Cooperative Co-Evolutionary Genetic Programming for High Dimensional Problems

Lino Rodriguez-Coayahuitl[1][0000−0002−7541−4772], Alicia Morales-Reyes[1][0000−0001−5052−7554], Hugo J. Escalante[1,2][0000−0003−4603−3513], and Carlos A. Coello Coello[2][0000−0002−8435−680X]

[1] Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla PUE 72840, Mexico {linobi,a.morales,hugojair}@inaoep.mx
[2] CINVESTAV-IPN,* Departamento de Computación, Mexico City, Mexico. ccoello@cs.cinvestav.mx

Abstract. We propose a framework for Cooperative Co-Evolutionary Genetic Programming (CCGP) that considers co-evolution at three different abstraction levels: genotype, feature and output level. A thorough empirical evaluation is carried out on a real-world high dimensional ML problem (image denoising). Results indicate that GP's performance is enhanced only when cooperation happens at an output level (ensemble-alike). The proposed co-evolutionary ensemble approach is compared against a canonical GP implementation and a GP customized for image processing tasks. Preliminary results show that the proposed framework obtains superior average performance in comparison to the other GP models. Our most relevant finding is the empirical evidence showing that the proposed CCGP model is a promising alternative to specialized GP implementations that require knowledge of the problem's domain.

Keywords: Genetic Programming · Evolutionary Machine Learning · Co-Evolutionary Algorithms · Ensemble methods · Image processing.

1 Introduction

High dimensional problems have been traditionally challenging for both Machine Learning [2] (ML) methods and Evolutionary Algorithms [31] (EAs). This issue is critical for Genetic Programming (GP), which in this case is *the evolutionary learning algorithm*, because of its complex structures and its need for large populations to reach acceptable solutions. Therefore, large scale learning problems have been out of the scope of GP-based solutions, hindering its raise as a competitive learning model.

In ML, several techniques have been devised to adapt learning algorithms to high dimensional problems. A successful example are convolutional neural networks that process images at a pixel-level [13]. In contrast, a mechanism

* The last author gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencia 2016*) and from a SEP-Cinvestav grant (application no. 4).

commonly adopted in GP to deal with high dimensional problems (e.g., image processing) uses special (high-level) primitives capable of processing groups of features from the input representation altogether. Thus, nodes in GP can represent sets of features or functions to process them (e.g., a mean function over an input space region) [1, 22]. This approach has resulted in satisfactory performance in some domains [7, 26, 16]. However, it usually requires a form of problem's domain knowledge, which contravenes the spirit of automated ML systems.

From a pure evolutionary computation (EC) standpoint, an approach that has been used for a long time to tackle large scale problems is Cooperative Co-Evolutionary Algorithms [21] (CCA). Potter & De Jong [21] originally proposed the CCA framework to tackle complex optimization problems through Genetic Algorithms (GA) by splitting the search space into multiple, smaller, sub-problems that are solved by semi-independent GA populations cooperating to solve the original (larger) problem. It is well documented that CCA has enabled several EAs to operate on very high dimensional optimization problems [17, 31, 19]. This evidence motivated us to explore CCA's suitability in a GP context for high-dimensional learning problems.

We propose alternative mechanisms for implementing a Cooperative Co-evolutionary GP (CCGP) and assess their performance in a high dimensional learning problem composed by more than four hundred feature variables: *image denoising*. Our working hypothesis is that a cooperative co-evolutionary approach will allow GP to scale its performance in ML problems with hundreds of input feature variables, without having to resort on high-level primitives. Therefore, the main contributions of this work are threefold:

- We introduce cooperative co-evolution in GP as a way to tackle high-dimensional ML problems. The proposed GP formulation obtains competitive performance in very high-dimensional and complex problems while directly processing raw data (pixels).
- We propose three different approaches to perform cooperative co-evolution in GP aiming at high dimensional problems. We experimentally compare their performance in a real-world, high-dimensional ML problem (natural image denoising).
- The best performing CCGP approach is evaluated extensively and compared to highly competitive baseline algorithms. Our experimental results show the superiority and competitiveness of our proposed approach.

Results indicate that CCGP is a viable alternative to the standard GP approach for high dimensional problems. This is important because in the proposed CCGP scheme, no special nodes are defined for the problem at hand, whereas in the typical GP model for high dimensional problems, those special nodes represent the weakest link in the design process of a GP-based ML solution, mainly because special nodes may require human expert knowledge of the problem's domain. In contrast, our proposed CCGP approaches are completely agnostic, therefore posing GP a step towards automation, and closer to modern general purpose ML frameworks, such as Deep Learning.

2 Related Work

According to [12], when approaching ML problems with GP, there are different abstraction levels to perform CCGP: (1) *genotypic*, (2) subroutine or *feature*, and (3) output or *ensemble* levels. In genotypic CCGP, co-evolving components fusion takes place at the individual representation level, by merging trees directly (in tree-based GP); for CCGP at feature level, co-evolving GP processes generate intermediate input data representations that can be fed into a ML model in order to enhance its performance (i.e., feature extraction). Finally, in a CCGP ensemble, multiple co-evolved species' outputs vote or average in order to achieve higher accuracy in classification or regression problems. In this paper we propose a novel framework to perform cooperative co-evolution at those abstraction levels, and experimentally compare their performance.

CCGP has been mostly studied at ensemble and feature levels [10, 11, 33, 20]. It should be noted, however, that the originally proposed CCA framework considers fusion at a genotypic level [21]. CCA research at a genotypic level in GP is scarce; one of the few works that covers this subject is presented in [12]. Krawiec & Bhanu presented several works on feature-level CCGP [10, 11], and in [12] they proposed a genotype-level CCGP and compared it to their previous approaches. However, it should be noted that [12] covers CCGP only for linear GP [3], and not for the original tree-based GP. A possible reason for this lack of interest could be the fact that performing genotype-level CCGP with the standard tree individual representation is difficult, since there is no obvious way to fuse genotypes for tree-based individuals other than standard GP-subtree crossover, and this might not yield the desired effect in a CCA scheme. Moreover, while in [10] and [11], cooperative co-evolution happens at feature level, the prediction stage is relegated to a different, simple ML algorithm, instead of using another GP process. In contrast, we propose to perform fusion at a feature level through a co-evolving GP species, thus effectively implementing a multi-layer GP system. This multi-layer GP is another relevant contribution of this paper.

On the other hand, GP-based ensembles have been proposed at least as early as in [9]. Many GP-based ensembles found in the literature follow a standard weighted averaging fusion technique [9, 28, 29]. In contrast, herein we propose to generate ensembles through an explicit co-evolutionary framework. Co-evolution generated GP ensembles have been thoroughly researched by Heywood et al. [14, 18, 15, 6]. However, their problem decomposition technique happens mainly at sample subset level, whereas we propose a feature subset approach.

Regarding GP approaches to tackle high dimensional problems, two are the most widely used: (a) using special primitives to process groups of features altogether [1, 22], and (b) using separate GP processes after applying clustering to input variables [27, 23]. The methods proposed here fall in the second category. Both approaches have disadvantages. In the first, it is required to define special nodes, and this might imply requiring some previous knowledge of the problem's domain, while in the second, division and execution of multiple GP processes may involve an increase in the computational cost. A main contribution in this paper is that, for the first time both GP approaches are directly compared.

3 Problem Statement

In supervised learning, ML algorithms search for a function f, mapping inputs ($\mathbf{x} \in \mathbb{R}^n$) to outputs ($y \in \mathbb{R}$) starting from a dataset of input-output pairs ($\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i \in 1,\ldots,d}$). For a regression problem with inputs in an $n-$dimensional space we have $f : \mathbb{R}^n \rightarrow \mathbb{R}$. That is, f receives as input n feature variables in order to make a scalar prediction. GP as a non-parametric ML method builds f from scratch by using primitives and feature variables as building blocks.

In high-dimensional learning problems, n is large enough so that f can become difficult to infer because GP needs to search among large tree structures that accommodate enough n feature variables in order to perform satisfactorily. Hence, this simple problem can be associated to an intractable search space.

We claim that it is less complex to search for multiple, *simpler*, functions $\tilde{f}_i$, such that by combining their outputs, they may outperform f. In our context, by *simpler* we mean that they are represented by smaller GP trees, and are restricted to a limited subset of features, and therefore can be more easily discovered by GP. In formal terms, our hypothesis is that it is computationally more efficient to search for p, lower dimensional, sub-functions $\tilde{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}$, such that $m << n$, that when combined can yield a $\tilde{f}$ function equivalent or even with superior performance than f. Two questions arise: (1) how can the original feature space be split?, and (2) how can we combine p sub-functions f_i?

In this work, we hold that in order to get a highly automated and agnostic ML design process, the feature space should be split in a random way. However, we recognize that some previous knowledge of the problem's domain can be used in order to perform an advantageous partition of the initial input representation. On the other hand, a correct method to merge partial solutions can be more difficult to assert which is the main topic in this research. In the next section we propose and discuss some possible approaches to address such task.

4 Cooperative Co-evolutionary GP

In general terms, CCAs split the search task into multiple, smaller, optimization processes. The main idea is to introduce *modularity* in EAs [21]. In combinatorial and numerical optimization problems, CCAs achieve this by distributing solutions' segments among a number of sub-populations; individuals' evaluation in each sub-population is performed by importing those segments from other sub-populations and assembling complete solutions for evaluation. Thus, individuals take turns to form part of such complete solutions and credit can be assigned to each one of them.

In order to import such problem decomposition strategy into the context of syntax tree-based GP (i.e., not LinearGP), we propose to introduce the concept of *main species*, that represents a partial solution that acts as a *holder* to which the rest of partial solutions attach to, in order to form a complete solution. Next, we detail three proposed approaches developed within the CCGP framework herein introduced. Each method is a CCA with standard tree-based

GP representation at genotype, feature or output level of the ML pipeline. The proposed methods adhere to the following procedure:

1. The input feature set is sampled, with replacement, to form p subsets with m randomly picked features variables each;
2. p species are created; each subset limits valid terminals for each species; all species are confined to one subpopulation (i.e., no inter-species breeding);
3. Additionally, there is a *main* species, that represents the type of individuals to which all other species *attach* to form a complete candidate solution;
4. At each generation, a complete candidate solution is assembled by randomly selecting one individual from each species and attaching them to one main species individual (also selected at random);
5. This complete candidate solution is sent to all subpopulations; each individual is evaluated by attaching it to the complete candidate solution (in order to form a complete chain of execution), for fitness assignment; this also applies to the main species individuals. Details of this procedure are given in Sec. 4.4;
6. The evolutionary process (evaluation, recombination, selection) occurs simultaneously in all subpopulations;

The differences among the proposed CCGP variants rely on the complete solutions assembly process and on the form that the main species take. These variants are described in detail next.

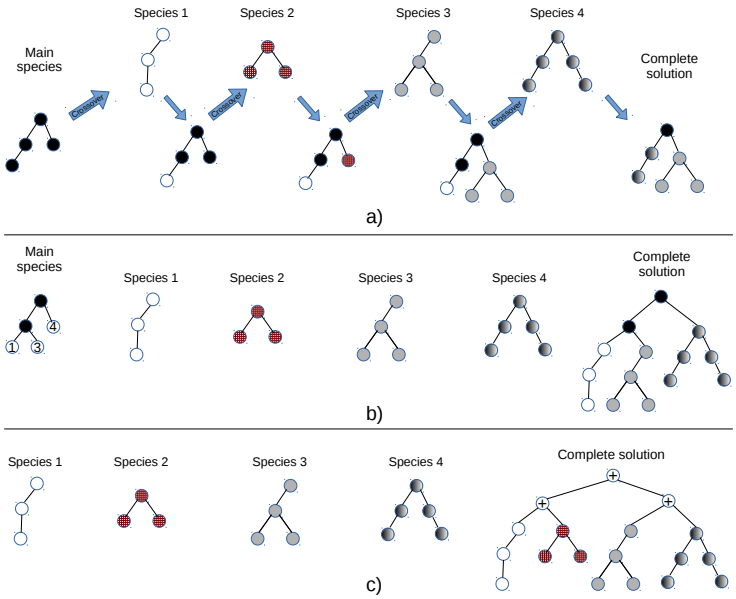

Fig. 1. Proposed CCGP models. From top to bottom: genotype-, feature- and ensemble-level models.

4.1 Genotype level

CCGP at genotype level occurs by fusing multiple sub-components while directly mixing syntax trees that represent each co-evolving species. Since the straightforward method to perform such recombination is the standard subtree crossover defined for GP, this is the method we propose to build complete solutions. Thus, subtree crossover is performed sequentially between a main species candidate and each additional co-evolving species. The idea here is that complete candidate solutions have useful subtrees that rely on a wide variety of input variables (because each species is limited to a certain subset of input variables). This is the most straightforward form that CCGP may take, and it is aligned to the original CCA framework proposed by Potter & De Jong. Fig. 1a shows the complete solution assembly process under this approach.

4.2 Feature level

We call fusion at *feature level* when species (other than the main one) represent encapsulated sub-components, that is, complete GP trees that merge to form a complete solution by connecting to leaf nodes of a main species individual. This scheme removes inconsistencies of genotype-level/fusion by crossover, and preserves integrity of both species and the main species individuals. Our aim is that co-evolved sub-components may represent *pseudo*-subroutines that act as a sort of feature extractor stages, while the main species act as the predictor stage that operates over those pre-processor stages' outputs, rather than having to work with a raw and large scale input space (hence the name *feature-level*).

In formal terms, this approach fuses p sub-functions f_i, by searching for a function (also by means of co-evolution) $g : \mathbb{R}^p \rightarrow \mathbb{R}$, that operates over auxiliary sub-functions outputs such that $\tilde{f} = g(f_1, f_2, ..., f_p)$. Fig. 1b shows this model.

4.3 Ensemble Level

At an ensemble or output CCGP level, each species represents a complete predictor to the problem at hand, and species fusion occurs by only aggregating the output generated by each predictor. Aggregation may take several forms. In this work, we propose to combine each species output by means of a simple sum. This approach bears some resemblance to ML ensemble methods [4] that combine multiple predictors outputs by a weighted sum (hence the name, *ensemble*-level). We proposed this model after observing an undesirable phenomenon in the feature-level CCGP where main individuals' function converged ignoring all but one of the sub-component species, and the search process then happened only in a single population, losing model's co-evolutionary nature and becoming a standard EA (this issue is detailed and discussed in Section 5.3). Therefore, through this approach, it becomes more difficult for species to avoid contributing to the global solution. Notice how this approach can be seen as a case where the main function is fixed to a GP tree composed by sum nodes only, and the evolutionary search for a main function is discarded. Fig. 1c depicts this concept. Thus, formally, this method proposes that $\tilde{f} = f_1 + f_2 + ... + f_p$.

4.4 Fitness Assignment

Fitness assignment in CCGP (step 5 in the general procedure) is carried out in two different ways. At the Genotype level, attaching each individual to a complete candidate solution that is sent to each species population, occurs by performing crossover between that assessed individual and the complete solution. In CCGP models at Feature and Ensemble level, attachment of individuals for evaluation happens by *replacing* the corresponding individual of that species in the complete candidate solution being used.

5 Experimental results

This section describes the empirical methodology followed for evaluation of proposed CCGP approaches. It also presents and discusses the obtained results.

5.1 Datasets

For validation, the proposed CCGP framework tackles image denoising as a high dimensional problem, where a clean image $\mathbf{x}$ is extracted from a noisy observation $\mathbf{y}$ such that, for an additive noise model, $\mathbf{y} = \mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is a contamination process. In this study, Additive White Gaussian Noise (AWN) is targeted, where $\mathbf{v}$ follows a Gaussian distribution with some given σ.

We used the Berkeley Segmentation Dataset (BSDS) [24] for training and testing purposes. We converted 200 images from BSDS to grayscale and randomly extracted $14,000$ patches of 21×21 pixels in size. We contaminated images (prior to patch extraction) with AWN noise level $\sigma = 50$. We set all GP variants to attack image denoising as a regression problem: the objective function is the minimization of the average mean square error (MSE), from attempting to predict the noise level in the central pixel from all patches in the training set. In a real life scenario, a generated model with this approach can be slid through a full image, in a convolutional fashion, in order to clean it. However, for this set of experiments, we limited ourselves to test generated models in a testing set comprised also by image patches. We used $12,000$ patches for training and 2000 for testing. BSDS had been used as testbed for different image denoising methods [25, 30, 5], including deep learning approaches [32], to which we compare later in Sec. 5.4. Fig. 2 shows sample images from BSDS.

Fig. 2. Sample images from the Berkeley Segmentation Dataset.

5.2 Parameters Settings

Table 1 summarizes parameters configuration for all experimental samples. Max, min and mean primitives are 2-arity functions that operate over two single scalars, and division is protected such that any attempt at dividing by zero returns 0. Both crossover and mutation are protected so that the maximum allowable tree depth is never exceeded. Training datasets are split in non-overlapping minibatches of 300 instances and at each generation, populations are evaluated using one minibatch. Minibatch-based evolution (on-line learning) in GP has been found to be successful for this type of ML problem [8, 23].

Table 1. Parameters configuration for empirical testing.

Parameter	Value
Pop Size	400 (per species, inc. main)
Generations	Variable (24 hrs.)
No. of species	8 + main
Max Tree Depth	6 (for all species, inc. main)
Crossover / Mutation rate	0.5 / 0.5
Pop Dynamics	Steady State
Primitives	$+$, $-$, $\times$, $\div$, x^2, sin, cos, $\sqrt{}$, max, min, mean, ReLU
Terminals	Individual pixels and constants within range $[-1, 1]$
Features per species	30 (from a total of 441)

In order to allow a fair comparison, all setups are run for the same limited amount of time. Subsets of feature variables that are allowed for each species are randomly assembled. However, since the target task predicts central pixel's noise level within image patches, it is set, as a requirement, that a central pixel appears in at least two random subsets.

For the feature-level approach, another restriction is defined: the main species' individuals must use, at least, 6 out of 8 total sub-component species. Main species' trees are parsed and the number of different species used by an individual are counted. If this constraint is not met, the fitness of such individual is set to ∞. This restriction is set in order to prevent the main species' individuals from becoming a "wired" function that connects to a single sub-component species, where the whole optimization process is confined to, while the rest of the species do not contribute to the co-evolutionary search.

5.3 Analysis of Results

Table 2 shows results obtained for all approaches tested after performing 10 independent runs. Results are shown in decibels Peak Signal to Noise Ratio (dB PSNR), such that higher is better. Fig. 3a depicts the fitness evolution in a feature-level run, in error terms; in this case, lower is better. This behavior is representative of feature-level runs in general. This result shows that the feature-level approach does converge, suggesting that this CCGP variant is capable of evolving multi-layer GP structures with sequential dependencies. However, a closer examination to the best solutions rendered by this approach revealed that

this is not the case (discussed below). Moreover, the feature-level approach is no match for ensemble models, which converge faster, and to better solutions, both in average and overall. Meanwhile, genotype-level is left further behind. This result indicates that ensemble CCGP is the best overall method.

Low performance of genotype-level CCGP can be explained by the fusion mechanism used in this approach: subtree crossover is a stochastic operation that even when given the same two parents trees, may render different offspring if performed multiple times. This means that even if a combination of different species individuals that rendered a good complete solution in a previous generation, are chosen again to form a complete solution, this time their merge may result in a bad complete solution; instability in this mechanism is very high for this approach to converge to any acceptable solutions.

Table 2. Results in dB for tested setups with different time frames.

	Output		Feature		Genotype	
	Avg	Best	Avg	Best	Avg	Best
4 Hrs.	**18.90 ± 0.52**	**19.46**	16.43 ± 1.64	18.73	14.80 ± 0.44	15.41
12 Hrs.	**20.62 ± 0.36**	**21.16**	17.67 ± 2.30	20.84	15.09 ± 0.25	15.41
24 Hrs.	**20.96 ± 0.50**	**21.61**	18.55 ± 2.27	21.35	15.09 ± 0.44	15.62

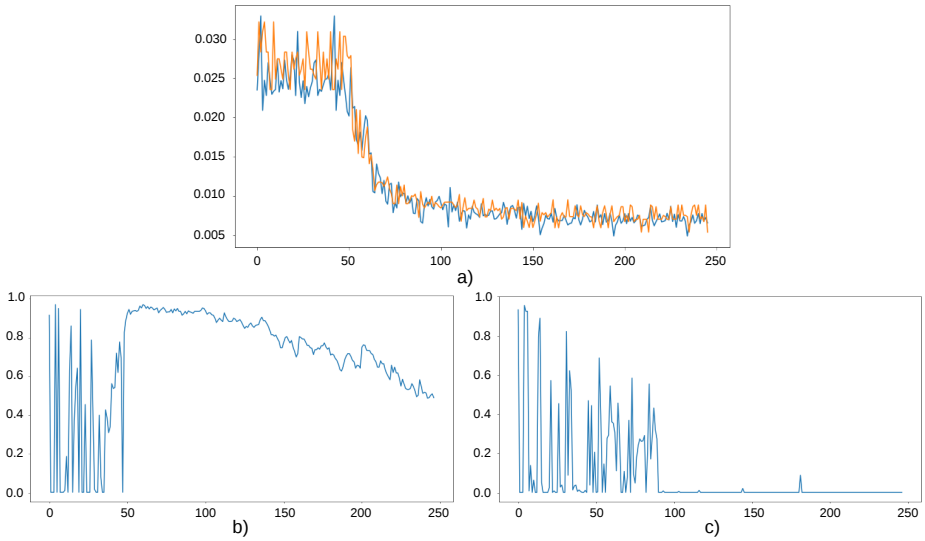

Fig. 3. Fitness and diversity from a feature-level CCGP run as generations elapse. (a) Fitness error (lower is better). In blue (orange) we show the training (testing) error. (b) and (c), phenotypic diversity in two sub-component populations, one (zero) means all solutions yield a different (the same) prediction.

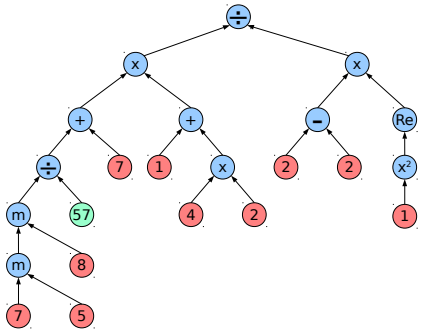

Fig. 4. Main species' individual example that complies with referencing at least 6 sub-components and not using them. Notice that root's right subtree always returns 0; thus, the whole tree also returns a constant.

Considerable higher performance at ensemble-level with respect to feature-level can be more arguable. Flexibility at feature-level, which also searches for an optimal main function, could play on its favor against a more restricted ensemble model with its predefined main function - given enough computational time. However, this appears not to be the case. On the contrary, this disadvantage is clearer in the first few generations of feature-level runs, when the evolutionary search appears to stagnate (according to Fig. 3a) apparently searching for a minimally acceptable main function. Even once feature-level CCGP variant escapes initial stagnation, it does not converge to solutions that reach performance of ensemble-level models.

Some dominant solutions are closely examined during this early evolutionary process, and phenotypic diversity is measured in each sub-population. It is observed that feature-level CCGP is surprisingly good at ignoring restriction imposed to the main species' individuals of using co-evolving sub-components: this CCGP variant managed to generate main species' individuals with subtrees that referenced too many sub-components, while at the same time completely ignored such subtrees. Fig. 4 depicts an example of such type of individuals.

Analyzing phenotypic diversity in species' sub-populations confirmed this behavior. Fig. 3b shows diversity in a sub-population as generations elapse. It can be observed that at early generations, diversity oscillates between some actual value and zero. When phenotypic diversity abruptly decays to zero, this implies that all individuals in the population generate the same prediction/fitness. This happens because main function individuals are ignoring this sub-component, i.e. regardless of the individual from this population which connects to the main functions, they all render the same result. This oscillating behavior in the first few generations depicted in Fig. 3b means that the main function individuals do not make a consistent use of this particular sub-component. It is only after a certain number of generations that diversity recovers and the approach begins to behave consistently, because now the main function individuals are working cooperatively with individuals from these species.

Further investigation on diversity of all species in different runs, revealed that by relying on mechanisms such as the one depicted in Fig. 4, the feature-level main individual still converges to "wired" functions that simply connect to one or two sub-component populations where optimization is happening, while the rest of the species do not contribute to the co-evolutionary process. As an example of this behavior, Fig. 3c shows diversity of a species' population and the moment it begins to be ignored by the main function individuals, which roughly coincides with the time at which overall CCGP run begins to converge. Meanwhile, ensemble-level models exhibited the exact opposite behavior, where in general, only one or two of eight sub-populations decay to zero diversity, suggesting that they may be acting as constant biasing factors in the sum structure, while the rest of the species do alter global solution performance, contributing to the co-evolutionary search. These results show that multi-layer GP architectures remain as very challenging to evolve.

5.4 Other GP approaches comparison

In this section, ensemble CCGP performance (the best performing model) is compared against two other GP models: a canonical GP representation, and a modern GP variant that makes use of special nodes aimed at high dimensional problems. The aim is twofold: (1) to collect evidence that supports our hypothesis that cooperative co-evolution can boost GP performance in high dimensional learning problems, as well as (2) to quantify how CCGP compares with respect to GP models tailored for tackling high-dimensional problems.

Two GP models called Low-level GP (LowGP) and Mid-Level GP (MidGP) are implemented. LowGP is a canonical GP that operates at individual pixel level; while MidGP uses special nodes and terminals that allow to process features groups. Table 3 summarizes the configurations used for these approaches. LowGP can only make use of primitives and terminals (the same used by CCGP), while MidGP can use special primitives, special terminals, and regular primitives and terminals. Special primitives are functions that receive as inputs variable-length vectors and whose output is a single scalar that can be processed by regular primitives. For an in-depth analysis of these GP models refer to [22].

Notice in Table 3 that non co-evolutionary GPs are setup with the same population size to that of a single CCGP species, but the max tree depth is extended in this case, to account for main functions on top of sub-component species, as well as with an increased number of subtrees (equivalent to sub-component species). Thus, a fair comparison against the proposed CCGP model is guaranteed: both LowGP and MidGP individuals can accommodate the same maximum number of nodes to that of a CCGP complete solution.

Table 4 shows the results obtained for both standard GP variants. It is observed that CCGP average performance is superior to both LowGP and MidGP. The proposed approach also presents a lower variance, thus providing evidence that ensemble CCGP is a viable method to step up GP's performance in high-dimensional learning problems, with the added benefit of not requiring a specialized primitives set. It should be noted, however, that MidGP's best solution

Table 3. Specific tested parameters for non CCGP models

Parameter	Value
Pop Size	400
Max Tree Depth	9
Primitives	Same as in Table 1
Special Primitives	mMean, mMax, mMin, mMed
Terminals	Same as in Table 1
Special Terminals	*Trimmers* 

outclasses CCGP's best result by a considerable margin, indicating that MidGP remains as the reference method to outperform within GP. For a more general comparison, consider that a deep network with 17 hidden layers [32], can score 27.20 dB PSNR given similar training and testing sets.

Table 4. Comparison to non-coevolutionary GPs. Expressed in dB; higher is better.

	LowGP		CCGP		MidGP	
Hrs	Avg	Best	Avg	Best	Avg	Best
4	17.81 ± 2.31	20.94	18.90 ± 0.52	19.46	**20.21** ± 3.74	**23.30**
12	18.13 ± 2.55	21.68	**20.62** ± **0.36**	21.16	20.27 ± 3.74	**23.35**
24	18.23 ± 2.64	21.81	**20.96** ± **0.50**	21.61	20.41 ± 3.82	**23.30**

6 Conclusions

This paper proposed and contrasted three different approaches to perform co-operative co-evolution within the GP framework at different abstraction levels: *genotype*, *feature* and *ensemble*. A thorough behavior and performance analysis of CCGP at a feature-level showed that synthesizing multi-layer GP architectures is surprisingly difficult, because GP tends to confine all optimization processes within a single sub-population, effectively losing properties of a true co-evolutionary search. This is an important result that shreds light on some future research guidelines.

For full empirical assessment, conventional GP variants were also compared. We can conclude that CCGP's performance sits in between a completely agnostic canonical GP, that only processes feature variables at an individual level, and higher-level GP variants that require problem domain knowledge to a lesser or greater extent. This is a very promising result, because with further research, CCGP could boost agnostic GP models to reach the best performances obtained by higher-level GP variants, or maybe to help reducing the amount of designer input knowledge necessary in higher-level GP models.

Acknowledgements. This work was partially supported by CONACyT under project grant A1-S-26314, *Integración de Visión y Lenguaje mediante Representaciones Multimodales Aprendidas para Clasificación y Recuperación de Imágenes y Videos.*

References

1. Al-Sahaf, H., Song, A., Neshatian, K., Zhang, M.: Two-tier genetic programming: Towards raw pixel-based image classification. Expert Systems with Applications **39**(16), 12291–12301 (2012)
2. Alpaydin, E.: Introduction to machine learning. MIT press (2009)
3. Brameier, M., Banzhaf, W.: A comparison of linear genetic programming and neural networks in medical data mining. IEEE Transactions on Evolutionary Computation **5**(1), 17–26 (2001)
4. Brown, G.: Ensemble learning. Encyclopedia of Machine Learning **312** (2010)
5. Chen, Y., Pock, T.: Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(6), 1256–1272 (2016)
6. Doucette, J.A., Mcintyre, A.R., Lichodzijewski, P., Heywood, M.I.: Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. Genetic Programming and Evolvable Machines **13**(1), 71–101 (2012)
7. Esfahanipour, A., Mousavi, S.: A genetic programming model to generate risk-adjusted technical trading rules in stock markets. Expert Systems with Applications **38**(7), 8438–8445 (2011)
8. Gathercole, C., Ross, P.: Dynamic training subset selection for supervised learning in genetic programming. In: International Conference on Parallel Problem Solving from Nature. pp. 312–321. Springer (1994)
9. Iba, H.: Bagging, Boosting, and Bloating in Genetic Programming. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Genetic and Evolutionary Computing Conference (GECCO'99). vol. 2, pp. 1053–1060. Morgan Kaufmann Publishers, San Francisco, California (July 1999)
10. Krawiec, K., Bhanu, B.: Coevolution and linear genetic programming for visual learning. In: Genetic and Evolutionary Computation Conference. pp. 332–343. Springer (2003)
11. Krawiec, K., Bhanu, B.: Visual learning by coevolutionary feature synthesis. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **35**(3), 409–425 (2005)
12. Krawiec, K., Bhanu, B.: Visual learning by evolutionary and coevolutionary feature synthesis. IEEE Transactions on Evolutionary Computation **11**(5), 635–650 (2007)
13. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. The Handbook of Brain Theory and Neural Networks **3361**(10), 1995 (1995)
14. Lemczyk, M., Heywood, M.I.: Training binary gp classifiers efficiently: A pareto-coevolutionary approach. In: European Conference on Genetic Programming. pp. 229–240. Springer (2007)
15. Lichodzijewski, P., Heywood, M.I.: Coevolutionary bid-based genetic programming for problem decomposition in classification. Genetic Programming and Evolvable Machines **9**(4), 331–365 (2008)
16. Liu, L., Shao, L., Li, X., Lu, K.: Learning spatio-temporal representations for action recognition: A genetic programming approach. IEEE Transactions on Cybernetics **46**(1), 158–170 (2015)
17. Liu, Y., Yao, X., Zhao, Q., Higuchi, T.: Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the 2001 Congress on Evolutionary Computation (CEC'2001). vol. 2, pp. 1101–1108. IEEE (2001)

18. McIntyre, A.R., Heywood, M.I.: Cooperative problem decomposition in pareto competitive classifier models of coevolution. In: European Conference on Genetic Programming. pp. 289–300. Springer (2008)
19. Miguel Antonio, L., Coello Coello, C.A.: Use of Cooperative Coevolution for Solving Large Scale Multiobjective Optimization Problems. In: 2013 IEEE Congress on Evolutionary Computation (CEC'2013). pp. 2758–2765. IEEE Press, Cancún, México (20-23 June 2013), iSBN 978-1-4799-0454-9
20. Park, J., Mei, Y., Nguyen, S., Chen, G., Johnston, M., Zhang, M.: Genetic programming based hyper-heuristics for dynamic job shop scheduling: cooperative coevolutionary approaches. In: European Conference on Genetic Programming. pp. 115–132. Springer (2016)
21. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: International Conference on Parallel Problem Solving from Nature. pp. 249–257. Springer (1994)
22. Rodriguez-Coayahuit, L., Morales-Reyes, A., H.J., E.: A comparison among different levels of abstraction in genetic programming. In: 2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). IEEE (Nov 2019)
23. Rodriguez-Coayahuitl, L., Morales-Reyes, A., Escalante, H.J.: Structurally layered representation learning: towards deep learning through genetic programming. In: European Conference on Genetic Programming. pp. 271–288. Springer (2018)
24. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: null. pp. 860–867. IEEE (2005)
25. Schmidt, U., Roth, S.: Shrinkage fields for effective image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2774–2781 (2014)
26. Shao, L., Liu, L., Li, X.: Feature learning for image classification via multiobjective genetic programming. IEEE Transactions on Neural Networks and Learning Systems **25**(7), 1359–1371 (2013)
27. Tran, B., Xue, B., Zhang, M.: Using feature clustering for GP-based feature construction on high-dimensional data. In: European Conference on Genetic Programming. pp. 210–226. Springer (2017)
28. Veeramachaneni, K., Arnaldo, I., Derby, O., OReilly, U.M.: FlexGP. Journal of Grid Computing **13**(3), 391–407 (2015)
29. Veeramachaneni, K., Derby, O., Sherry, D., O'Reilly, U.M.: Learning regression ensembles with genetic programming at scale. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation. pp. 1117–1124 (2013)
30. Yan, R., Shao, L., Liu, L., Liu, Y.: Natural image denoising using evolved local adaptive filters. Signal Processing **103**, 36–44 (2014)
31. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences **178**(15), 2985–2999 (2008)
32. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing **26**(7), 3142–3155 (2017)
33. Zou, X., Bhanu, B.: Human activity classification based on gait energy image and coevolutionary genetic programming. In: 18th International Conference on Pattern Recognition (ICPR'06). vol. 3, pp. 556–559. IEEE (2006)