A Hybrid Neural Network-Genetic Programming Intelligent Control Approach

Francesco Marchetti[0000−0003−4552−0467] and Edmondo Minisci[0000−0001−9951−8528]

Intelligent Computational Engineering Laboratory, University of Strathclyde, Glasgow, UK
{francesco.marchetti,edmondo.minisci}@strath.ac.uk

Abstract. The proposed work aims to introduce a novel approach to Intelligent Control (IC), based on the combined use of Genetic Programming (GP) and feedforward Neural Network (NN). Both techniques have been successfully used in the literature for regression and control applications, but, while a NN creates a black box model, GP allows for a greater interpretability of the created model, which is a key feature in control applications. The main idea behind the hybrid approach proposed in this paper is to combine the speed and flexibility of a NN with the interpretability of GP. Moreover, to improve the robustness of the GP control law against unforeseen environmental changes, a new selection and crossover mechanisms, called Inclusive Tournament and Inclusive Crossover, are also introduced. The proposed IC approach is tested on the guidance control of a space transportation system and results, showing the potentialities for real applications, are shown and discussed.

Keywords: Intelligent Control · Genetic Programming · Neural Networks · Optimal Control · Space Transportation System

1 Introduction

Many different Artificial Intelligence (AI) techniques have been used in the past decades for various IC applications [18], where the definition of IC can be summarized as follows: *a controller can be defined intelligent if it can deal autonomously with unforeseen changes in the environment, in the control system or in the goals, by relaying on techniques pertaining to the fields of Artificial Intelligence, Operations Research and Automatic Control* (Saridis [16] and Antsaklis [1]). Mainly three different classes of AI techniques have been used for IC both alone and hybridized: Fuzzy Logic (FL), Evolutionary Computing (EC) and Machine Learning (ML). Among these different techniques, NNs are certainly the most common, mainly due to their flexibility and their ability in being integrated in control systems of any kind, e.g. as was done in [7].

GP can be also used for control applications (Marchetti et. al. [12], Chiang [2]) and as suggested by Koza et. al. [9], it is particularly interesting for its ability to produce interpretable control laws for nonlinear systems. In fact, GP possess two

key advantages in comparison to NNs: 1) it can create a regression model from scratches by interacting with the environment, so without the need of providing a huge amount of training data; and 2) it produces a human-readable mathematical equation, which can be easily interpreted by the designer. Such interpretability is of particular importance for control applications, where in order to assess the reliability and behaviour of a control system the control equation must be known. Considering that an intelligent controller must possess the ability to learn and adapt online, hence the computational load of this adaptation or learning process must be very low, the main aim of this work is to propose an IC approach that combines GP and NN, which has the interpretability of GP without its excessive computational cost for online learning and adaptation.

In this respect, a novel hybrid approach to IC is introduced, where a NN is used to optimize online the GP control law found during the offline training process. Moreover, to increase the robustness of the proposed control method, two new features have been introduced in the considered GP algorithm: the Inclusive Tournament and the Inclusive Crossover. It is shown that these alternative heuristics allow for a greater robustness of the GP control law when varying environmental conditions and uncertainties are considered.

To the authors best knowledge, the control approach introduced in this work is not present in the literature. In fact, the classical approach to hybridize GP and NN is in a Neuroevolutionary manner where GP or another evolutionary algorithm to optimize the topology and weights of a NN which then is used for control purposes, as in [6, 15] or inside a Reinforcement Learning (RL) control framework as in [5, 8].

The paper is structured as follows: Section 2 introduces the concepts behind the NN Optimization of the GP control law; Section 3 introduces the new tournament and crossover mechanism employed in this work; in Section 4 the chosen test case is presented, and the obtained results are shown and discussed in Section 5; the final conclusions and future work directions are then presented in Section 6.

2 Neural Optimization of the Genetic Programming Control Law

The core of this work consists in the use of a feedforward NN to optimize online a control law obtained offline using GP. The idea of this hybrid approach comes from the need to keep the interpretability of a control law produced using GP, while having the advantages of a NN, namely the ability to perform learning and adaptation online with a much lower computational cost.

As schematised in Fig. 1, once the GP control law is obtained offline, the NN is used to optimize online the key coefficients of such control law, that are:

- The *weights* inserted as multiplying factors of the input variables of the GP control law. All the weights are independent and initialized to 1. The weights are inserted after the GP control law is created, as depicted in Fig. 2.

– The *real valued coefficients* inserted in the GP control law during its creation. Also for these values as for the weights, they are all independent. This means that if the same coefficient is used multiple times in the GP law, these are considered as different coefficients by the optimization process.

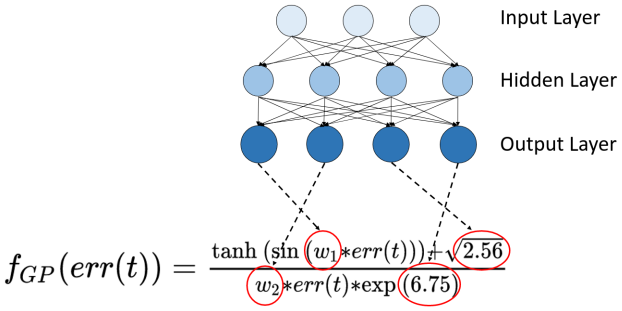


Fig. 1: Schematic of the genetic programming control law update

$$f_{GP}(err(t)) = \frac{\tanh(\sin(err(t)) + \sqrt{2.56}}{err(t) * \exp(6.75)}$$

$$\downarrow$$

$$f_{GP}(err(t)) = \frac{\tanh(\sin(w_1 * err(t)) + \sqrt{2.56}}{w_2 * err(t) * \exp(6.75)}$$

Fig. 2: How the GP control is modified to insert the multiplying factors. The red rectangles highlights where the weights are inserted

The procedure to go from the offline creation of the GP control law to its online adaptation is described by the following three steps schematized also in Fig. 3:

1. The GP control law is obtained offline by simulating a control task where a disturbance is considered. The GP control law is created to control the plant against that particular disturbance scenario: noise is introduced in the form of uncertainties in the physical models and perturbations of the applied disturbances. The main goal of this phase is to find a robust GP control law, in order to perform well also on unseen disturbance scenarios.

2. Define a set of disturbance cases uniformly covering the whole domain of the possible disturbance scenarios (Fig. 4a), referred as "disturbance domain" from this point onward, and perform an optimization of the obtained GP control law on each of the disturbance points. The optimization is considered successful if the performed trajectory is the same or within a small range defined by the user from the reference trajectory. The results obtained from the successful optimizations are stored to produce a training dataset for the NN. Such optimization of the GP control law performed on the training set is done using classical optimization techniques.
3. The NN is trained on the training dataset produced at the previous step and the created model is then tested on a predefined set of disturbance scenarios (Fig. 4b) considering also the uncertainties on the physical models.

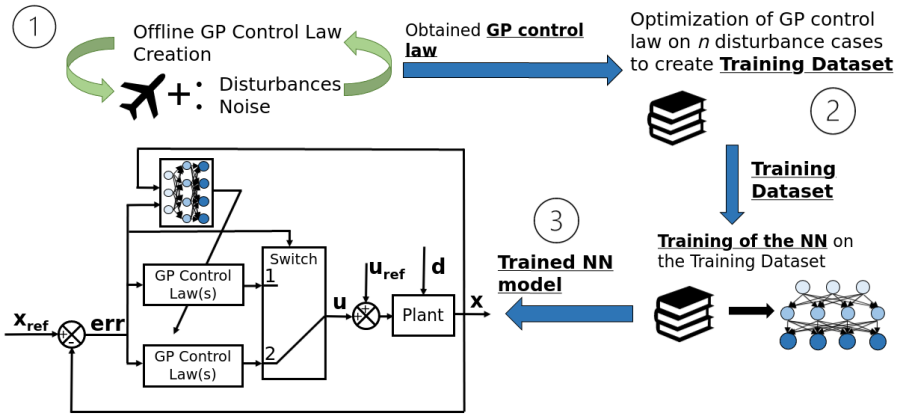
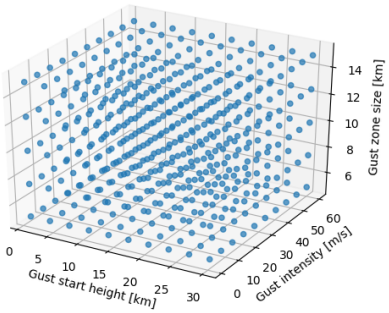


Fig. 3: Schematic of the process to obtain the proposed Hybrid Neural Network-Genetic Programming Intelligent Control Approach. The proposed control scheme is depicted in the lower left part.

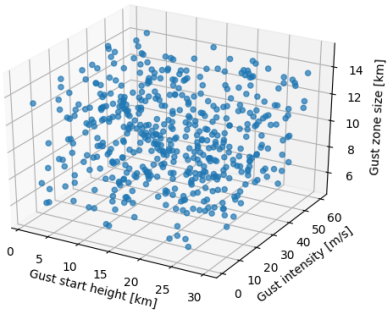
According to the taxonomy presented in [18], the proposed control approach can be classified as G0, E2, C1, since the goal is predefined by the user and it is followed by minimizing the tracking error (G0); the environment is defined but subject to time varying disturbances, modelled by the introduction of environmental disturbances and uncertainties (E2); the GP control law parameters are updated online by the NN (C1).

3 Inclusive Genetic Programming

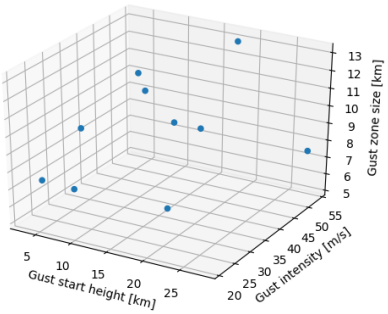
When using GP in a control environment, it is important to assess the robustness of the produced control law. In fact, it is desirable to obtain a control law,



(a) Training Scenarios - Disturbance scenarios used to create the training dataset for the NN



(b) Test Scenarios - Disturbance scenarios used to test the proposed control approach



(c) Disturbance scenarios used to compare the control laws for the SPG and IGP comparison

Fig. 4: Different sets of disturbance scenarios considered in this work

which is capable to control the desired plant also in the presence of uncertainties and unforeseen events. Since an intelligent controller is expected to work on nonlinear systems, to increase the robustness of the GP control law and describe their behaviour in a more accurate way, it is important for it to capture the nonlinear behaviour of the plant. In order to do so, the GP must maintain a certain degree of diversity in the population to preserve also complex structures able to capture nonlinearties, which would be discarded otherwise due to their excessive complexity.

To maintain the population diversity and improve the robustness of the GP control law, a new tournament and crossover mechanisms, which can be seen as approaches to create and exploit different niches [17], are introduced: the Inclusive Tournament (IncT) and the Inclusive Crossover (IncC).

The main concept behind the IncC and IncT is the following: the individuals in the population are divided into n categories (niches) according to their genotypic diversity, i.e. the number of nodes in an individual or individual length. The categories are created in an evenly distributed manner (linearly divided) between the maximum and minimum length of the individuals in the population, then the individuals are assigned to the respective category according to their length. The same number of categories is kept during the evolutionary process, but their size (the interval of individuals lengths that they cover) and the amount of individuals inside them change at every generation. Once the categories are created, both the crossover and tournament selection are performed considering individuals from different categories in order to maintain the population diversity. Finally, before starting the evolutionary process, a search for the population with the highest entropy [14] is performed, in order to start with a diversified initial population.

3.1 Inclusive Tournament

The IncT consists in performing a Double Tournament [10] on each category of the considered population as in Algorithm 1. Moreover, the selection mechanism applied to the considered individuals is modified in order to favour those individuals that respect the applied constraints (called Modified Double Tournament in Algorithm 1). The employed evolutionary strategy is based on the $\mu + \lambda$ algorithm.

Algorithm 1 Inclusive Tournament

```
Divide the population into n categories based on the length of the individuals
while Number of selected individuals < μ do
    for i in number of categories do
        if Number of selected individuals from i-th category < total number of individuals in i-th category then
            Select one individual in i-th category with Modified Double Tournament selection
        end if
    end for
end while
```

3.2 Inclusive Crossover

Algorithm 2 describes the mechanism behind the IncC. A one point crossover is applied between two individuals which are selected from two different categories. The list of exploitable categories is continuously updated in order to avoid selecting always from the same categories. About the two individuals chosen, one is the best of the considered category, in order to pass the best performing genes

to the future generation and the other is selected randomly in order to maintain a certain degree of diversity in the population. Moreover, a mechanism to avoid breeding between the same or very similar individuals is used (lines 7-14 in Algorithm 2). Here n_l is a preset constant defining the maximum number of loop iterations, needed to avoid possible infinite loops.

Algorithm 2 Inclusive Crossover

1: **if** *List of exploitable categories* is empty **then**
2: *List of exploitable categories* ← list of all filled categories
3: **end if**
4: Select randomly two different categories from *List of exploitable categories*
5: Remove chosen categories from *List of exploitable categories*
6: Select the **best** individual from the first category and select a **random** individual from the second category
7: n = 0
8: **while** The selected individuals have the same fitness and n $< n_l$ **do**
9: Repeat lines 4 to 6
10: **if** *List of exploitable categories* is empty **then**
11: *List of exploitable categories* ← list of all filled categories
12: **end if**
13: n = n+1
14: **end while**
15: Apply crossover to the chosen individuals

3.3 Robustness Analysis

To quantify the improvements in the robustness of the obtained GP control law made possible by the IncT and IncC, a comparison between the Standard Genetic Programming (SGP) and the Inclusive Genetic Programming (IGP) was made on the creation of a control law for the test case introduced in Section 4, considering the uncertainties in the models and a perturbation of the design disturbance scenario. Such perturbation was introduced as a random variation of $\pm 10\%$ of the design gust v_g and gust range Δ_g at every generation of the evolutionary process. Such variation is not cumulative across generations and its magnitude is different at every generation. Regarding the disturbance scenario, more is explained in Section 4.1.

The creation of the GP control law was performed on 10 different disturbance scenarios (Fig. 4c), which were spread uniformly over the disturbance domain. Each line in Table 1 is a control law created on a different disturbance scenario. Both GP algorithms were set to create two control laws at the same time, since two control parameters were present in the considered test case (Individual 1 and 2 in Table 1). The SGP algorithm employs a Double Tournament selection and a single point crossover. The GP algorithms were set as follows:

- *Primitives*: $+, ++$ (ternary addition), $-, *, \tanh, \sqrt{.}, \log, \exp, \sin, \cos$. All the primitives were modified in order to avoid numerical errors.
- *Fitness Functions*: two fitness functions were simultaneously considered
  1. $\min fitness_1 = RMSE(objectives)$ where:

$$objectives = \Big(\int \mid \hat{err}_v(t) \mid dt, \int \mid \hat{err}_\chi(t) \mid dt, \int \mid \hat{err}_\gamma(t) \mid dt, \int \mid \hat{err}_h(t) \mid dt, orbital\ requirements \Big)$$

     where the considered errors are scaled with the maximum values of the states, and

$$orbital\ requirements = \left| \frac{h_{reffinal} - h_{final}}{h_{max}} \right| + \left| \frac{v_{abs} - v_{orbit}}{v_{max}} \right| + \left| \frac{\chi_{abs} - \chi_{orbit}}{\chi_{max}} \right| + \left| \frac{\gamma_{reffinal} - \gamma_{final}}{\gamma_{max}} \right|$$

  2. $\min fitness_2 = ||constraints\ violation||_2$
- *Termination Criteria*: $generation\ number = 150$
- *Double Tournament Parsimony size*: 1.6 for both algorithms
- *Double Tournament Fitness size*: 2 for both algorithms
- *Population Size*: 300 individuals for both algorithms
- *Crossover Rate*: 0.2 for IGP, 0.7 for SGP
- *Mutation Rate*: 0.7 for IGP, 0.2 for SGP
- *Elitism Rate*: 0.1 for both algorithms

For the IGP, the mutation rate was set at 0.7 at the beginning of the evolutionary process in order to explore the search space. Then, when feasible individuals were found ($fitness_2 = 0$), the mutation rate was decreased by 0.01 and the crossover rate was increased by the same quantity at each generation, until the crossover rate reached the value of 0.65. Moreover, to avoid bloating, both the IGP and SGP make use of the bloat control operators implemented in the DEAP library.

After the creation of the control laws, these were tested on 500 disturbance scenarios evenly distributed in the disturbance domain (Fig. 4a), in order to asses the robustness of the obtained control laws. The results of this comparison are listed in Table 1.
The major highlights of this comparison are:

- From the column "Disturbance Cases Solved" in Table 1, it can be seen how the IGP control laws can perform successfully on average on 207/500 disturbance scenarios while the SGP ones on 81/500, hence a greater robustness is achieved.

Table 1: Results of GP control law creation using the Inclusive GP and Standard GP. The highlighted values indicates which GP algorithm performed better on the considered disturbance scenario.

	Fitness 1	Fitness 2	Successful	Length Individual 1	Depth Individual 1	Length Individual 2	Depth Individual 2	Disturbance Cases Solved
IGP	**0.8688**	0	no	65	14	26	12	**20/500**
SGP	1.9712	0	no	2	1	6	4	0/500
IGP	**0.5427**	0	no	71	23	9	6	60/500
SGP	0.7811	0	no	2	1	8	2	**150/500**
IGP	**0.4832**	0	no	3	1	15	5	**73/500**
SGP	1.8142	0	no	2	1	5	4	1/500
IGP	**0.6076**	0	**yes**	57	17	14	6	44/500
SGP	0.6603	0	no	2	1	2	1	**209/500**
IGP	**0.4986**	0	**yes**	41	16	75	18	**436/500**
SGP	0.9315	0	no	2	1	4	3	45/500
IGP	**0.5272**	0	**yes**	68	21	35	13	**248/500**
SGP	1.0006	0	no	6	4	2	1	57/500
IGP	**0.4985**	0	**yes**	21	7	88	14	**428/500**
SGP	0.7328	0	no	2	1	4	3	104/500
IGP	**0.7218**	0	no	10	5	122	25	**276/500**
SGP	0.7888	0	no	2	1	5	3	186/500
IGP	**0.5520**	0	**yes**	75	14	100	25	**434/500**
SGP	1.0065	0	no	2	1	2	1	50/500
IGP	**0.7602**	0	no	33	9	5	3	**52/500**
SGP	1.1208	0	no	2	1	2	1	8/500

- Looking at the columns from "Length Individual 1" to "Depth Individual 2" in Table 1, it can be seen how the SGP can always find substantially smaller individuals than IGP. Nonetheless, they outperform those found with IGP in terms of robustness (last column in Table 1) only two times out of 10. Moreover, in terms of fitness 1 values, the individuals created with SGP always perform worse than those created with IGP. These observations suggest that indeed more complex (e.g. bigger) individuals have better performances than smaller ones hence they are capable of capturing more efficiently the nonlinearties of the plant.
- Using the same number of maximum generations and population size, the SGP is never able to find a successful control law on the disturbance scenario of design, while the IGP is able to find a successful control law 50% of the time as reported in the column "Successful" in Table 1. Here successful means that the final values of the states are within 1% range of their reference values. This suggests that IGP can find better solutions than SGP with a smaller computational budget.

4 Test Case

The system considered to test the proposed controller is the FESTIP-FSS5 Single-Stage-to-Orbit vehicle (D'angelo et. al. [3]). The main peculiarities of such vehicle are its lifting body shape and the use of an aerospike engine, which can be used during the entire ascent trajectory. This vehicle is very different from

a standard multistage rocket, having greater control capability, and hence is an interesting framework for the design and testing of intelligent controllers. Here, only those aspects that influence the design of the controller will be described. For a more detailed description of this vehicle please refer to [3].

The aim of the considered controller is to perform the guidance of the vehicle by tracking a reference trajectory. In particular the reference trajectory obtained in [11] is considered.

4.1 Disturbance Scenario

The control capabilities of the proposed controller are tested by simulating different disturbance scenarios consisting in a gust acting in a certain altitude range with constant intensity and by considering uncertainties in the aerodynamic and atmospheric models. The uncertainties formulation was taken from [13].

Each disturbance scenario is described by three parameters as in Eq. (1)

$$disturbance\ scenario = (h_{start}, v_g, \Delta_g) \tag{1}$$

where $h_{start} \in [1, 30]\ km$ is the altitude at which the gust starts, $v_g \in [1, 60]\ m/s$ is its intensity and $\Delta_g \in [5, 15]\ km$ is the width of the gust zone. The gust is applied as in Eq. (2)

$$v = \begin{cases} v, & h < h_{start} \\ v - v_g, & h_{start} \leq h \leq h_{start} + \Delta_g \\ v, & h > h_{start} + \Delta_g \end{cases} \tag{2}$$

5 Results

The code for the algorithms and the models have been implemented in Python 3 and rely on the open source library DEAP [4] for the GP part, and Tensorflow for the NN. All the simulations were run on a Laptop with 16GB of RAM and an Intel® Core™ i7-8750H CPU @ 2.20GHz × 12 threads and multiprocessing was used. The code developed in this work is open source and can be found at https://github.com/strath-ace/smart-ml.

In this section, other than presenting the final results on the performances of the proposed controller, also the influence of the optimization algorithm used to create the training dataset and the architecture of the employed NN are analyzed in order to understand their influence on the whole controller creation process.

Note that the formulation of the test case used in this section does not consider the constraints that were implemented during the creation of the control law using GP in Section 3. This because the aim of the next steps of the proposed approach (the optimization of the GP control law and the control tests using also the NN) is to maintain the trajectory within 1% from the reference one, and then the constraints satisfaction is implied.

The training dataset for the NN was produced by performing an optimization of the best performing control law obtained with IGP (fifth row in Table

1) on 500 different disturbance scenarios evenly distributed in the disturbance domain (Fig. 4a). The goal of the optimization process is to find the optimal GP control law parameters explained in Section 2 in order to obtain a controlled trajectory which is within 1% range of the reference trajectory. The optimal GP parameters and the related trajectory are then stored to produce the training dataset for the NN. In order to understand the influence of the employed optimization algorithm on the whole process and on the training phase, a comparison of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) and Nelder-Mead (NM) algorithms was made and the results are listed in Table 2. These two algorithms were chosen since both are implemented in the *scipy optimize* library and BFGS is a gradient based method, while NM is a direct search method.

From these results it can be seen that BFGS is six times faster and can converge better than NM since it can perform a successful optimization of the GP control law on 24.6% more disturbance scenarios than the latter. Such better performances can be explained by the fact that, by being gradient based, BFGS can converge faster to the local minimum, while NM makes use of the whole optimization budget at its disposal (greater computational time) but without using the information from the gradient it is not able to successfully converge to the minimum.

Table 2: Comparison of two different optimization algorithms

	BFGS	Nelder-Mead
Successes	439/500	316/500
Computational Time	1 day 03h27m27s	6 days 12h33m19s

The results of the optimization using both algorithms were used to produce two different training datasets for the NN which were structured as in 3.

$$training\ dataset = \begin{bmatrix} \boldsymbol{t'_1} & \boldsymbol{x'_{11}} & \dots & \boldsymbol{x'_{1s}} & \boldsymbol{err'_{x11}} & \dots & \boldsymbol{err'_{x1s}} & \boldsymbol{C_{11}} & \dots & \boldsymbol{C_{1n}} \\ \boldsymbol{t'_2} & \boldsymbol{x'_{21}} & \dots & \boldsymbol{x'_{2s}} & \boldsymbol{err'_{x21}} & \dots & \boldsymbol{err'_{x2s}} & \boldsymbol{C_{21}} & \dots & \boldsymbol{C_{2n}} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \boldsymbol{t'_m} & \boldsymbol{x'_{m1}} & \dots & \boldsymbol{x'_{ms}} & \boldsymbol{err'_{xm1}} & \dots & \boldsymbol{err'_{xms}} & \boldsymbol{C_{m1}} & \dots & \boldsymbol{C_{mn}} \end{bmatrix} \quad (3)$$

The *training dataset* matrix in Eq. (3) has dimensionality $[m \times (2s + n + 1)]$, where m denotes the different optimizations performed, s is the number of the states parameters, and n refers to the optimized parameters typical of the considered GP equation. In this case, each component represented in Eq. (3) is a column vector containing 500 points of the performed trajectory, e.g. $\boldsymbol{x'_{11}}$ contains the trajectory of the state x_1 obtained using the optimized values of the GP control laws from $\boldsymbol{C_{11}}$ to $\boldsymbol{C_{1n}}$. The same explanation is valid for the tracking errors, which are measured as the differences between the obtained trajectory and the reference one.

The training datasets obtained with both algorithms were used to train three different NN architectures with fully connected layers: A) one hidden layer with 30 neurons per layer (Configuration 30 with 1,629 trainable parameters); B) two hidden layers with 25 neurons per layer (Configuration 25x25 with 2,014 trainable parameters); C) two hidden layers with 30 neurons per layer (Configuration 30x30 with 2,559 trainable parameters). The NNs were then tested five times on a predefined set of 500 disturbance scenarios obtained randomly and uniformly distributed in the disturbance domain (Fig. 4b). These 500 disturbance scenarios are different than those used in the previous steps of the process in order to further test the robustness of the proposed controller. The three NN architectures were chosen in order to observe the effects of the variation of the number of trainable parameters on the overall performances. The obtained results are listed in Table 3.

Table 3: Success rates (%) of the three different NN configurations tested on five different runs on the same test points using the NN trained on the training datasets produced with the BFGS and Nelder-Mead optimization.

	BFGS						Nelder-Mead					
	Success rate (%)						Success rate (%)					
	Run 1	Run 2	Run 3	Run 4	Run 5	Mean $\pm$ σ	Run 1	Run 2	Run 3	Run 4	Run 5	Mean $\pm$ σ
	Configuration 30											
NN	71.0	70.2	68.2	71.2	68.0	**(69.72 $\pm$ 1.36)**	68.0	68.0	71.0	67.2	69.6	**(68.76 $\pm$ 1.13)**
GP	67.8	70.6	70.0	69.0	67.8	**(69.04 $\pm$ 0.43)**	65.6	65.6	66.6	65.6	66.8	**(66.04 $\pm$ 1.41)**
Global	84.8	84.6	83.6	84.0	84.2	**(84.24 $\pm$ 0.50)**	82.6	82.6	82.2	81.6	84.8	**(82.76 $\pm$ 0.92)**
	Configuration 25x25											
NN	69.0	69.4	69.6	72.6	71.6	**(70.44 $\pm$ 1.34)**	66.0	70.6	69.0	70.6	67.2	**(68.68 $\pm$ 1.72)**
GP	70.6	71.2	70.2	71.6	70.6	**(70.84 $\pm$ 0.97)**	62.4	69.4	66.2	70.2	68.0	**(67.24 $\pm$ 1.36)**
Global	86.2	84.8	83.8	85.6	86.2	**(85.32 $\pm$ 0.54)**	80.0	82.6	82.2	82.8	82.2	**(81.96 $\pm$ 1.08)**
	Configuration 30x30											
NN	71.2	71.6	68.2	70.8	72.0	**(70.76 $\pm$ 1.84)**	68.6	68.0	68.6	69.2	69.0	**(68.68 $\pm$ 2.78)**
GP	71.0	68.6	69.8	72.8	68.0	**(70.04 $\pm$ 1.00)**	68.8	68.0	69.4	67.0	68.8	**(67.24 $\pm$ 0.41)**
Global	85.0	84.8	83.0	84.4	86.0	**(84.64 $\pm$ 0.83)**	82.4	83.4	84.4	82.4	81.2	**(82.76 $\pm$ 1.08)**

In Table 3, *success rate* means how many successful trajectories (within 1% from the reference trajectory) were obtained over 500 different disturbance scenarios (Fig. 4b). The results in Table 3 were obtained by performing a control action on the selected disturbance scenario using in parallel: 1) the non optimized GP control law (GP approach, rows GP in Table 3); and 2) the GP control law always optimized by the NN (NN approach, rows NN in Table 3). *Global*, in Table 3, refers to the total success rate achieved when considering both the GP and NN approaches. This comparison highlighted how when using alone either the GP approach or the NN one, the success rate is $\sim 70\%$ ($\sim 68\%$ for those obtained with NM) of the tested disturbance scenarios. While if the global amount of success is considered, the success rate gets to $\sim 85\%$ ($\sim 82\%$ for those obtained with NM). This suggests that the two approaches NN and GP can be complementary, that is, the GP approach is successful on some disturbance scenarios where the NN approach fails and vice-versa. The observed

complementarity indicates that a controller as depicted in Fig. 3 could be effective by enhancing the GP control law performances optimizing it online with a NN when the GP control law fails, which is, when the tracking error on the states becomes greater than a certain threshold. It can also be observed that, in line with the results obtained from the BFGS and NM comparison, the lower success rate of the NM algorithm translates into a smaller training dataset for the NN than the one obtained with the BFGS algorithm, which as a consequence influenced the precision and success rate of the NN.

6 Conclusion

In this work, a new hybrid approach to IC, consisting in an NN based online optimization of a GP control law produced offline, has been proposed. Results coming from tests with different settings of disturbances show that the approach is more robust that using only the GP trained offline.

Moreover, the creation of the training dataset using two different optimization algorithm showed the importance of this phase, highlighting how using different optimization methods can lead to different results. In fact, a gradient based method produced better results both in terms of accuracy (24.6% more successes) and of computational time (six times faster) than a direct search method. As a consequence, the controller trained with the dataset produced using the BFGS algorithm achieved a slightly greater robustness on test cases than the one trained with the NM algorithm.

To improve the performance of the GP, two new features were also devised, implemented and tested: Inclusive Tournament and Inclusive Crossover. The Inclusive Tournament and Inclusive Crossover were designed to improve the robustness of the GP control law since it is a key aspect in control applications. In comparison with a standard implementation of GP using standard single point crossover and Double Tournament selection, the control law created with IGP can control the plant successfully on average on 25.2% more disturbance cases than the one created with the SGP.

Different future research directions lie ahead: the creation of the training dataset for the NN requires further investigation in order to assess the possible benefits of using a global optimizer instead of a local one and also to understand if the creation of the training dataset could be avoided in order to speed up the process, for example by using a RL framework. Different NN architectures will be tested, e.g. a Radial Basis Function (RBF) network, in order to assess which could be the best NN configuration for this application. Finally a more in depth analysis of the Inclusive Crossover and Tournament will be performed in order to fully understand the extent of the introduced improvements in comparison to the SGP.

References

1. Antsaklis, P.J.: Defining Intelligent Control. Report to the Task Force on Intelligent Control. IEEE Control Systems Society pp. 1–31 (1993)

2. Chiang, C.H.: A genetic programming based rule generation approach for intelligent control systems. 3CA 2010 - 2010 International Symposium on Computer, Communication, Control and Automation **1**, 104–107 (2010). https://doi.org/10.1109/3CA.2010.5533882
3. D'Angelo, S., Minisci, E., Di Bona, D., Guerra, L.: Optimization Methodology for Ascent Trajectories of Lifting-Body Reusable Launchers. Journal of Spacecraft and Rockets **37**(6) (2000)
4. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagńe, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research **13**, 2171–2175 (2012)
5. Gomez, F., Miikkulainen, R.: Efficient Non-linear Control Through Neuroevolution. In: J., F., T., S., M, S. (eds.) Machine Learning: ECML 2006. Springer, Berlin, Heidelberg (2006)
6. Gomez, F.J., Miikkulainen, R.: Active guidance for a finless rocket using neuroevolution. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **2724**, 2084–2095 (2003)
7. Johnson, E., Calise, A., Corban, J.E.: Reusable launch vehicle adaptive guidance and control using neural networks. In: AIAA Guidance, Navigation, and Control Conference and Exhibit (2001). https://doi.org/10.2514/6.2001-4381
8. Kamio, S., Mitsuhashi, H., Iba, H.: Integration of Genetic Programming and Reinforcement Learning for Real Robots. In: Genetic and Evolutionary Computation — GECCO 2003. pp. 470–482. Springer, Berlin, Heidelberg (2003)
9. Koza, J., Keane, M., Yu, J., Bennett III, F., Mydlowec, W.: Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming. Genetic Programming and Evolvable Machines **1**(1/2), 121–164 (2000). https://doi.org/10.1023/A:1010076532029
10. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **2439**, 411–421 (2002)
11. Marchetti, F., Minisci, E., Riccardi, A.: Single-Stage to Orbit Ascent Trajectory Optimisation with Reliable Evolutionary Initial Guess. [SUBMITTED TO] Optimization and Engineering
12. Marchetti, F., Minisci, E., Riccardi, A.: Towards Intelligent Control via Genetic Programming. 2020 International Joint Conference on Neural Networks (IJCNN) (2020)
13. Pescetelli, F., Minisci, E., Maddock, C., Taylor, I., Brown, R.E.: Ascent trajectory optimisation for a single-stage-to-orbit vehicle with hybrid propulsion. In: 18th AIAA/3AF International Space Planes and Hypersonic Systems and Technologies Conference 2012. pp. 1–18 (2012)
14. Rosca, J.P.: Entropy-Driven Adaptive Representation. Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications pp. 23–32 (1995), ftp://ftp.cs.rochester.edu/pub/u/rosca/gp/95.ml.gpw.ps.gz
15. Salichon, M., Turner, K.: A neuro-evolutionary approach to Micro Aerial Vehicle control. Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10 pp. 1123–1130 (2010). https://doi.org/10.1145/1830483.1830692
16. Saridis, G.N.: Toward the Realization of Intelligent Controls. Proceedings of the IEEE **67**(8), 1115–1133 (1979). https://doi.org/10.1109/PROC.1979.11407
17. Shir, O.M.: Niching in Evolutionary Algorithms. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing. Springer Berlin Heidelberg (2012)

18. Wilson, C., Marchetti, F., Di Carlo, M., Riccardi, A., Minisci, E.: Classifying Intelligence in Machines : A Taxonomy of Intelligent Control. Robotics **9**(3), 64 (2020). https://doi.org/10.3390/robotics9030064