2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Crowding and Mutation Improvements in an EA for Flight Control Correction in a Flapping-Wing Vehicle

John C. Gallagher
Electrical and Computer Enginnering
University of Cincinnati
Cincinnati, OH, USA
john.gallagher@uc.edu

Michael W. Oppenheimer
Autonomous Control Branch
Wright-Patterson Air Force Base
Dayton, OH, USA
michael.oppenheimer@us.af.min

Eric T. Matson
Computer and Information Technology
Purdue University
West Lafayette, IN
ematson@purdue.edu

Abstract – Small Flapping-Wing Micro Air Vehicles (FW-MAVs) may be subjected to either or both of manufacturing defects or in-service damage that render their pre-designed controllers less than adequately effective. Even minor damage to wings, for example, can remove the vehicle's ability to reliably follow waypoint trails even if that same damage does not result in a catastrophic loss of altitude. One solution to this problem is to adapt the core wing motion scripts (wing gaits) in an attempt to use wing motion to compensate for losses of force and torque generation due to in-service damage or manufacturing faults. This approach presents a number of challenges - especially if it is to be deployed in a resource restricted vehicle in an online mode during an actual mission. Prior to this paper, we had presented only anecdotal treatment of fully unrestricted, 3D flight. In this paper, we will definitively establish the utility of EA adaptation of flight control in unrestricted flight in a pendulum-stable FW-MAV. We will, additionally, introduce mutation and crowding modifications that provide demonstrable utility in a manner amenable to implementation on a resource-restricted micro vehicle. The paper will conclude with a discussion of open-issues and the potential application of the reported methods to other problems.

Keywords—Flapping-Wing Micro Air Vehicle; Evolvable and Adaptive Hardware; Evolutionary Computation; Adaptive Control

I. INTRODUCTION

Consider a person treading water in a swimming pool by sculling his arms across the surface of the water in rhythmical arcs while his body extends straight down into the water and perpendicular to the bottom of the pool. Assuming the arms swept out fully-symmetric strokes aft to fore and further assuming the strokes were identical between the two arms, when averaged over a whole sweep cycle, all forces and torques except for a single force pointed up and out of the pool should cancel out. In a sense, the person could "hover" in place in the water subject only to a small limit-cycle around the fixed point that would occur during the arm sculling. One could further imagine introducing selected and planned asymmetries in those arm sculling motions to introduce additional net forces and torques to not only support one's self at the surface of the water (lift), but also to produce net torques that allow spinning in place (rolling) or moving forward or backwards along the surface of the water (translation). In the seminal works [1] – [3], the above strategy was formalized as a means to control a small FW-MAV. An assumption was made that the wings would flap with cosine gaits (motion patterns). Knowing that pattern, and assuming symmetry wing-to-wing, one could produce a model of lift and determine a frequency with which to flap to counteract gravity or move up and down in a controlled way. Likewise, one could introduce shape parameters that modify the underlying cosine motions differentially across the body to produce roll and/or translation. Analytic model-based force and torque controllers that compute desired motion shape parameters and frequencies can be embedded in fairly traditional waypoint controllers to produce full flight control. Once per wing flap, the waypoint controller would determine a desired relative position, the individual roll, translation, and lift controllers would produce their suggestions for frequency and/or shape parameter, these would be communicated to an allocator that would decide what is most important to correct, and the determined parameters would be sent to the wing motion controllers. Over the next wing flap, efforts would be directed at "correcting" the vehicle in space to be closer to what is desired. This would repeat once per wing flap. The method is, in our opinion, somewhat elegant and is definitely amenable to implementation in simple on-board hardware.

What, however, happens when the derived models internal to the roll, lift, and translation sub-controllers are not of sufficient fidelity? Informal qualitative observations suggest that the loss of even a few percents of what wing force production is expected by the models prevents reliable waypoint tracking. These observations were more formally explored using model-checking methods in [4]. Perhaps surprisingly, though the core control is brittle in ability to track specific waypoints, it is robust in keeping the vehicle in the air – if at the incorrect position. If one doesn't mind being at "the wrong place", then the core model-based controllers can keep the vehicle aloft even for somewhat significant wing force production deficits of tens of percents. That one could stay in

the air at all under a fairly wide range of wing damage cases suggested to us that it would be possible to apply online machine learning to the core axis controllers (roll, altitude, and translation) to adapt to ongoing wing damage and restore the ability to properly track waypoints. The question, of course, is just how to accomplish this feat.

From a hardware perspective, one must immediately presume significant computational constraints. Especially at insect-scales, there simply is not room for complex computational hardware or power sources to run it. Any methods used must be amenable to implementation in very simple, very small, and very energy efficient hardware. From a machine learning / evolutionary computation perspective, the challenges are compounded by the fact that learning "new axis controllers" – in whatever form that takes – represents a hugely non-stationary online optimization problem. Measurements of position accuracy would by definition be corrupted by some level of random noise. If the learning is being done online and the vehicle is flying normal missions, one can expect some level of cyclic large-scale behavior (I.E. hover station keeping, followed by transit, followed by hover station keeping, etc.). We now have that form of non-stationary optimization. Perhaps most insidiously, the random ordering of optimization candidates effects the scores. Presenting for evaluation a highly effective solution AFTER a very poor solution will taint the score of the highly effective solution simply because that poor solution leaves the vehicle in such a bad state that the great controller can't possibly fix it in its given evaluation window. We now have presentation-order instability. Note that, even with these challenges present, we still must complete any learning without crashing the vehicle and in an amount of flight time that is reasonable to end users. Taking several months of flight time to correct behavior is, for example, not acceptable under any practical circumstances.

In previous work over the years, we struggled to improve learning efficacy and learning speed in light of the significant hardware and learning-environment challenges summarized above [5]-[9]. The referenced work represents a wide variety of approaches that were unified in two significant respects. The first was that they could be arguably be implemented using limited hardware resources and the second was that they employed EA and EA-like methods to adapt the core wing motion functions as a means of compensating for wing damage (I.E. learn basic gait functions that, when combined with broken wings, again produced forces that were compatible with the system models already implicit in the controllers instead of trying to relearn the models themselves). Only very recently were we confident that we had a learning method capable of providing such correction when the vehicle was flying unconstrained trajectories in 3D space [9]. Our previous report of success with the new algorithm, however, was largely based on anecdotal observation and lacked any significant amount of analysis into algorithm operation. Further, that report focused on specific modifications required for the method to function with the latest generation of FW-MAV which places restrictions on legal wing gaits and, thus, required significant reworking of the underlying learning methods. This paper will take that new algorithm as a base and, after a brief presentation of problem specific concepts and discussion of previous work, provide the following further analyses and introduction of additional modifications:

i. One can easily hypothesize that many critical performance metrics (time to achieve of an acceptable solution, percent yield of acceptable solutions, online performance, etc.) would correlate strongly to the size of the population. This paper will quantitatively explore those effects.

ii. Smaller populations may show some benefits related to efficiency of hardware implementation and less flight time being spent culling out poor candidate solutions. On the other hand, they may suffer deliterious effects from the pre-mature loss of population diversity or simply lack of sufficient diversity at the outset of optimization. After conisdering the results of (i) above, this paper will explore the use of a genotypic crowding-metric survivor selection to combat any issues that may have been uncovered.

iii. As the work presented here represents what is in our opinion the first fully-functional method that achieves appropriate balance of all our concerns across-the-board for online FW-MAV adaptation, this paper will more comprehensively address the next steps enabled by having met this milestone.

II. Background and Related Work

A. The Flapping-Wing Vehicle

Detailed descriptions of kinematics, dynamics, basic control, and basic evolved wing gait oscillator adaptation can be found in [1] – [3]. Figure 1 provides an orthographic view of the simulated vehicle. Each wing can be independently and actively rotated around its wing root. We refer to the angular positions of the left and right wings around its root as ϕ_L and ϕ_R respectively. For this work, the wing is modeled as being hinged at the shoulder and it rotates to an attack angle of $\pm\alpha$ under the influence of air pressure the wing sweeps through its ϕ range. The vehicle itself may move freely into any position or pose in 3D space under the influence of gravity and lift and drag forces generated by the wings.

The vehicle is guided along a waypoint path under the control of a "three-level" controller (Fig. 2). Once per wing flap, the path planner would produce a desired relative roll angle, relative altitude change, and relative translational change. These desires are communicated, as appropriate, to each of a model-based controller for each of the vehicle roll angle (Body Roll Command Tracking Controller), altitude (Body X Command Tracking Controller), and lateral translation (Body Z Command Tracking Controller) respectively. Each of these three tracking controllers computes desired body frame torques and forces and, using an inverted model of the vehicle, generates a flapping frequency (BXCT) and candidates for shape parameters (BRCT and BZCT) to modify the *assumed* cosine wing gaits stored in the lowest level wing motion controller (left and right split-cycle oscillators in Fig. 2). The allocator chooses from among the

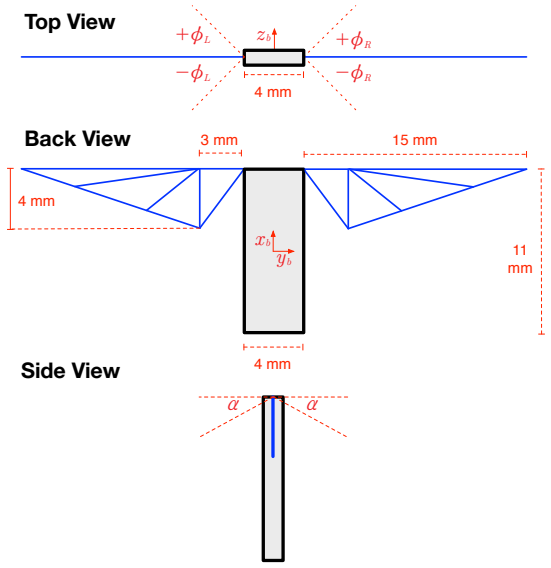

Fig. 1. Orthographic View of Conceptual FW-MAV.

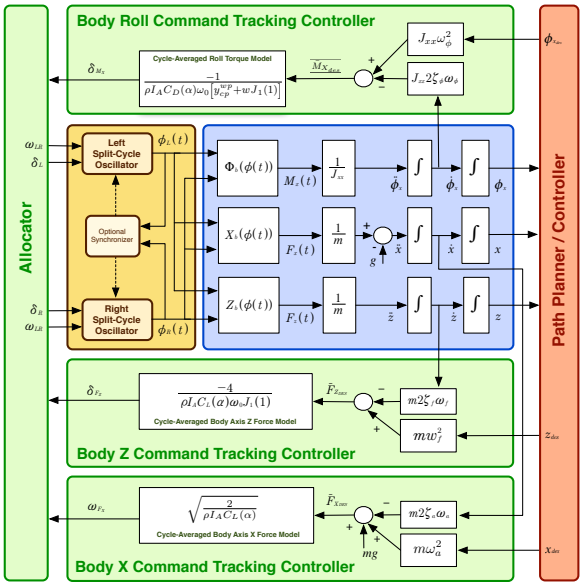

Fig. 2. – Control Schematic

shape and frequency parameter alternatives provided by the three model-based tracking controllers and instructs the left-wing and right-wing oscillators to produce the required modified cosine wing gaits on each wing. Due to mechanical constraints imposed on the current generation of the physical vehicle, the two wings must meet at the full forward angular position at the beginning and end of each wing beat. They must also move monotonically either to the front or back until they achieve maximum magnitude, then they may reverse. There is no strict condition that the wings meet simultaneously at full backward extension. Of note here is that the standard wing motion (split-cycle oscillators in Fig. 2) controllers have cosines hard-coded in them and both the shape and frequency parameters communicated to them modify the base cosines as required.

The introduction already discussed what could go wrong and suggested what we should do about it. One could attempt to learn the system models inside each of the three tracking controllers (horizontal green boxes in Fig. 2), but this, as an exercise in online system identification, might incur significant computational expense. In previous work, we employed various EAs, in an online mode, to evolve the *core oscillation functions inside the wing oscillators* instead. The idea was to match the force generation behavior of the wing/oscillation combination to vehicle models in the control rather than match the models to what the broken wings were doing when driven by cosines. Various reports or historical efforts are available in [5]-[9]. For this work, we start from the most recent and, prior to what we will report here, most successful attempt reported in [9].

III. The MINIPOP EA

In the Base MINIPOP EA Method, we represent wing motion functions as lookup tables of 256 individual wing angle positions each stored with a precision of eight bits. This choice is consistent with the limitations of the actual hardware used in several generations of the physical implementations of these vehicles. The core controller, without learning, would therefore store cosines in these tables and the wings would be actuated by playing back those values at different speeds (frequency shape parameter) or by introducing timing offsets inside the table to distort the cosine according to the needs of the body axis controllers (delta shift shape parameter). Our learning method allows an EA to evolve these wing motion tables (wing gaits) to optimize a performance based objective function. For this purpose, we maintain a library of eight precomputed "basis functions" (Fig. 3) that can be blended at run time to generate a variety of periodic functions that can replace the otherwise hard-coded cosine functions in the split-cycle oscillators. Each possible base oscillation function is specified by an array of eight integers each of which corresponds to the identity of one of the basis functions. The function used is the average of the identified functions, which can be computed quickly at run time with integer arithmetic. For each of 256 angular positions of the desired function, one adds the values from eight identified basis functions and then logically shifts right twice to divide by eight. This composed function would then be used as the core wing gait. Why we chose the specific bases shown is beyond the scope of this paper, that said, those choices were influenced by safety concerns and mechanical constraints of the current generation physical vehicle. The genome we use to encode wing motions is also shown in Fig 3. Each genome is the concatenation of the base function indices for each wing (eight values for each of the two wings shown in numbered LF and RF blocks of Fig. 3) as well as a learned gain multiplier that would be applied to the frequency given to the oscillators by the allocator (shown in the FM box of Fig 3.). The use of an evolved gain was also added in response to constraints imposed by the mechanisms of the current generation physical vehicle and represents a way around the problem of not being able to reverse motions in a wing flap in process. This physical wing-motion requirement is more completely discussed in [9].

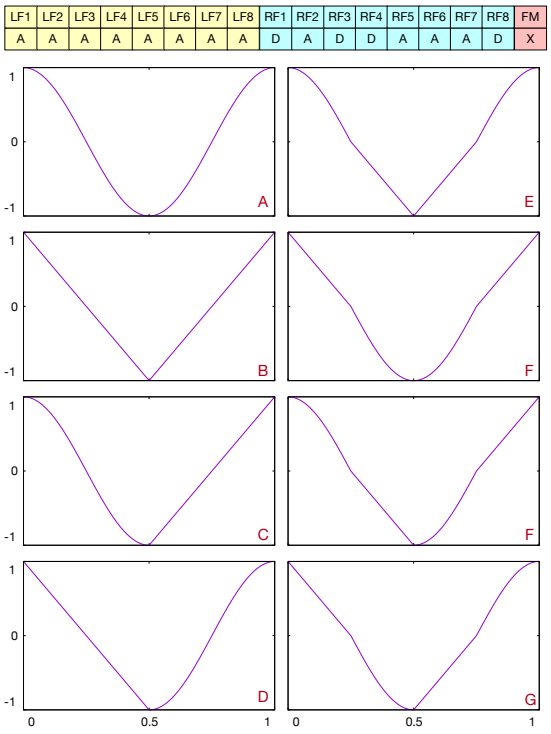

Fig. 3 –Basis Functions and Genome Encoding

```
1  done_evolving = FALSE;
2  initialize_population(&population);
3
4  // By convention, the champion genome will always be stored
5  // in slot zero of the population.  Slots are indexed from 0
6  // to POPULATION_SIZE-1
7
8  while (!done_evolving)
9   {  online_champ_error  = 0.0;
10     champion_index      = 0;
11     challenger_index    = uniform_random(1, POPULATION_SIZE-1);
12
13     champion_genome     = get_genome(&population,champion_index);
14
15     challenger_genome   = get_genome(&population, challenger_index);
16
17     child_genome        = uniform_crossover(champion_genome,
18                                             challenger_genome,
19                                             crossover_rate);
20
21     child_genome        = mutate(child_genome, mutate_rate);
22
23     champion_error      = evaluate_error(champion_genome);
24     online_champ_error += champion_error;
25
26     challenger_error    = evaluate_error(challenger_genome);
27
28     champion_error      = evaluate_error(champion_genome);
29     online_champ_error += champion_error;
30
31     child_error         = evaluate_error(child_genome);
32
33     // If the child is better than the champion, copy the
34     // child genome into the champion slot of the population
35
36     if (child_error < champion_error)
37       put_genome(child_genome, &population, champion_index);
38
39     // If the child is better than the challenger, copy the
40     // child genome into the population slot that had been
41     // occupied by the challenger, then do nothing else to
42     // the population
43
44     if (child_error < challenger_error)
45        put_genome(child_genome, &population, challenger_index);
46
47     // If you are doing the following branch, then the child was not
48     // better than the challenger.  Check to see if the champion is
49     // better than the challenger. If it is, copy the champion genome
50     // into the slot in the population that the challenger had
51     // occupied
52
53     else
54        if (champion_error < challenger_error)
55           put_genome(champion_genome, &population, challenger_index);
56
57
58     // Look at the average of the two champion evaluations done in
59     // this trip through the loop.  If this "recent online champion
60     // fitness" is less than a target error level, then set the
61     // done_evolving flag to true so that the evolution loop can be
62     // exited
63
64   online_champ_error = online_champ_error / 2.0;
65   if (online_champ_error < ERROR_TARGET) done_evolving = TRUE;
66  } // end while loop
```

Fig. 4 – Pseudo-Code for Basis Function Learning EA

Fig. 4 provides pseudo-code of the MINIPOP method used in [9] and also, in modified form, in this paper. It is a small population stochastic hill climbing, non-generational, EA that employs a strict form of elitism and periodic presentation of the current champion as a means of combating deceptive candidate evaluations cause by the serialized presentation of candidates in an ongoing task. In an offline learning process, we would "return" the vehicle to a low error state before presenting a truly novel candidate solution so as to not corrupt the evaluation of that novel candidate with blame inherited from a previously presented candidate. For example, a previous candidate could put the vehicle so far from a desired path that it couldn't completely recover even if a current candidate solution were of high quality. That current candidate would inherit some of the blame for the previously presented terrible candidate. One could, between evaluation candidates, use a perfect controller to reset the system to a good state to "clear the palate" between candidate evaluations. Of course, we don't know what that perfect control solution is and if we did – we wouldn't be running an EA anyway. Therefore, as a heuristic we allow the current champion to. Note that every call to govern the vehicle for a period between candidate evaluations. `evaluate_error()` in Fig 4. consists of sending the oscillator patterns represented by the genome to the vehicle and allowing that controller to control the vehicle for a user selected number of wing flaps before returning. The objective function used is the absolute positional error between the vehicle's position at the end of the evaluation period and the waypoint that is being attempted.

Of most interest, and novel to this paper, are evidence-based modification to the mutation operation at line 21 of Fig. 4 and more careful study, and potential modification of, the crowding-style survivor replacement mechanism implicit at lines 44 and 45 of Fig 4.

In the one previous work where the current version of the MINIPOP EA and presented genome encoding had been used [9], we encoded the frequency gain (FM in Fig. 3) as a float type and evolved that setting as a continuous value with a Gaussian mutation. In the real vehicle, the idea of continuously-variable gain settings is unrealistic, as we would most likely implement that capability as discretized values achieved by discretized clock frequency manipulations. Therefore, in this work we limited gain values to eight discrete levels of gain ranging from 1.0 to 1.4375 with a step size of 0.0625. We still employed Gaussian mutation with a standard deviation of 0.125 – though with actual achieved gains being binned into one of the legal values. As it is not clear that the EA would reliably find solutions under these conditions, through testing was obviously required.

Also in [9], we implicitly employed crowding-style replacement. In lines 45 and 46 of Fig. 4., we can see that the winner of the child/challenger tournament occupies (replaces) the challenger slot in the population. Crowding mechanisms attempt to promote extended population diversity and, possibly, formation of niching, by ensuring that selected survivors replace population members that are most alike to them as measured by some genotypic or phenotypic metric. We generate candidate children via crossover of a challenger and the champion that is then mutated. Assuming that the resultant child is *similar* to the challenger, then putting the winner of a challenger and child into the challenger's slot represents a simplified crowding metric. When we first implemented this idea, it was justified on an ad-hoc basis. Here we will compare this implicit crowding to both an explicitly computed crowding metric and to random replacement in an effort to assess the wisdom and utility of that previously ad-hoc decision.

IV. Experimental Conditions and Definitions

All runs reported here employed the following parameter settings: Population size is swept through a range of 8 to 88 with a step size of 8. Wing Flaps per Evaluation is set to 120, which represents approximately one second of flight time when the vehicle is at hover. Mutation probability is set to 3.125% - which means that any slot of the genome has that percent chance of mutating. Basis function slots mutate to any of the eight legal basis function index values with uniform probability. Gain mutates as previously described. The uniform crossover probability is 50%, meaning that each allele position of a child has a 50/50 chance of receiving its value from one of the two parents. Each evolutionary trial reported was ran until the average online error of the champion, collected over one trip through the algorithm loop starting at line 8 and ending at line 66, was less than 0.001 meters (1 mm) OR when five thousand evaluations of candidates and champions had been completed. Those five thousand evaluations would correspond to approximately seventy minutes of flight time. Evolutions and candidate evaluations are conducted online while the vehicle is attempting to follow a path with 70 waypoints as quickly as possible, but no longer than 70 minutes flight time. The target flight path was designed to have multiple altitude changes as well as a large number of distances between points and sharpness of turns required. When a trial is run, each of the wings of the vehicle was assigned a random damage coefficient between 0.6 and 1.0 that is multiplied by the amounts of drag and lift forces produced by that wing. Modeling of wing damage in this way, on a cycle-averaged basis, is supported by arguments made in [10] and [11].

For purposes of results evaluation, we define three disjoint sets of qualitative behavior into which all evolved solutions may fall into. These are: *Full Failure* – Solutions that are full failures will not approach the first waypoint within 1 mm, in any spatial dimension. In fact, they most often orbit some random point far from the first waypoint or fly off to infinity and beyond. *Partial Failure* – The vehicle can approach and maintain desired altitudes withing 1mm of the first waypoint, but it cannot maneuver to subsequent waypoints in the world XY plane; *Full Success* – The vehicle during evolution makes

Fig. 5 – Yield Rates and Median Time to 2nd Waypoint for All Replacement Variants

it to at least the second desired waypoint. For purposes of yield results reported below, verification of failure or success status was determined by exercising the champion genome *after evolution had completed*. As actual evolutionary learning is online and it is not switched off so that continuous improvement is desired, we use as our "time to success" a control achievement that predicts eventual full success of whatever champion genome is present at the end of the trial. We note that the ability to reach the 2nd of the desired waypoints during evolution is a strong predictor of *Full Success* and thus for current purposes use "2nd waypoint time" as the time when the vehicle's flight becomes viable – even if additional learning can occur. Fewer than 1% of learning trials that acquired the 2nd waypoint, over all of the learning experiments presented in this paper failed to achieve a champion capable of full success. Possible reasons for this observation will be discussed in the conclusions.

When we use an explicitly computed crowding metric, it is computed as follows: We first flattened each wing's eight elements to a discrete probability distribution over the relative appearances of each of the eight possible basis functions. The distance between any two genomes was computed as the Euclidian distance between the two distributions. We chose this distance metric because the calculation could be accomplished with a modest number of integer additions, shifts, and comparison operations.

V. Experimental Results

Before delving into detailed investigations, it may be prudent to at least qualitatively assess the need to evolve both wing motion patterns (gaits) and oscillator gains -- as it is not immediately apparent adapting both simultaneously is required to restore proper flight control. For the experiments underlying this preliminary discussion, we set the population

size to sixty-four and left other parameters set as previously indicated.

Over five thousand trials, when evolving wing gaits only while freezing flapping frequency gains to 1.0, the vast majority of evolved solutions were *Full Failures.* Under this condition, *full successes* were rarely seen when the damage to each wing was very slight (no more than 2%) as presumably wing gait modification can provide at least some mild lift enhancement and the drag forces are sufficiently balanced, wing-to-wing, to enable controlled motion in the XY world plane. Under this condition *partial failures were* also rarely seen when damage to the wings was slight, but sufficiently unbalanced as to ruin XY plan navigation. Over another five thousand trials evolving gain only while leaving the wing motion functions frozen at cosines, we again saw the vast majority of results being *Full Failures* or *Partial Failures*. In this set, the dividing line between Full and Partial Failures seems to be whether or not one of the selectable, discrete, gain levels was sufficient to boost lift enough to get one close to a desired altitude. In the rare case that this condition is met and when, additionally the deficits across the wings are very similar (I.E. the damage is balanced), then *Full Successes* could be observed. These ten-thousand trials strongly suggest that in the vast majority of cases, both gait and gain learning are required for success. With this need established, at least empirically, we can move on to exploration of the full algorithm with gain and gait learning enabled.

Fig 5. presents performance statistics for the presented MINIPOP EA with discretized gain mutation and three different survivor replacement options: random replacement (random), implicit crowding replacement (i_crowd), and genome distance crowding replacement (g_crowd). Perhaps the statistic of most relevance to end users of a FW-MAV is the failure rate, or the percentage of runs that did not achieve fully successful waypoint flight control. Note that at smaller population sizes, both crowding variants have smaller failure rates than the random replacement. Also note that, at least in terms of yield, the two crowding variants are indistinguishable from each other. Looking at median time to achievement of the 2nd waypoint, which is our chosen predictor of eventual full success, we see a clear winner emerge. The implicit crowding method, which corresponds to the original choice made in [9], is consistently faster in achieving the 2nd waypoint success condition under identical meta-parameter settings across the three variants tested. The collected data suggest that to achieve a failure rate of 2% or less one must set population size to 32 or greater, with the price of higher yields after that being paid by longer expected times to flight success. At this point, we can conclude that the algorithm, as presented with discretized gain levels, multiset encoding of the wing gaits, and meta-parameters set as described can restore flight control at least 98% of the time in eight minutes or less 50% of the time (Q2) and in less than eleven minutes 75% of the time (Q3). This is within acceptable limits for practical use of the method and this is the first time we have established that with large data sets for fully unconstrained flight. Of course, there are additional observations we can make if we dig more deeply, and these observations may lead to additional improvements.

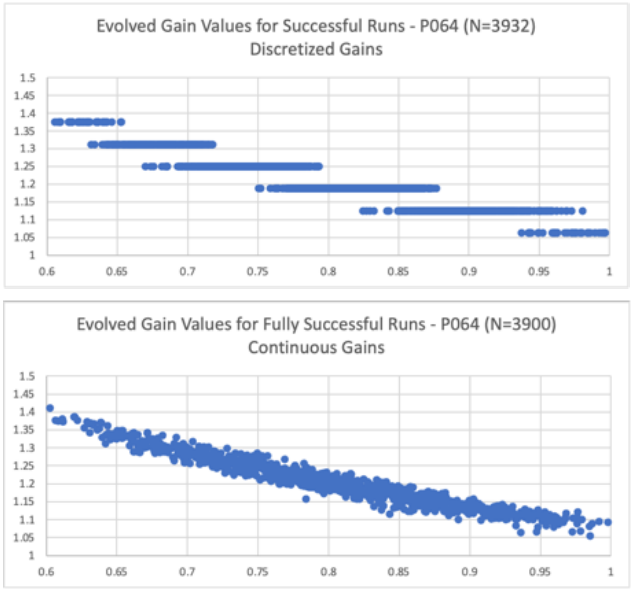

Fig. 6 – Evolved Gain Values for Discretized vs. Continuous Gain Options

VI. Additional Observations of Evolved Solutions

One could, and perhaps should, tune other EA parameters to decrease learning time and increase yield. Simulations are expensive to run, however, so sweeping through search parameter spaces blindly may not be the best choice. So, let us see what problem space information we can glean from experiments already run and speculate on how that information might inform what we do going forward.

First, let us examine what happens when evolve gain levels are discretized, like we did in this paper, vs. what we see when we evolved continuous valued gain values [9]. Fig. 6 shows graphs of average wing damage multiplier across the two wings vs. the actual gain value that evolved. Evolved gain is shown on the y-axis and average wing damage across the two wings is on the x-axis. Both graphs are based on 4000 trials at population size 64 with only the *fully successful* individuals shown. Note that if we allow for fully continuous gains to be evolved, then we will see a continuously variable selection of gaits will be evolved. Looking at the graph for the discretized gains reveals something interesting, however. Note that for large sections of average deficits, there are viable, fully successful, solutions that draw from more than one possibility for gain value. One such region runs from approximately 0.7 to 0.75 and in that region gains of 1.25 and 1.1875 are both viable, presuming of course, they are paired with an appropriate selection of evolved wing gaits. Although one might be tempted to think that the wing gait adjustments' primary value is in balancing drag forces so that world XY plane navigation functions – the above provides some evidence that boosts in lift are also possible via changes in gait. If this is true, one might argue that at least in some cases gain evolves to get altitude control "in the ballpark" and that gait evolution provides additional fine adjustment of both altitude as well as roll and translation control. It also implies

that the amount of lift correction that can be provided spans several of the levels of lift correction that can be achieved by gain. We can presume, therefore, that this problem space is somewhat target rich and that there are potentially many genotypes – even those using significantly different gain values – that correct for the same wing damage levels. To test this idea, we created a variant of the simulation code that attempted multiple independent evolutionary runs for each damage case. At the time of the writing of this paper, we have conducted six hundred damage case trials where each damage case was given ten independent evolutionary runs. These trials were run at our presumed optimal parameter settings of implicit crowding and population 32. All of these trials had *at least* 2 of 10 full successes with the vast majority of them having 9 or 10 full successes. Further, we observed ranges of successful gain values as well as ranges of wing gaits all being successful for the same wing error conditions. Although this is preliminary evidence only, it does strongly suggest that failures to find solutions are more likely failures of the search algorithm then they are indications that solutions don't exist. The representation supports plenty of solutions, we just need to find them. These initial experiments also suggest several other EA enhancements that could reduce failure rates while still keeping population and time to success values small. Adaptive hypermutation is one possibility. A more sophisticated crowding metric might be another. Although our implicit crowding currently seems the best choice, there is no reason to think it is the ultimately best one.

Second, let us turn our attention now to anecdotal observations about *partial failures* that occur when using discretized gains. *Partial failures*, regardless of which EA variant (base or crowding) produced them, end up with genomes that are both highly symmetric across the wings (I.E. the leaned gait functions are highly similar) and which are dominated by basis functions A and B (Fig. 3). This implies that the *partial failures* overwhelmingly learn to adopt gait functions that are very close to the "normal" cosine function or to a fairly similar triangle wave. In other words, in terms of gait, the *partial failures* learn to do what the model-based tracking controllers expect as being correct for undamaged wings. Understanding this very consistent pattern in the phenotype underlying partial failures will be instructive in lowering the number of partial failures and increasing overall yield of viable solutions.

VII. Discussion and Conclusions

We had made previous report of the success of online evolution for restoring waypoint control in fully 3D flight [9]. That report, however, was limited in potential practical application in that we previously allowed evolution of real-valued gains. This would be impractical in our eventual use case in which a requirement to implement in simple computational hardware (presumably without floating-point capability) would be paramount. Also, the assumption of continuously variable gain control is problematic in its own right. The algorithm version presented here overcomes those final practical limitations. We also verify that crowding is crucial to promoting acceptable yields while minimizing computational resources. It is therefore not unreasonable to conclude that we have delivered on promises (i) and (ii) from this paper's introduction. What about promise (iii)? Frankly, this is the more interesting discussion, so let us proceed to that.

First, the results here suggest that crowding has some utility for this problem. We have yet to beat the implicit crowding method, but there is no reason to think it can't be beaten. The genotypic crowding metric we used does account for the redundancy effects of a multiset encoding as all redundant genome variants would map to the same Probability Distribution Function (PDF) that serves as the endpoint for whatever distance metric is used. On the other hand, it does not properly represent distance in the phenotypic sense. The genome PDF distance between a purely cosine gait function (Basis A), as represented by the genome, and a purely triangle gait function (Basis B), as represented by the genome, is the same as the distance between a pure cosine and something that was made purely of Basis E even though we know that phenotypically Basis E is closer to Basis A than is Basis B. The obvious fix for this is to compute the distance metric for crowding replacement by comparing the phenotypes directly. As this would entail computing and scanning through 256 element, rather than 16 element, tables, it would be more expensive – but perhaps not prohibitively so. This is work under way and it might provide better overall results at a larger, but perhaps still acceptable, cost in onboard computational hardware.

Second, the persistent commonalities in the phenotypical structure of all *partial failures* are obviously of interest. We have an as of yet untested, but perhaps viable, hypothesis on this matter that draws from observations made elsewhere in this paper. The vehicle we are modeling here flies in a *pendulum stable* mode. That means it flies with its body frame x-axis (see Fig. 1 for axis definition) is parallel to the world z-axis (world frame altitude). For a vehicle with the bulk of its weight below its wings flying relatively slowly, this is a reasonable assumption. Under these conditions, the control of altitude is largely decoupled from the control of other degrees of freedom. This can also be gleaned from an examination of the control diagram in Fig. 2. What does this mean from an EA perspective? Even if our objective function does not reflect it directly, the nature of the problem itself is somewhat multi-objective due to the nature of the controller. Even if *we* are measuring a scalar objective, ultimately the controller is implicitly solving altitude control as an issue separate from roll and forward XY plane position control. Note in Fig 2. That the axis controllers only ever provide one gain and that allocation is only necessary for the delta shift alternatives being provided by the body z and body roll sub-controllers. Since our scalar objective function collapses quality of altitude control and everything else into one value (I.E. distance to target waypoint in three-space) one could evolve wing gait functions that, in combination with a specific gain value, are so effective at solving the altitude problem that any searches off of that local "single objective" optimal are of sufficiently low quality to make any moves off

that localized optimal non-viable. Interestingly, for nearly all such partial successes, we see the EA adopt wing gaits that are very similar to the cosine gaits the core controller was designed to work with. A perhaps simple expedient would be to promote symmetry breaking during evolution by encouraging the use of asymmetric wing gaits across the two wings. Multiple mechanisms to accomplish this exist, including using a replacement strategy that promotes asymmetry of gaits across the wings. The use of this and similar expedients, along with a detailed analysis of partial success cases and how they arise, is currently underway.

Third, recall that we observed that reaching the second waypoint during learning was a strong, but not perfect, predictor of if the eventual champion genome achieved at termination of learning were capable of providing quality flight control using ONLY the champion genome going forward. In fewer than 1% of all cases, this was not true and, although with learning switched on the vehicle could follow waypoints, it could not with learning switched off. This implies that in rare cases, the population of wing oscillator solutions actually contains sub-niches that are situationally appropriate for specific flight modes and that are dynamically selected as needed during flight as a side-effect of sampling the population. We could detect this by monitoring flights on an ongoing basis and noting if specific champion genomes appeared on a cyclic basis correlated to specific modes of flight. If this were occurring, we would have essentially created a hybrid-state machine controller in which the refinements were alternative wing motion oscillator settings and the state switches were mediated by stochastic selection and evaluation of state connected performance. Under some conditions, we might actually desire controllers, so this possibility is also currently under study so that we may encourage or discourage its evolution according to the desires of the end users.

Finally, we, in other work, developed and reported upon a method to directly extract estimates of actual wing lift and drag force deficits using multiple observations of wing gait tests that we would be conducting anyway while evolving better oscillator patterns [12] – [14]. It is ironic that the whole reason for this EA learning exercise was to avoid doing any system identification, but then we later find a way to leverage not doing system identification into doing system identification via a back door. One of the conditions for the implicit system ID to function is that we must allocate forces asymmetrically across the wings. The simple expedient of using a crowding metric (phenotypic or genotypic) across the wing gait functions for the two wings (compare two multisets of 8 elements from one vehicle) might be able to inexpensively constrain learning in a manner that is more amenable to application of that method. Likewise, niching would provide similarly independent solution samples that could be used and would maximize the utility of the entertainingly ironic backdoor system identification just previously mentioned. Both expedients to that end are also currently under study.

DISCLAIMER

The views expressed in this paper are those of the authors and do not reflect the official views of the United States Air force nor the Department of Defense.

REFERENCES

[1] D. Doman, M. Oppenheimer, and D. Sigthorsson, "Dynamics and control of a minimally actuated biomimetic vehicle: part I – aerodynamic model", Proceedings of the AIAA Guidance, Navigation, and Control Conference. 2009.

[2] M. Oppenheimer, D. Doman, and D. Sigthorsson, "Dynamics and control of a minimally actuated biomimetic vehicle: part II – control." Proceedings of the AIAA Guidance, Navigation, and Control Conference. 2009.

[3] D. Doman, M. Oppenheimer, M. Bolender, and D. Sigthorsson, "Altitude control of a single degree of freedom flapping wing micro air vehicle." Proceedings of the AIAA Guidance, Navigation, and Control Conference. 2009.

[4] J. Goppert, J. Gallagher, I. Hwang, and E. Matson, "Model checking of a flapping-wing micro-air vehicle trajectory tracking controller subject to disturbances", The 2nd International Conference on Robot Intelligence Technology and Applications (RiTA 2013). Dec 18-20, Denver CO. 2013.

[5] J. Gallagher, D. Doman, and M. Oppenheimer, "The technology of the gaps: an evolvable hardware synthesized oscillator for the control of a flapping-wing micro air vehicle." IEEE Trans. on Evolutionary Computation, IEEE Press, vol. 16, no. 6. 2012

[6] J. Gallagher and M. Oppenheimer, "An improved evolvable oscillator and basis function set for control of an insect-scale flapping-wing micro air vehicle", Journal of Computer Science and Technology, vol. 27, no. 5, pp. 966-978. Springer. 2012.

[7] J. Gallagher and M. Oppenheimer, "An improved evolvable oscillator for all flight mode control of an insect-scale flapping wing micro air vehicle", IEEE Congress on Evolutionary Computation. IEEE Press. 2011.

[8] Gallagher, J.C., and Sam, M. (2021). Improvements to Speed and Efficacy in Non-Stationary Learning in a Flapping-Wing Air Vehicle: Constrained and Unconstrained Flight. In *Proceedings of the 2021 International Conference on Evolvable Systems in the IEEE Symposium Series on Computational Intelligence*. (ICES 2021). Orlando FL

[9] J. C. Gallagher, E. T. Matson and R. Slater, "Real-Time Learning of Wing Motion Correction in an Unconstrained Flapping-Wing Air Vehicle," *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, Italy, 2022, pp. 26-33, doi: 10.1109/IRC55401.2022.00010.

[10] M.W. Oppenheimer, I.E. Weintraub, D.O. Sigthorsson, and D.B. Doman, "Experimental Measurements of Cycle Averaged Forces for a Flapping Wing Vehicle ," in *AIAA SciTech - AIAA Guidance, Navigation, and Control Conference*, Kissimmee, 2015.

[11] M. Ol and K. Granlund, "Abstraction of Aerodynamics of Flapping-Wings: Is it Quasi-Steady? ," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Nashville, 2012.

[12] J. Gallagher, L. Humphrey, and E. Matson, "Maintaining model consistency during in-flight adaptation in a flapping-wing micro air vehicle", Robot Intelligence Technology and Applications 2, Advances in Intelligent Systems and Computing vol. 274, pp. 517-530, 2014.

[13] J. Gallagher, S. Boddhu, E. Matson, and G. Greenwood, "Improvements to evolutionary model consistency checking for a flapping-wing micro air vehicle", 2014 IEEE International Conference on Evolvable Systems (ICES), pp. 203 – 210. 2014.

[14] J. Gallagher, M. Sam, S. Boddhu, E. Matson, and G. Greenwood. "Drag Force Extension to Evolutionary Model Consistency Checking for a Flapping-Wing Micro Air Vehicle", 2016 IEEE Congress on Evolutionary Computation