An Incremental Approach to the Evolutionary Design of Autonomous Controllers for Micro-unmanned Aerial Vehicles

Fabio Ruini and Angelo Cangelosi

Abstract— **The work presented herein aims to provide a quantitative measure of the impact deriving from the adoption of an incremental approach to evolution within the context of Evolutionary Robotics. Notwithstanding the large amount of published researches relying on incremental evolution, little quantitative analysis have been performed so far to provide the answer to a basic question: is incremental evolution beneficial for evolutionary approaches to autonomous robotics? The application we use as testbed is a computer-based model of MAV (Micro-unmanned Aerial Vehicles) autonomous navigation. Either single individuals embedding a neural network controller or teams made of several MAVs are subjected to different tasks across a simulated three-dimensional world. These tasks are: (1) navigation to a target area, (2) tracking of a moving target, and (3) execution of a behaviour requiring spatial and temporal coordination among members of the same team. The results obtained are compared with those generated via direct evolution. There are no evidences of systematic advantages deriving from the adoption of an incremental evolution approach.**

I. Introduction

Focusing on the domain of evolutionary computation, for 'incremental evolution' - as concisely stated by Barlow [1] - we refer to "the process of evolving a population on a simple problem and then using the resulting evolved population as a seed to evolve a solution to a related problem of greater complexity". Incremental evolution is an alternative approach to the traditional 'direct evolution' methodology, where a population is evolved to tackle the most complex problem directly.

The inspiration for an incremental approach in artificial evolution clearly derives from biological evolution. Animal species (and this is particularly true for humans) have acquired over the time the ability to perform extremely complex tasks. These abilities have not appeared from the middle of nowhere at a certain step along the evolutionary path, rather they have been progressively built up on top of simpler behaviours used as prerequisites. If it is true that these simple behaviours have been sometimes quite evident in their manifestations (e.g., in order to learn how to run, the humans have gone through a complex series of sequential steps involving standing on their legs, walking, etc.) this has not always been the case. Sometimes, in fact, the underlying capabilities needed as prerequisites for the development of more complex behaviours remained 'silent' over the time, i.e. not expressed in form of explicit behaviours before abruptly appearing. This is what has been demonstrated by Gould introducing the concept of 'punctuated equilibrium' [2], a phenomena that later on has been easily identified among many others areas outside the biological evolution domain, such as the introduction of government policies [3], the diffusion of technological innovations [4], etc. Depending on the fitness formulae used and the specifications of the problem tackled, both continuous evolution and punctuated equilibrium dynamics (see for example the classic work by Lindgren [5]) can be seen as result of computer-simulated evolutionary processes. Incremental evolution, by definition, recreates punctuated equilibrium-like dynamics, even though the same sort of phenomena can emerge from direct evolution, especially when complex multi-parameters fitness functions are used.

In the work presented herein we propose the application of an incremental approach to evolution for an Evolutionary Robotics (ER) [6] model, aimed to the development of autonomous controllers for MAVs (Micro-unmanned Aerial Vehicles). Incremental evolution can be seen as a way to force the evolutionary algorithm used to generate a punctuated equilibrium evolutionary dynamics, thus manually directing the evolution toward the desired goal following incremental steps.

The remainder of this paper is structured as follows. In the opening section we provide a brief introduction about the usage of incremental approaches within the robotics and, more specifically, Evolutionary Robotics fields. Section II describes the functioning of the ER model employed as basis for the work presented herein and show the results obtained by non-incremental evolutionary runs of the corresponding computer simulator. Section III illustrates the characteristics of the incremental approach adopted and illustrates the results it has generated, performing a comparison with those obtained through direct evolution. The results are then discussed in details in Section IV, and the following conclusions are drawn in Section V.

A. Incremental evolution in autonomous robotics

The idea of using an incremental approach to evolution is very well known in the autonomous robotics and evolutionary computation fields. Gomez and Miikkulainen [7] describe the advantages of incremental evolution already back in 1997, mentioning among others the possibility offered by this approach for evolving behaviours otherwise not obtainable, as well as the better generalisation capabilities exhibited by controllers designed following this paradigm. But the origin of incremental evolution can be dated back even further.

F. Ruini and A. Cangelosi are with the Centre for Robotics and Neural Systems, School of Computing and Mathematics, University of Plymouth, Drake Circus, PL4 8AA Plymouth, UK {fabio.ruini, a.cangelosi}@plymouth.ac.uk

Rodney Brooks, for example, taking inspiration from his previous work on behavior-based robotics, was among the firsts to propose an incremental approach to be used within the genetic programming domain in 1992 [8]. This should not be surprising taking into account the fact that Brook's subsumption architecture itself [9] could be easily seen as a way to mimic incremental evolution, building increasingly complex behavioural modules on top of simpler ones. A tribute must then be payed to Inman Harvey's and his research group in Sussex for their studies on the SAGA (Species Adaptation GAs) framework [10]. Thanks to its work, Harvey's group - that could be considered the 'father' of the incremental approach to evolution in autonomous robotics - has provided a coherent theoretical framework for incremental evolution. Framework that has been used by a significant number of researchers all over the world as basis for their works.

During more recent times, Mouret and Doncieux [11] have attempted to make order into the field, providing a classification of the possible approaches to incremental evolution in autonomous robotics in four categories: (1) 'staged evolution', (2) 'environmental complexification', (3) 'behavioural decomposition', and (4) 'fitness shaping'. Staged evolution employs multiple fitness functions that correspond to multiple sub-tasks of increasingly difficulty: the population is initially evolved to perform the simplest task, then the fitness function is modified leading to the solution of the second task, and so on. Environmental complexification is similar to staged evolution, but the complexity of the task can be modified continuously operating on certain parameters. Behavioural decomposition (also called modular evolution) relies on the decomposition of a neural controller into separate task-based sub-controllers, each of these is evolved independently from the others. An evolutionary algorithm then combine all of these modules into a master neurocontroller. Finally, fitness shaping uses a weighted sum of multiple evaluation criteria in order to create a fitness gradient that evolution tries to follow.

As from the above classification, incremental evolution does not necessarily involve a progressive complexification of the controller architecture, although this is frequently the case especially when neural network are employed. Usages of this approach are abundant in literature. A good example consists in Stanley and Miikkulainen's work [12], where they present the NEAT (NeuroEvolution of Augmenting Topologies) method, a framework for evolving neural network topologies along with synaptic weights. The results they have obtained applying NEAT on a reinforcement learning task used as benchmark demonstrate how this approach can outperform those based on fixed neural networks topologies. In Stanley's case, the topology of the network changes over time following evolutionary dynamics. But it is also common the case in which is the experimenter who decides the topology the 'global' controller must have, manually joining various sub-modules dedicated to different functions. Togelius [13] (also reviewed in [14]) provides a further classification based on the possible ways in which different neural modules could be incrementally attached to an existing controller. He defines 'incremental evolution' as the evolution of a one layer network using multiple fitness functions, 'modularised evolution' as the evolution of multiple layers or multiple networks with a single fitness function, and 'layered evolution' as the evolution of a multi-layered network using multiple fitness functions, specific for each layer. On this basis, Tomko and Harvey [14] have pointed out that it is also of fundamental importance to consider how new units/modules are connected to the main controller during incremental evolution. Their findings highlight the detrimental effect generated by the use of random large connection weights, rather suggesting the linkage of additional neural modules using connection weights with zero values.

As reviewed by recent researches carried out by Petrovic [15], [16], many works that can be found within the abundant Evolutionary Robotics literature have employed, to different extents, an incremental approach to evolution. The topics are as differentiate as possible: from the control of unmanned aerial vehicles [1] to 6-legged robots [17], passing through artificial vision systems [18] and autonomous learning [19]. Though, most of the published works simply justify the reason for using an evolutionary approach as consequence of non-better specified 'issues' in evolving the desired behaviour through direct evolution. Rarely a quantitative analysis of the advantages coming from the adoption of an incremental approach is provided. One of the few exceptions to this trend consists in the work carried out by Walker [20], who performs an accurate comparison between the performances generated by a direct and an incremental methods for a multi-variable symbolic regression problem. Walker interestingly takes into account the full 'computational costs' of both approaches, intended as the number of evaluations of the fitness formula required by the two alternatives. The results he collected demonstrate that no significant advantages in terms of full computational costs are guaranteed by the adoption of an incremental approach. Another recent interesting study, leading to similar evidences, is the one carried out by Christensen and Dorigo [21] comparing the performances generated by two popular approaches to incremental evolution (behavioral decomposiition and environmental complexity increase) against the results obtained through several non-incremental evolutionary algorithms. According to their results none of the incremental evolutionary strategies perform any better than the non-incremental methodologies. This stream of criticisms seems to have had an effect also on a fierce supporter of the incremental approach as Harvey, that in his already mentioned work states how - according to the analysis carried out - incremental evolution seems to outperform direct evolution only under specific conditions.

In conclusion literature presents results supporting both the arguments, with a recent increase in the number of works that look at the phenomena with a skeptical eye. Anyway, the impression is that it is still extremely difficult to provide a definitive answer to the dilemma whether incremental evolution is 'better' or not than direct evolution. The study presented herein aims to give an additional contribute to the

topic, with the awareness of not being able to provide any conclusive answer due to the amount of variables that should be taken into account for a comprehensive and definitive analysis.

II. The simulation model

The computer model [22], [23] we use in this paper as testbed scenario is aimed at the automated design of autonomous controllers for MAVs and implemented according to the Evolutionary Robotics approach. The task the MAVs are subject to consists of navigation toward a certain target area - within a three-dimensional obstacle-free environment - relying on a mixture of local and global information. The assumption underlying the model consists in a high level system - aware of the location of the target area - always available and able to broadcast this information in real-time to the aircraft. The MAVs can then match this knowledge with proprioceptive information related to their current position and orientation, finding in this way the path to be followed for reaching the target area.

The virtual reference environment is a three-dimensional area with size 1,000 (X) x 1,500 (Z) x 600 (Y) graphical units (GUs)[1]. The aircraft have an approximate length of 3.5 GUs, while the target is constituted by a sphere with a 15 GUs radius.

A. The neural architectures used

Twelve different topologies of feed-forward neural networks have been tested as controllers for the MAVs (see Table I). The various architectures differ from each other because of the rotations the aircraft can perform (yaw only, yaw + pitch, yaw + pitch + roll), the absence/presence of a layer of hidden units, and the input encoding used (continuous rather than discrete).

TABLE I
NEURAL NETWORK ARCHITECTURES USED

Arch.	Pitch	Roll	Hid.	Input
1	No	No	No	D
2	No	No	No	C
3	Yes	No	No	D
4	Yes	No	No	C
5	Yes	Yes	No	D
6	Yes	Yes	No	C
7	No	No	Yes	D
8	No	No	Yes	C
9	Yes	No	Yes	D
10	Yes	No	Yes	C
11	Yes	Yes	Yes	D
12	Yes	Yes	Yes	C

Common among all the architectures is a basic structure composed of a set of input neurons[2] encoding the MAV-target distance and the horizontal angle between the two, as well as two output units. One of these output neurons is continuous and generates the yaw rotation in the time-unit; the other one is Boolean and must be activated by the MAV when it has reached the target area in order to signal the accomplishment of the mission. This unit can only be activated once per mission.

The sets of different rotation axis employed clearly have an impact on the architecture of the neural controller, concerning both the input and the output layers. Each rotation requires a specific output unit in order to be executed, as well as a set of input neurons carrying the information required by the MAV to perform the rotation in the proper way. In the basic configuration mentioned above (yaw only), the controller receives in input the horizontal angle to the target (calculated according to the MAV's heading). The possibility of pitching requires an additional input neuron feeding the network with the information related to the vertical MAV-target angle. Finally, when rolling is possible, the MAV needs to know its current roll status in order to predict the outcome of any further rotation (a 1 degree pitch when the roll angle is 0 produces a complete different results than when the MAV is upside down, i.e. 180 degrees). Figure 1 shows the topology of one of these controllers.

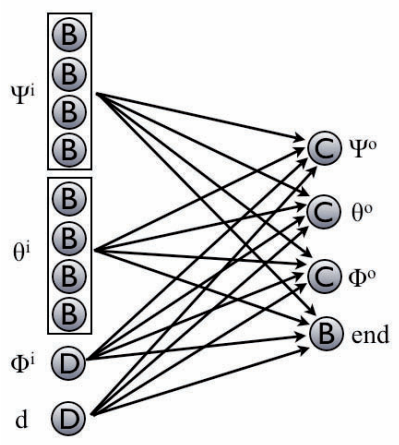

Fig. 1. Example of a NN controller. The architecture above corresponds to nr. 5.

All of the non-input and non-Boolean neurons belonging to the network get activated according to a tan-sigmoid function (slope 1.0), which output values are within $[-1.0; +1.0]$. Summation is the only aggregation function used. For topologies having a hidden layer, all the intermediate neurons (10 for every architecture) are associated with a bias value.

B. Experimental setups and evolutionary alghoritm details

Three experimental setups, labeled *A*, *B*, and *C* respectively, have been elaborated.

In scenario *A* a single MAV has to reach the target area, without exiting from the environment boundaries, and activate its Boolean output neuron once got close enough. Setup *B* is exactly the same as *A*, with the only difference

[1]Irrlicht, the 3D graphical engine software used for building the model, considers the height as the Y axis.

[2]We use the term 'set', because of the different encoding used: only one neuron is employed when the input information is encoded continuously, 4 are used instead when the same information is discretised and encoded according to the Gray Code. The only exception to this trend is the input information related to the roll status, which is discretely encoded using a single neuron.

consisting in that the target area now moves attempting to 'escape' from the approaching aircraft. When the MAV gets closer than 35 GUs to the target, the latter moves away choosing - among 26 possible equidistant final positions - the one maximising its distance from the robot. The starting population used in these two setups consists in 30 individuals (a relatively small population, for which we have taken inspiration from the micro-bial approach [24]), with connection weights and biases for their controllers randomly assigned at the beginning of the evolution within the $[-10.0; +10.0]$ range. Each controller is tested for four epochs, starting each of these from a different position and with the target area randomly dislocated. A MAV starts a test epoch with 5,000 energy units (EU) available; during each time-step it consumes 1 EU, while moving 2 GU along its heading direction. The rotations generated by the controller in the time-unit are included into the $[-3.0; +3.0]$ range. It is worth noting that, in order to simulate a more realistic flying behaviour, every time the aircraft performs a yaw a corresponding amount of roll is automatically applied[3]. As mentioned above, the Boolean output can be activated only once during the entire individuals life-span. When this neuron turns to 1, as well as when the MAV exits from the environment boundaries or runs out of energy, the current test epoch is immediately considered concluded.

Once all the members of the current generation have been evaluated, the five individuals having scored the best performance according to the fitness function in use (1) are selected for reproduction. The best one is copied to the next generation without any modifications (elitism), while the other four are subjected to a process of random mutation which affects each of their genes - with probability 0.1 - by a random value picked within the $[-0.05; +0.05]$ range. The genome is implemented via parametric encoding, with each gene constituted by a real value and representing either a connection weight or a bias. Five new individuals, with a random set of connection weights and biases, are introduced at any new generation to reduce the risk of premature convergence within the population. The process is iterated for a certain amount of generations and then repeated from the scratch for a few times (we will call each of these runs an *evolutionary seed*) in order to obtain results the less affected by randomness as possible.

Scenario *C* is slightly different than the previous two. Now teams of 4 MAVs, sharing the same controller within each squad, are used in place of single individuals. The target is a non-movable one, but it has to be reached at the same time (i.e., within a restricted amount of time-steps, since the simulator works in discrete time) by two or more MAVs in order for the test to be concluded successfully. To cope with this task, the controllers have been made more complex, with the insertion of two additional input neurons. Both of these neurons are Boolean and they get activated respectively when: (a) there is a teammate within a 25 GUs distance; (b) the target is in *damaged* status (i.e., a MAV has recently approached it and activated its *end operation* unit when situated at a proper distance). The target area starts each test epoch in a *non-damaged* status. When it is successfully approached by a MAV, its state switches to *damaged* and does not change unless 75 time-steps have passed since the last MAV approaching, in which case it returns to the previous *non-damaged* status.

From a genetic algorithm point of view, the modifications respect to the previous scenario are minor. The teams are tested for a longer amount of time (8 epochs rather than 4), each MAV starts with a larger amount of energy available (15,000 EUs instead than 5,000) and the fitness formula has required a few adjustments (see Equation(2)).

The fitness formulas used in these two scenarios are the following:

$$fitness = \alpha + \beta * 100 \tag{1}$$

$$fitness = \langle\alpha\rangle + \gamma * 50 + \beta * 100 \tag{2}$$

Equation (1) involves two parameters: α, which represents the average value - across the four epochs of testing - for the differential between the MAV-target distance at the beginning and at the end of the tests (thus representing the distance covered on the way to the target); β, indicating the overall number of tests succeeded. α is set to 0 in case the MAV has concluded the test because exited from the environment boundaries or ran out of energy, and assumes non-zero values when the *end test* neuron has been used.

Equation (2) adds a new parameter to (1), γ, which represent the amount of tests concluded half-successfully, i.e. with at least one MAV managing to properly approach the target, but not two or more at the same time. Furthermore α has been modified to $\langle\alpha\rangle$, thus indicating that it now represents the average distance covered by all the four MAVs during the eight test epochs.

C. Basic results for A, B, and C setups

Evolutions in scenarios *A* and *B* have been run for different amounts of generations according to the complexity (intended as the number of connection weights) of the neural networks used: 5,000 generations for architectures 1, 2, 7, and 8; 10,000 generations for 3, 4, 9, and 10; 20,000 generations for 5, 6, 11, and 20. Given M_s the speed of the MAV, five different evolutionary runs have been performed for each architecture in scenario *B*, with the target moving at speeds T_s equal to $\frac{M_s}{5}$, $\frac{M_s}{4}$, $\frac{M_s}{3}$, $\frac{M_s}{2}$ and M_s respectively.

The results for scenario *A* are reported in Table II, while Tables III and IV show the outcome of the simulations carried out within the *B* scenario for T_s equal to $\frac{M_s}{2}$ and $\frac{M_s}{3}$. The reasons for considering this subset of the *B* simulations only are discussed in details in [23]. Essentially, with targets moving at M_s the MAVs can not ever perform the task successfully, since the target has an advantage in terms of freedom of movements compared to the aircraft. At the same time, speeds as $\frac{M_s}{3}$, $\frac{M_s}{4}$, and $\frac{M_s}{5}$ generate quite similar results, with significantly higher success rates than those

[3]This only applies to architectures where roll is used, namely 5, 6, 11, and 12.

obtained by the aircraft when tackling targets travelling at $\frac{M_s}{2}$. $\frac{M_s}{3}$ could then be used as representative for all the values of M_s smllaer than $\frac{M_s}{2}$. The two tables show the average fitness values scored by the entire population at the end of the evolution[4], as well as the maximum (i.e., the best individual's fitness). The average and maximum success rates (intended as percentage of tests concluded successfully) are also reported. Since at any given generation 5 new random individuals are introduced (corresponding to 16.66% of the entire population), the maximum average success rate which is legitimate to expect can not be higher than 83.34%.

TABLE II
RESULTS FOR THE A SETUP

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	978.4878	1420.9	0.7976	1
2	989.2763	1426.5	0.8171	1
3	904.1651	1413.3	0.6963	0.9992
4	858.6135	1331.4	0.5677	0.8916
5	749.7256	1309.5	0.4891	0.9419
6	602.7285	1050.9	0.1857	0.4693
7	1005.3	1428.6	0.8235	1
8	997.0632	1430.6	0.8236	1
9	881.6735	1399.3	0.6665	0.999
10	934.6812	1413.7	0.7247	0.9986
11	688.2933	1272	0.4209	0.8556
12	623.8767	1111.8	0.3104	0.6556

TABLE III
RESULTS FOR THE B NON-INCREMENTAL SETUP ($T_s = \frac{M_s}{2}$)

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	976.2088	1421	0.8001	1
2	976.9735	1420	0.7823	1
3	855.3585	1386.6	0.5982	0.9979
4	744.2322	1252	0.4076	0.8197
5	645.5993	1158.3	0.2849	0.5969
6	551.6119	996.7838	0.1597	0.396
7	983.8354	1421.7	0.7914	1
8	933.4124	1430.4	0.814	1
9	756.3261	1323.7	0.5035	0.9642
10	851.7244	1336.5	0.5376	0.8724
11	643.1058	1126.9	0.1807	0.5829
12	700.0007	1162.4	0.3384	0.6609

As expected, the complexity of the controllers used affects the performance of the MAVs. For the simplest 2D scenario (controllers 1, 2, 7, and 8) all the topologies produces good results, leading to MAVs able to successfully perform the task 100% of times. The 3D scenario where only yaw and pitch are allowed (controllers 3, 4, 9, and 10) also prove to be not particularly challenging for the evolutionary process. The average success rate for the entire population slightly decreases, but the best controllers still can, among all the cases, perform at least 80% of tasks with success. Things get more complicated when using controllers 5, 6, 11, and 12 (i.e., adding roll among the possible rotations available to the aircraft) and the results are contrasting also. In Scenario *A* the discretisation of the input information seems to have a clearly positive impact on the performances of the controllers. For Scenario *B* we find instead evidence of both positive and negative impacts. Curiously, some of the controllers (e.g., nr. 10) have evolved with more accurate behaviours for Scenario *B* ($T_s = \frac{M_s}{3}$) than for Scenario *A*.

[4]This value is calculated as the average for the last 50 generations, over 10 evolutionary seeds.

TABLE IV
RESULTS FOR THE B NON-INCREMENTAL SETUP ($T_s = \frac{M_s}{3}$)

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	979.821	1422.8	0.7948	1
2	991.5413	1426.2	0.8177	1
3	889.1834	1407.7	0.6918	1
4	787.9387	1258.3	0.4385	0.7998
5	673.2221	1212.7	0.3354	0.7137
6	596.2457	1063.5	0.2001	0.5255
7	995.336	1428.3	0.8081	1
8	994.9137	1431.4	0.8153	1
9	806.9692	1354.3	0.5497	0.9785
10	951.2047	1417	0.7276	1
11	577.6951	1090.6	0.1955	0.4829
12	744.1598	1241.6	0.4435	0.8123

The evolution in Scenario *C*, due to the more complex behaviour to evolve, has been run for: 10,000 generations for architectures 1, 2, 7, and 8; 20,000 generations for architectures 3, 4, 9, and 10; 40,000 generations for architectures 5, 6, 11, and 12. The results from this setup are presented in Table V. In this case the table contains an extra column indicating the percentage of tests concluded half-successfully, i.e. with at least one MAV having properly reached the target, but not two or more MAVs having done the same in a coordinated fashion.

TABLE V
RESULTS FOR THE C NON-INCREMENTAL SETUP

Arch.	Av. fit.	Max fit.	Av. s. rate	Max s. rate	Half s. rate
1	1106.8	1701	0.4672	0.8814	0.3669
2	1011.4	1505.3	0.3164	0.5818	0.4316
3	709.269	1104.2	0.0648	0.2834	0.5914
4	683.148	948.381	0.0037	0.0636	0.7636
5	645.664	1015.2	0.0533	0.239	0.5794
6	582.288	822.920	0.002	0.0386	0.5128
7	1008.4	1628.3	0.4266	0.8425	0.2926
8	1228.2	1834.6	0.5391	0.9184	0.1754
9	743.587	1131.5	0.0747	0.2844	0.6201
10	752.741	1153.4	0.0662	0.2778	0.647
11	562.598	917.089	0.0365	0.1855	0.4713
12	764.835	1123.9	0.058	0.2316	0.677

Not surprisingly, these results highlight worse performance for the controllers than those obtained within Scenarios *A* and *B*. The new task is indeed more complicate, since on top of the basic navigation behaviour the aircraft are now also required to coordinate among themselves. The difficulty is testified by the data relative to the maximum success rate obtained by the controllers in *C*, that score good performances only for architectures 1, 2, 7, and 8, i.e. those where the limited rotations available to the MAVs make the task relatively easier. Nonetheless, from a behavioural

perspective it is interesting to observe the strategies evolved by the successful controllers within this setup (see Figure 2). Rather than looking for each other, assemble and then fly together toward the target, the team members navigate independently to the target area. Once the first arrives there it keeps circling around the target waiting for a teammate to arrive. When at least two MAVs are in proximity of the target, their behaviour suddenly changes: they aim to the target and activate their dedicated output Boolean unit.

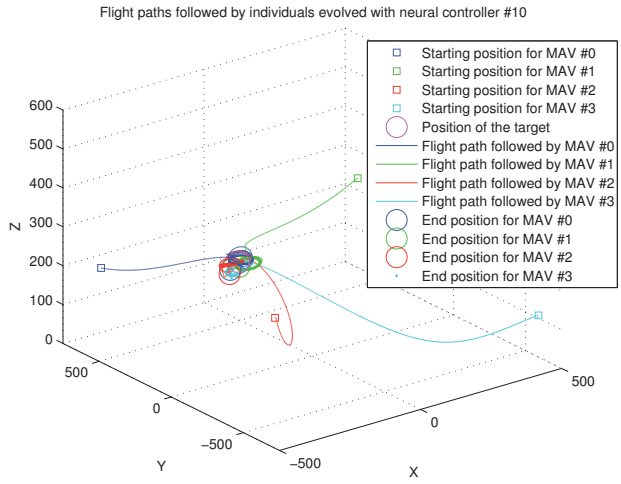

Fig. 2. Flight paths followed by the members of a MAVs team using controller nr. 10 within the C setup. It is possible to see how the MAVs converge to the target and then keep flying around it while waiting for the proper conditions to appear.

III. The incremental approach

The incremental approach adopted in this work is particularly straightforward. Evolving from Scenario *A* to *B* the connection weights and biases of the individuals belonging to the last generation of the best population evolved in *A* are loaded from the memory and used as starting point for the new evolutionary process. Things are slightly more complicated moving from Scenario *A* to *C*, because of the different network topologies used in the latter. In this case, the connection weights coming from the two extra input neurons are added to the A controllers and their values are set to 0 at the beginning of the second stage of the evolutionary process (as suggested in [14]).

A. Results of evolution from scenario A to B

Starting from populations of individuals already evolved within the *A* scenario, the results demonstrate how 5,000 generations of further evolution are enough to allow these individuals to generalise their target reaching abilities to moving targets as well. The only situations for which this does not happen are when controllers 5 and 6 are used (furthermore, for these two architecture the performance of the controller dramatically decreases). Also controller 10, for a target moving at half the speed of the MAVs, generates worst results when incrementally evolved than when direct evolution is employed. The absolute results are summarised in Tables VI and VII. Tables IX and X show the comparison between the incremental and the non-incremental results.

TABLE VI
Results for the A-to-B incremental setup ($T_s = \frac{M_s}{2}$)

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	955.6403	1428.1	0.82	1
2	1002.6	1434.1	0.8133	1
3	864.5446	1378.6	0.5477	0.985
4	783.5742	1321	0.5264	0.9667
5	602.3626	1110.4	0.1358	0.4975
6	579.5491	970.2302	0.0116	0.1485
7	1007.4	1435.3	0.8228	1
8	999.0517	1434	0.8164	1
9	877.6665	1393.3	0.6306	0.996
10	688.5148	1149.1	0.2359	0.5317
11	554.3824	1167.4	0.3026	0.8108
12	772.3867	1298.9	0.4516	0.9472

TABLE VII
Results for the A-to-B incremental setup ($T_s = \frac{M_s}{3}$)

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	952.8566	1408.7	0.7546	0.9965
2	994.8059	1432.1	0.8183	1
3	928.2931	1413.3	0.6271	0.9985
4	851.2675	1359.3	0.5777	0.957
5	624.5276	1143.5	0.1971	0.5888
6	597.6601	964.9632	0.0053	0.0867
7	978.4332	1424.5	0.7931	1
8	995.912	1432.1	0.8222	1
9	887.8627	1400.9	0.6454	1
10	741.8259	1321.9	0.4466	0.9405
11	611.1595	1237.6	0.3862	0.9103
12	796.4933	1333.7	0.5018	0.9642

B. Results of evolution from scenario A to C

In this case the incremental evolutionary process has lasted for 10,000 generations, twice the duration of the one used for the *B* scenario, due to the more sophisticated neural architectures used. The absolute results are summarised in Table VIII, while Table XI show the comparison between the incremental and the non-incremental results. Just at a first glance it is possible to see how the controllers evolved do not score impressive results.

IV. Analysis of the incremental results

For what concerns incremental evolution from A to B, interestingly, the critical variable for the success of the incremental approach seems to be the complexity of the neural architecture used. For simple networks, as controllers 1-6 (feed-forward NNs without hidden layers) are, the effects seems to be limited. More complex architectures benefited much more instead from the incremental process. This phenomena is evident comparing the results scored by architectures 5 and 6 against controllers 11 and 12. The average success rate of controller 5 dropped by 52.33% and 41.23% for $T_s = \frac{M_s}{2}$ and $T_s = \frac{M_s}{3}$ respectively,

TABLE VIII

RESULTS FOR THE A-TO-C INCREMENTAL SETUP

Arch.	Av. fit,	Max fit.	Av. succ.	Max succ.	Half succ.
1	1139.8	1725.5	0.4893	0.9001	0.3401
2	1347.2	1904.7	0.6257	0.9903	0.2062
3	692.937	1046.9	0.0477	0.2299	0.6816
4	512/0017	755.597	0.0023	0.0428	0.6639
5	475.425	742.073	0.0058	0.0711	0.4788
6	545.109	929.720	0.0289	0.1873	0.3052
7	1196.9	1776.6	0.5288	0.9289	0.3058
8	1386.5	1921.5	0.6543	0.9962	0.1782
9	699.835	1076.4	0.0731	0.2693	0.606
10	710.436	1058.3	0.0409	0.1896	0.6722
11	400.554	654.008	0.0084	0.0646	0.3519
12	568.793	910.571	0.0309	0.1459	0.5339

while architecture 6 scored -92.74% and -97.35%. Viceversa, controller 11's performance increased by 67.46% and 97.54%; architecture 12 improved as well, scoring +33.45% and +13.15%. The simplest architectures - as 1, and 2, but also 7, and 8 - do not show any significant difference in the results obtained following the two alternative approaches, presumably because how to perform the task (which, for these controllers, is essentially 2D navigation) is quite easy to be learnt and direct evolution already found nearly-optimal solutions. Things are more interesting and variegated for the architectures of intermediate complexity, such as 3, 4, and 9[5]. Architectures 4 and 9 improved their performances among all the parameters measured, for both $T_s = \frac{M_s}{2}$ and $T_s = \frac{M_s}{3}$. For architecture 3 the fitness values scored (both average and maximum) are pretty similar among direct and incremental evolution, but the average and maximum success rates decreased.

TABLE IX

INCREMENTAL VS NON-INCREMENTAL EVOLUTION FOR THE B SETUP ($T_s = \frac{M_s}{2}$)

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	-2.11%	+0.50%	+1.65%	0.00%
2	+2.62%	+0.99%	+4.82%	0.00%
3	+1.07%	-0.58%	-8.44%	-1.29%
4	+5.29%	+5.51%	+29.15%	+17.93%
5	-6.70%	-4.14%	-52.33%	-16.65%
6	+5.06%	-2.66%	-92.74%	-62.50%
7	+2.40%	+0.96%	+3.97%	0.00%
8	+7.03%	+0.25%	+0.29%	0.00%
9	+16.04%	+5.26%	+25.24%	+3.30%
10	-19.16%	-14.02%	-56.12%	-39.05%
11	-13.80%	+3.59%	+67.46%	+39.10%
12	+10.34%	+11.74%	+33.45%	+43.32%

Opposite results have been obtained by the incremental evolution from *A* to *C*, i.e. the further evolution of an architecture specialised in basic navigation to perform a cooperative task. In this case, the only controllers that have gained advantage from the second evolutionary process have been the simplest ones: 1, 2, 7, and 8. All the others architectures (with the only exception of controller 6, for which direct evolution did not succeed) have seen their performances dropping consistently, both in terms of average and maximum fitness, as for what concerns the success rate.

[5]For the purposes of this analysis, we do not take into account controller 10, for which incremental evolution has not been able to generate a proper behaviour.

TABLE X

INCREMENTAL VS NON-INCREMENTAL EVOLUTION FOR THE B SETUP ($T_s = \frac{M_s}{3}$)

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	-2.75%	-0.99%	-5.06%	-0.35%
2	+0.33%	+0.41%	+0.07%	0.00%
3	+4.40%	+0.40%	-9.35%	-0.15%
4	+8.04%	+8.03%	+31.74%	+19.65%
5	-7.23%	-5.71%	-41.23%	-17.50%
6	+0.24%	-9.27%	-97.35%	-83.50%
7	-1.70%	-0.27%	-1.86%	0.00%
8	+0.10%	+0.05%	+0.85%	0.00%
9	+10.02%	+3.44%	+17.41%	+2.20%
10	-22.01%	-6.71%	-38.62%	-5.95%
11	+5.79%	+13.48%	+97.54%	+88.51%
12	+7.03%	+7.42%	+13.15%	+18.70%

TABLE XI

INCREMENTAL VS NON-INCREMENTAL EVOLUTION FOR THE C SETUP

Arch.	Av. fit.	Max fit.	Av. s. rate	Max s. rate	Half s. rate
1	+2.98%	+1.44%	+4.73%	2.12%	-7.30%
2	+33.20%	+26.53%	+97.76%	+70.21%	-52.22%
3	-2.30%	-5.19%	-26.39%	-18.88%	+15.25%
4	-25.05%	-20.33%	-37.84%	-32.70%	-13.06%
5	-26.37%	-26.90%	-89.12%	-70.25%	-17.36%
6	-6.39%	+12.98%	+1345%	+385.23%	-40.48%
7	+18.69%	+9.11%	+23.96%	+10.26%	+4.51%
8	+12.89%	+4.74%	+21.37%	+8.47%	+1.60%
9	-5.88%	-4.87%	-2.14%	-5.31%	-2.27%
10	-5.62%	-8.25%	-38.22%	-31.75%	+3.89%
11	-28.80%	-28.69%	-76.99%	-65.18%	-25.33%
12	-25.63%	-18.98%	-46.72%	-37.00%	-21.14%

V. CONCLUSION AND FUTURE WORKS

As we have discussed in Section I, it is difficult to draw from a single experiment definitive conclusions about the validity of a complex and widely applicable approach such as incremental evolution. Not only the particular task analysed, but also countless variables seem to have an impact on the profitable applicability of an incremental approach, as demonstrated by the study presented herein. The challenge consists in identifying these variables and provide a theoretical framework - a set of guidelines - that researchers willing to experiment in incremental evolution could rely on in the future in order to fully benefit from this approach.

Based on the results described in this paper, we could suggest two aspects deserving further investigations. First, it seems to be necessary that the evolutionary algorithm might be free to explore a large space of solutions. Incremental evolution performed from Scenario *A* to *B* has been particularly beneficial for the more complex architectures (i.e., those characterised by a larger amount of connection weights),

while generating limited results for the others. Second, the incremental process has to go through a series of steps very close to each other. This has been demonstrated by the fact that direct evolution, in the testbed scenario discussed herein, has clearly outperformed the incremental approach for Scenario *C*.

The findings resulting from this work will be taken into account for the next steps of the project, involving the testing of the evolutionary controllers developed in computer simulations on physical robotics platforms. This work will be carried out during the next few months in collaboration with the EPFL's Laboratory of Intelligent Systems (LIS) in Lausanne, Switzerland. The platform that will be used for these experiments is the senseFly's *swinglet*[6] [25], a robotic aircraft which has already been tested for various tasks and that seem particularly well fit for successfully hosting such a kind of autonomous controllers.

VI. Acknowledgments

Effort sponsored by the Air Force Office of Scientific Research, Air Force Office Material Command, USAF under grant number FA8655-07-1-3075.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

The U.S. Government is authorized to reproduce and distribute reprints for government purpose notwithstanding any copyright notation thereon.

References

[1] G. Barlow, C. Oh, and E. Grant, "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming," *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*, 2004.

[2] S. Gould, *The Structure of Evolutionary Theory*. Belknap Press of Harvard University Press, 2002.

[3] F. Baumgartner and B. Jones, *Agendas and Instability in American Politics*. The University of Chicago Press, 1993.

[4] C. Loch and B. Huberman, "A punctuated-equilibrium model of technology diffusion," *Management Science*, vol. 45, no. 2, pp. 160–177, 1999.

[5] K. Lindgren, "Evolutionary phenomena in simple dynamics," *Artificial Life II*, pp. 295–312, 1991.

[6] S. Nolfi and D. Floreano, *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press, 2000.

[7] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, no. 5, pp. 317–342, 1997.

[8] R. Brooks, "Artificial life and real robots," *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 3–10, 1992.

[9] ——, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[10] I. Harvey, "Artificial evolution: A continuing saga," *Evolutionary Robotics. From Intelligent Robotics to Artificial Life*, vol. 2217, pp. 94–109, 2001.

[11] J.-B. Mouret, S. Doncieux, and J.-A. Meyer, "Incremental evolution of target-following neuro-controllers for flapping-wing animats," *From Animals to Animats 9. Proceedings of SAB 2006, the 9th International Conference on Simulation of Adaptive Behavior*, pp. 606–618, 2006.

[6] http://www.sensefly.com/products/swinglet

[12] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[13] J. Togelius, "Evolution of a subsumption architecture neurocontroller," *Journal of Intelligent and Fuzzy Systems*, vol. 15, no. 1, pp. 15–20, 2004.

[14] N. Tomko and I. Harvey, "Do not disturb: Recommendations for incremental evolution," *Proceedings of ALIFE XII, the 12th International Conference on the Synthesis and Simulation of Living Systems*, 2010.

[15] P. Petrovic, "Overview of incremental approaches to evolutionary robotics," *Proceedings of the 1999 Norwegian Conference on Computer Science*, pp. 151–162, 1999.

[16] ——, "A step towards incremental on-board evolutionary robotics," *Proceedings of SCAI'01, the Seventh Scandinavian Conference on Artificial Intelligence*, pp. 3–12, 2001.

[17] D. Filliat, J. Kodjabachian, and J.-A. Meyer, "Incremental evolution of neural controllers for navigation in a 6-legged robot," *Proceedings of the Fourth International Symposium on Artificial Life and Robotics*, 1999.

[18] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: Artificial evolution, real vision," *From Animals to Animats 3, Proceedings of SAB 1994, the 3rd International Conference on Simulation of Adaptive Behaviour*, pp. 392–401, 1994.

[19] E. Tuci, M. Quinn, and I. Harvey, "An evolutionary ecological approach to the study of learning behavior using a robot-based model," *Adaptive Behavior*, vol. 10, no. 3-4, pp. 201–221, 2002.

[20] M. Walker, "Comparing the performance of incremental evolution to direct evolution," *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, pp. 119–124, 2004.

[21] A. Christensen and M. Dorigo, "Incremental evolution of robot controllers for a highly integrated task," *From Animals to Animats 9*, pp. 473–484, 2006.

[22] F. Ruini and A. Cangelosi, "An evolutionary robotics 3d model for autonomous mavs navigation, target tracking and group coordination," *Proceedings of IJCNN 2010, International Joint Conference on Neural Networks*, 2010.

[23] ——, "Extending the evolutionary robotics approach to flying machines: An application to mav teams," *Neural Networks*, no. 22, pp. 812–821, 2009.

[24] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi, "Evolutionary robotics: the sussex approach," *Robotics and Autonomous Systems*, vol. 20, pp. 205–224, 1996.

[25] S. Leven, J.-C. Zufferey, and D. Floreano, "A minimalist control strategy for small uavs," *Proceedings of IROS 2009, the IEEE/RSJ International Conference on Intelligent RObots and Systems*, pp. 2873–2878, 2009.