011).
- *Phenotype dimensions dictate behaviors*: Figure 11 demonstrates that a moderately different choice in phenotypes can have a large impact on the system behavior. Here, we substituted the final angle, ϕ (averaged over 2 seconds, with range $\pm 10°$), for the x-position phenotype. The behavior is very qualitatively different. Despite using NGCS, only three of the produced solutions in this run actually land; all of the others climb without bound. This change in behaviors is a result of only a change in the phenotype used within MAP-Elites, demonstrating the importance of phenotype selection.

7 Conclusions

In this work we examined MAP-Elites for use as a search algorithm to generate successful controllers for autonomous UAVs, even with wind disturbances. We discovered that the most useful phenotypes were highly coupled, limiting the population that could be supported. We partially addressed this by introducing

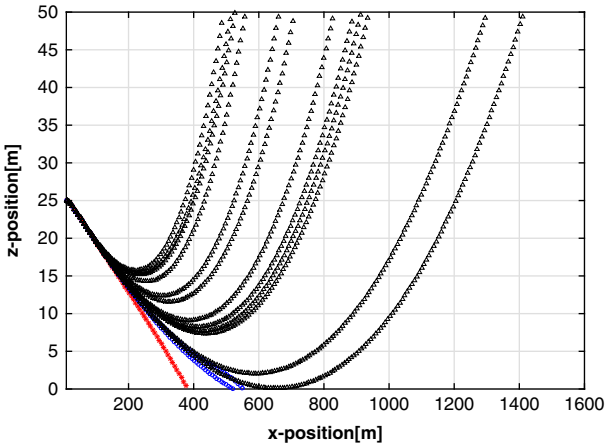

Figure 11 Final population flight profiles using the ϕ phenotypes

a second controller which is substituted when the UAV is <5 m above ground level and still traveling downward. This produced a wider variety of phenotypic behaviors and a median population twice as large. Our final controllers resulted in vertical landing speeds lower than dropping the aircraft from 50 cm above the ground. The softest landings had a glide angle of <1° and vertical speed of <1 m s^{-1}.

These studies were limited by simulation-based factors: first, our simulation does not account for near-ground changes in aerodynamics and wind. Second, our simulation does not account for aerodynamic forces on the body, or any part of the aircraft when outside of the range which we could calculate coefficients of lift and drag using XFOIL. Finally, our simulation does not account for any changes in air pressure with altitude.

Future work on this topic includes implementing such a controller in a 6 DOF simulator, working toward physical implementation. We suspect that the solution quality we produced can be improved upon by casting the problem within the framework of multiple objectives and incorporating a framework that will allow optimization over those multiple objectives simultaneously (Yliniemi & Tumer, 2014, 2015). Additionally, we are researching MAP-Elites for more complex control problems, like vertical takeoff and landing of fixed-wing UAVs; small UAVs typically have a large enough thrust-to-weight ratio to be physically able to perform this maneuver, but it requires a skilled pilot to perform.

Acknowledgements

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) Space Grant College and Fellowship Training Program Cooperative Agreement #NNX15AI02H, issued through the Nevada Space Grant. The authors would also like to thank the Nevada Advanced Autonomous Systems Innovation Center (NAASIC) for their continued support.

References

Alexis, K., Nikolakopoulos, G. & Tzes, A. 2011. Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances'. *Control Engineering Practice* **19**(10), 1195–1207.

Anderson, J. D. 2010. *Fundamentals of Aerodynamics*, 5th edition. McGraw-Hill Education.

Bauer, C. 1995. Ground state-fly state transition control for unique-trim aircraft flight control system. US Patent 5,446,666. 29 August. US Patent and Trademark Office `https://www.google.com/patents/US5446666`.

Baxt, W. 1991. Use of an artificial neural network for the diagnosis of myocardial infarction. *Annals of Internal Medicine* **115**(11), 843–848.

Cully, A., Clune, J., Tarapore, D. & Mouret, J. 2015. Robots that can adapt like animals. *Nature* **521**(7553), 503–507.

Drela, M. 2013. XFOIL 6.99 subsonic airfoil development system (software). Released under the GNU GPL. `http://web.mit.edu/drela/Public/web/xfoil/`.

Ecarlat, P., Cully, A., Maestre, C. & Doncieux, S. 2015. Learning a high diversity of object manipulations through an evolutionary-based babbling. In *Proceedings of the Workshop Learning Object Affordances, IROS*, 1–2.

Federal Aviation Administration (FAA) 2008. *On Landings Part II*. Technical Report FAA-P-8740-12 AFS-8 (2008) HQ 101128, FAA.

Federal Aviation Administration (FAA) 2015. *Registration and Marking Requirements for Small Unmanned Aircraft*. Technical report 80 FR 78593, FAA.

Foster, J. & Neuman, F. 1970. Investigation of a digital automatic aircraft landing system in turbulence. NASA Technical Note D-6066, National Aeronautics and Space Administration.

Gautam, A., Sujit, P. & Saripalli, S. 2014. A survey of autonomous landing techniques for UAVs. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1210–1218. IEEE.

Green, W. E. & Oh, P. Y. 2006. Autonomous hovering of a fixed-wing micro air vehicle. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, 2164–2169. IEEE.

Hornik, K., Stinchcombe, M. & White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* **2**(5), 359–366.

Jorgensen, C. C. & Schley, C. 1995. A neural network baseline problem for control of aircraft flare and touchdown. In *Neural Networks for Control*, Miller, W. T., Sutton, R. S., Werbos, P. J. (eds). A Bradford Book (March 2, 1995). 403.

Kim, H., Jordan, M. I., Sastry, S. & Ng, A. Y. 2003. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems*.

Lehman, J. & Stanley, K. O. 2011. Abandoning objectives: evolution through the search for novelty alone. *Evolutionary Computation* **19**(2), 189–223.

Lewis, F., Jagannathan, S. & Yesildirak, A. 1998. *Neural Network Control of Robot Manipulators and Non-Linear Systems*. CRC Press.

Lewis, M. S. 2011. *Beaufort Wind Chart – Estimating Winds Speeds*. Technical report, National Oceanic and Atmospheric Administration.

Li, G. & Baker, S.P. 2007. Crash risk in general aviation. *JAMA* **297**(14), 1596–1598.

Li, W. & Harris, D. 2006. Pilot error and its relationship with higher organizational levels: HFACS analysis of 523 accidents. *Aviation, Space, and Environmental Medicine* **77**(10), 1056–1061.

Mellit, A. & Pavan, A. M. 2010. A 24-h forecast of solar irradiance using artificial neural network: application for performance prediction of a grid-connected PV plant at Trieste, Italy. *Solar Energy* **84**(5), 807–821.

Mouret, J. & Clune, J. 2015. Illuminating search spaces by mapping elites. Cornell University Library, Preprint - April 21, 2015. *arXiv preprint arXiv:1504.04909*.

Nguyen, A. M., Yosinski, J. & Clune, J. 2015. Innovation engines: automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, 959–966. ACM.

Oncu, M. & Yildiz, S. 2014. *An Analysis of Human Causal Factors in Unmanned Aerial Vehicle (UAV) Accidents*. PhD thesis, Naval Postgraduate School.

Saeed, A. S., Younes, A. B., Islam, S., Dias, J., Seneviratne, L. & Cai, G. 2015. A review on the platform design, dynamic modeling and control of hybrid UAVs. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 806–815. IEEE.

Shappell, S., Detwiler, C., Holcomb, K., Hackworth, C., Boquet, A. & Wiegmann, D. A. 2007. Human error and commercial aviation accidents: an analysis using the human factors analysis and classification system.. *Human Factors* **49**(2), 227–242.

Shepherd, J. III & Tumer, K. 2010. Robust neuro-control for a micro quadrotor. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 1131–1138. ACM.

Watson, D. M., Hardy, G. H. & Warner, D. N. Jr 1983. Flight-test of the glide-slope track and flare-control laws for an automatic landing system for a powered-lift STOL airplane. NASA Technical Paper 2128, NASA Scientific and Technical Information Branch.

Yliniemi, L., Agogino, A. K. & Tumer, K. 2014a. Multirobot coordination for space exploration. *AI Magazine* **4**(35), 61–74.

Yliniemi, L., Agogino, A. & Tumer, K. 2014b. Simulation of the introduction of new technologies in air traffic management. *Adaptive Learning Agents, Part 3. Connection Science* **27**(3), 269–287.

Yliniemi, L. & Tumer, K. 2014. PaCcET: an objective space transformation to iteratively convexify the pareto front. In *10th International Conference on Simulated Evolution and Learning (SEAL)*.

Yliniemi, L. & Tumer, K. 2015. Complete coverage in the multi-objective PaCcET framework. In *Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 1525–1526.