Amplitude-oriented Mixed-type CGP Classification

Karlo Knezevic
University of Zagreb, Faculty of El. Engineering and Computing
Unska 3, Zagreb, Croatia
karlo.knezevic@fer.hr

Stjepan Picek
KU Leuven, imec-COSIC
Kasteelpark Arenberg 10
3001 Leuven, Belgium
stjepan@computer.org

Julian F. Miller
University of York
York, UK
julian.miller@york.ac.uk

ABSTRACT

Evolutionary algorithms have proved their worth on various optimization problems over the course of years. However, some techniques like genetic programming (GP) and Cartesian genetic programming (CGP) are not restricted only to optimization problems but can be also used in classification tasks. In this paper, we consider mixed-type CGP (MT-CGP) and test it on a number of benchmark binary and multi-class problems. Following that, we introduce a new representation for our algorithm where each node also has an accompanying weight factor called the amplitude. Our results suggest that this version of CGP is more powerful and able to obtain higher accuracies when compared to the mixed-type CGP or the standard CGP. Finally, we introduce the L1 regularization into MT-CGP in order to facilitate even further feature reduction.

CCS CONCEPTS

•**Computing methodologies → Classification and regression trees; Regularization;** *Discrete space search;*

KEYWORDS

Mixed-type CGP, Classification, Binary, Multi-class, Amplitude, L1-regularization

ACM Reference format:
Karlo Knezevic, Stjepan Picek, and Julian F. Miller. 2017. Amplitude-oriented Mixed-type CGP Classification. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082500

1 INTRODUCTION

Evolutionary algorithms (EAs) are well established as a viable choice for solving difficult, real-world problems. Two of the algorithms belonging there are genetic programming (GP) and Cartesian genetic programming (CGP) [11] where their main difference stems from the encoding of solutions. GP encodes its solutions as trees, while CGP uses a more general structure – graphs.

When considering CGP, one can easily see that it has been used on a wide variety of problems like circuit design [3], cryptography [12], and **classification** [2]. Naturally, in the process of applying CGP, researchers also developed a number of variants such as the self-modifying CGP [7], cyclic CGP [17], and **mixed-type CGP** [6].

However, an impression that cannot be overlooked is that CGP (and its variants) up to now received more attention and obtained better results when considering optimization problems than classification problems. Therefore, in order to build confidence in CGP for classification tasks, more experiments must be made. There, we consider scenarios where using CGP would be beneficial when compared with well-known classifiers such as Decision trees or Support Vector Machines. The first conclusion is that CGP could have clear advantage over GP since it can support an arbitrary number of output nodes where each output represents one class, which is a scenario important when dealing with the multi-class classification. Moreover, with techniques like GP or CGP there is also an implicit feature selection since one can expect that GP/CGP will use only a subset of the feature set.

In this paper, we start with the mixed-type CGP algorithm (MT-CGP) that uses functions which dynamically select the data type of their inputs, which in turn determines the output type. In addition, as far as we know, we are the first to introduce the concept of *amplitudes* to the CGP where each node has a parameter that is also evolved. With such a concept we are able to obtain solutions that are much more fine tuned. In GP and CGP there is also an "implicit" feature selection since usually only a subset of features is really used in the end solution. However, here we extend our setting and use the concept of amplitudes in order to obtain fitness function with a regularization expression which enables us to conduct explicit feature selection during the evolution process. With this set of experiments we aim to provide a further insight when it would be beneficial to use MT-CGP in the classification tasks. As it could be expected, different algorithms and settings have varying behavior on the problems we consider but some trends can be observed.

2 RELATED WORK

Völk et al. used a novel representation of CGP in order to classify breast X-rays in order to detect cancer [16]. The same problem is tackled by Ahmad et al. but here the authors used CGP to evolve artificial neural networks [2]. Turner and Miller also evolved CGP encoded artificial neural networks to classify three problems where one of them was the breast cancer dataset [15]. Harding et al. developed a new type of CGP called the mixed-type CGP and tested it on a number of binary classification problems [6]. Their results showed that the algorithm can offer competitive results when compared to a number of classification techniques.

Ahmad et al. also used CGP to evolve artificial neural networks with a goal of classification of heart arrhythmia types [1]. Smith investigated CGP with an implicit context representation in order

GECCO '17 Companion, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3067695.3082500

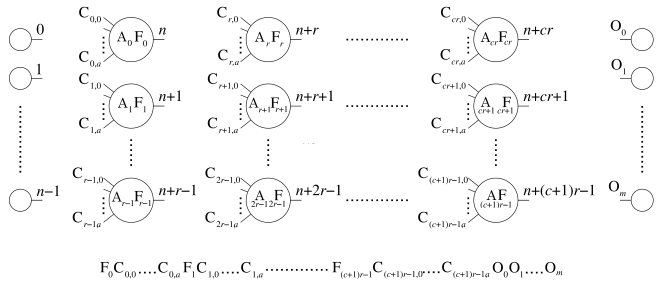

Figure 1: MT-GCP with the node amplitude and encoding.

to recognize Parkinson disease and to detect breast cancer [14]. Stepping aside from the classification on medical data, Leitner et al. used CGP to classify Mars terrain types [9].

3 MIXED-TYPE CGP

3.1 Basic Algorithm and Node Amplitude

Mixed-type CGP can handle multiple data types, in our case, scalars and vectors. Scalars are real numbers with a double precision and vectors consist of scalars. To be able to handle both data types together with a flexibility in the choice of functions, we have encoded scalars and vectors as matrices of dimensions 1×1 and $1 \times n$, where n represents the vector length.

In MT-CGP, even if the functions expect and return matrices, the best suitable mathematical operation still needs to be determined. Consider the "multiplication" function where the function takes two inputs and returns one output. There are four cases that need to be determined: two matrices representing scalar, two matrices representing vectors without having necessarily same dimensions, and a combination of a scalar and vector or vice versa.

Here, we introduce the node amplitude concept where the amplitude can increase or decrease the node function value and better fit to the learning data. It is represented with a single real value assigned to the each node that undergoes mutation in the evolution strategy (ES). Figure 1 shows the amplitude novelty in the functional nodes and bellow the graph we show the equivalent genotype encoding. Next, Algorithm 1 shows MT-CGP graph execution and the amplitude usage. First, set of active nodes is determined, where active node is one that is used in the end solution. The graph output is based on active nodes connected from the graph input to the output. Each active node output is calculated by multiplying value node function and the node amplitude.

A simple $(\mu + \lambda)$ evolution strategy is used to evolve individuals with $\mu = 1$ and $\lambda = 4$. Accordingly, MT-CGP does not use the crossover operator. There are numerous ways of implementing the mutation operator and we have implemented several mutation operators to be used in the tuning phase. In the experiments we use a single active node mutation as the best performing one. This type of mutation was introduced by Goldman et al. [4] where point mutations are applied until an offspring is generated in which an active gene is changed. Apart from often giving better performance, using this mutation operator reduces the number of parameters used in the algorithm.

Algorithm 1 MT-CGP with the amplitude graph execution.

Input: Λ – genotype length, I – input nodes size, N – output nodes size, Ω – genotype
initialize boolean array of active nodes $\alpha[\Lambda]$
initialize array of node values $\omega[\Lambda + I]$
determine active nodes α and set the input data to ω
$\pi = 0$
repeat
  if $\alpha[\pi]$ **then**
    $\phi = \Omega[\pi].connections$
    $\zeta = NodeFunctionIndex[\pi].function$
    $A = \Omega[\pi].amplitude$
    $\omega[\pi + I] = A \cdot Calculate(\omega, \phi, \zeta)$
  end if
  $inc(\pi)$
until $\pi < \Lambda$

3.2 Classifier and Cost Functions

In our implementation each output in MT-CGP represents an output of a classifier function. This means that the individual will have the same number of the output nodes as the dataset classes. The index of the maximal output node represents the input data class as follows:

$$C(x) = max_i\{\sigma(f_0(x)), ..., \sigma(f_c(x))\}, \quad (1)$$

where x represents the input data, $C(x)$ data class, c number of classes, and σ the sigmoid function. The sigmoid function is used to scale all values to the interval $[0, 1]$.

We implemented three cost functions: MT-CGP individual error with a constant node amplitude ($A_{node} = const.$), individual error with a variable amplitude ($A_{node} \in [0, 1]$) and regularization cost function. The regularization cost function is described as sum of the ratio of maximal classifier output and output of i-th output node and a sum of all active nodes amplitudes (L1 regularization). The i index stands for the true class of an example.

$$cost_1 = error,\ A_{node} = \text{const}. \quad (2)$$

$$cost_2 = error,\ A_{node} \in [0, 1]. \quad (3)$$

$$cost_3 = \sum_{1}^{N} \frac{\sigma(f_{max}(x))}{\sigma(f_{true\ class}(x))} + \frac{\lambda}{2} \sum_{i} |w_i|,\ \text{i} \in \text{ active nodes}. \quad (4)$$

The main features of the L1 regularization are sparse outputs and feature selection. Our motivation for this cost function usage is to minimize the classifier model complexity. The comparison of active nodes is shown in Table 3 where we can see that the best solutions from the cost function 3 are the shortest where the main reason is the usage of the L1 regularization.

4 EXPERIMENTS AND SETTING

4.1 Experimental Setting

In Table 1 we present all relevant settings we use in our experiments. We concluded that a larger number of nodes do not improve classifier accuracy and consequently the final testing we conduct with 100 nodes. Apart from giving better classifier generalization,

Table 1: Parameters for MT-CGP.

Parameter	Value
Genotype length	100 nodes
Shortcut connections	disabled
Evolutionary strategy	(1+4)
Mutation type	single active gene
Node amplitude	[0-1] or constant 1.0
Maximum evaluations	500 000
Runs per experiment	30

Table 2: Accuracy vs. genotype length.

Problem \ Nodes	100 (Acc. %)	300 (Acc. %)	500 (Acc. %)
Breast cancer	97.4	**97.9**	**97.9**
Diabetes	72.9	**74.2**	73.2
Heart	81.4	**82.3**	81.7
Phoneme	86.0	**86.9**	85.3
Glass	69.1	72.3	**73.2**

Table 3: Active nodes number vs. cost function.

Problem \ Cost	Cost 1	Cost 2	Cost 3
Breast cancer	25	22	**9**
Diabetes	34	24	**13**
Heart	40	25	**7**
Phoneme	36	25	**13**
Glass	34	22	**17**

a lower number of nodes improves the testing speed, which makes the evolution process much faster. In Table 2 we show how the accuracy changes with the increase of the genotype length. The best solutions are given in bold style.

The implemented function set consists of four functions types: vectorial, mathematical, statistical, and miscellaneous. Detailed function description can be found in [6]. In Table 3 we give the average number of active nodes for all test problems and cost functions.

4.2 Problems

In our investigation, we explore in total five datasets where four of them are binary classification scenarios and one is the multi-class scenario. All datasets are taken from [10]. For each of the classification tasks, the inputs presented to the program are both the vector of attributes representing the object as well as the individual attribute values of the vector. For example, if the data consists of 10 attributes, the MT-CGP program would have 11 input nodes, where the last one is a vector of all data attributes. Evolved programs are then able to select the most appropriate inputs to use in the classifier.

The first dataset we consider is the Breast Cancer dataset. This dataset consists of 660 instances where each instance has 9 attributes. The second dataset is called Diabetes1 which consists of 768 instances and 8 attributes for each instance. Next, we use the Heart1 dataset, which consists of 920 instances where each instance has 35 attributes. This dataset aims to discover heart disease in a patient. Finally, the last binary dataset is Phoneme_CR dataset that

Table 4: Tuned parameters for DT and SVM.

Problem	DT		SVM	
	confidence	# objects	C	γ
Breast cancer	0.10	5	25	0.10
Diabetes	0.20	3	50	0.08
Heart	0.25	2	10	0.01
Phoneme	0.25	4	20	10
Glass	0.25	5	30	0.70

Table 5: MT-CGP results.

Problem	$Cost_1$ (% Acc.)		$Cost_2$ (% Acc.)		$Cost_3$ (% Acc.)	
	Best	Median	Best	Median	Best	Median
Breast cancer	98.50	97.88	**99.13**	98.00	97.50	97.17
Diabetes	74.00	70.90	**79.70**	74.00	75.50	70.60
Heart	**81.70**	77.30	77.80	72.50	80.20	76.00
Phoneme	**85.00**	80.00	82.00	78.00	83.40	81.40
Glass	78.00	72.00	**82.00**	78.00	68.00	63.00

Table 6: Machine learning results.

Classifier	Breast c. % Acc.	Diabetes % Acc.	Heart % Acc.	Phoneme % Acc.	Glass % Acc.
NB	**98.08**	76.04	**80.87**	74.40	50.00
DT	97.31	77.60	75.22	87.00	74.00
SVM	97.69	**78.12**	78.69	**88.4**	**86.00**

consists of 5 000 instances and 5 attributes. Note that all the aforesaid datasets are also explored by Harding et al. when investigating MT-CGP [6]. As an example of a multi-class dataset (more precisely, there are six classes), we use the Glass dataset, which consists of 214 instances where each one has 9 attributes.

4.3 Machine Learning Classifiers

We use three machine learning (ML) classifiers for result comparison: Naive Bayes (NB), C4.5 (DT), and Support Vector Machines (SVM) with a radial kernel function. With the C4.5 algorithm we investigate the influence of the confidence factor parameter that is used for pruning, where smaller values relate to more pruning. We conducted a tuning phase for each dataset. Table 4 shows tuned parameters for Decision Tree and SVM. Confidence factor has influence to pruning and number of objects is minimum number instances per leaf. When one uses radial kernel, factor γ is called precision and C is complexity parameter.

5 RESULTS AND DISCUSSION

Here, besides giving the results for three cost functions as used in MT-CGP, we also give results for three well-known ML classifiers. All the experiments with Naive Bayes, Decision tree, and SVM were done in Weka [5] with a 10-fold cross validation. Table 5 shows best and median results per each dataset we got with MT-CGP and in Table 6 are results obtained by aforementioned classifiers. As it can be seen, for the Breast cancer dataset, MT-CGP outperforms all ML algorithms and the second cost function performs the best. For the Diabetes dataset the results are similar and again, with the second cost function we achieved the best result.

Table 7: Average rankings based on Friedman and post-hoc analysis.

Cost function	Ranking	$p_{Hochberg}$
$Cost_1$	2.09	0.033006
$Cost_2$	**1.18**	–
$Cost_3$	2.73	0.000579

With first cost function, MT-CGP had highest classification accuracy on the Heart dataset. For the Phoneme dataset SVM and DT outperform our MT-CGP with the cost function 1. The results for multi-class classification for the Glass dataset are the best with the SVM, but compared with NB or DT, MT-CGP has better results.

Finally, we conduct a statistical analysis to determine whether there are any statistically significant differences in the performance of three versions of MT-CGP. After running experiments on our datasets, we observe that some of them do not follow the normal distribution nor do they have the same variance. Therefore, we conduct nonparametric statistical analysis [13]. As a measure of efficiency of algorithms, we again use accuracy where we average it over all runs.

Since we have several algorithms and test scenarios, we use a multiple comparison test. The simplest test for multiple comparisons is the Friedman test where the goal of this test is to answer whether there are global differences between related samples obtained. We display average ranks obtained by each algorithm in the Friedman test in Table 7. Since on the basis of the Friedman test we see that the $Cost_2$ function gives the best ranking we use it as a control method in the post-hoc analysis (Hochberg test [8]). In our experiments, we use the level of significance α of 0.05 and we can conclude that the second cost function performs the best. Naturally, all our experiments are done with a relatively low number of measurements so exploring the setting with a higher number of measurements is of high relevance.

6 CONCLUSION

In this paper, we conducted a number of classification tasks with the mixed-type CGP where we explored three cost functions and both binary and multi-class classification. Our results show that MT-CGP should be regarded as a viable option in classification tasks especially when the number of classes is larger than two (i.e., binary classification). We emphasize as a particularly successful setting where we also evolve the amplitude for each node ($Cost_2$ function).

Other settings also offer good results but it could be hard to justify higher computational complexity coming with MT-CGP. Still, one extra advantage over methods like SVM could be the readability of solutions.

ACKNOWLEDGMENTS

This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882.

REFERENCES

[1] Arbab Masood Ahmad, Gul Muhammad Khan, and Sahibzada Ali Mahmud. 2013. Classification of Arrhythmia Types Using Cartesian Genetic Programming Evolved Artificial Neural Networks. In *Proceedings of 14th International Conference on Engineering Applications of Neural Networks (EANN 2013), Part I (Communications in Computer and Information Science)*, Lazaros S. Iliadis, Harris Papadopoulos, and Chrisina Jayne (Eds.), Vol. 383. Springer, Halkidiki, Greece, 282–291.

[2] Arbab Masood Ahmad, Gul Muhammad Khan, Sahibzada Ali Mahmud, and Julian Francis Miller. 2012. Breast Cancer Detection Using Cartesian Genetic Programming Evolved Artificial Neural Networks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 1031–1038.

[3] Z. Gajda and L. Sekanina. 2009. Gate-level optimization of polymorphic circuits using Cartesian Genetic Programming. In *2009 IEEE Congress on Evolutionary Computation*. 1599–1604.

[4] Brian W. Goldman and William F. Punch. 2013. Reducing Wasted Evaluations in Cartesian Genetic Programming. In *Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Şima Etaner-Uyar, and Bin Hu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 61–72.

[5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* 11, 1 (Nov. 2009), 10–18.

[6] Simon Harding, Vincent Graziano, Jürgen Leitner, and Jürgen Schmidhuber. 2012. MT-CGP: Mixed Type Cartesian Genetic Programming. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 751–758.

[7] Simon L. Harding, Julian F. Miller, and Wolfgang Banzhaf. 2011. Self-Modifying Cartesian Genetic Programming. In *Cartesian Genetic Programming*, Julian F. Miller (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 101–124.

[8] Yosef Hochberg and Yoav Benjamini. 1990. More powerful procedures for multiple significance testing. *Statistics in Medicine* 9, 7 (1990), 811–818.

[9] J. Leitner, S. Harding, A. Forster, and J. Schmidhuber. 2012. Mars Terrain Image Classification using Cartesian Genetic Programming. In *11th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. Turin, Italy.

[10] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[11] Julian F Miller and Peter Thomson. 2000. Cartesian genetic programming. In *European Conference on Genetic Programming*. Springer, 121–132.

[12] Stjepan Picek, Domagoj Jakobovic, Julian F. Miller, Lejla Batina, and Marko Cupic. 2016. Cryptographic Boolean functions: One output, many design criteria. *Applied Soft Computing* 40 (2016), 635 – 653.

[13] David J. Sheskin. 2007. *Handbook of Parametric and Nonparametric Statistical Procedures* (4 ed.). Chapman & Hall/CRC.

[14] Stephen L. Smith. 2011. Cartesian Genetic Programming and its Application to Medical Diagnosis. *IEEE Computational Intelligence Magazine* 6, 4 (Nov. 2011), 56–67.

[15] Andrew James Turner and Julian Francis Miller. 2013. Cartesian Genetic Programming Encoded Artificial Neural Networks: A Comparison Using Three Benchmarks. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. ACM, New York, NY, USA, 1005–1012.

[16] Katharina Voelk, Julian F. Miller, and Stephen L. Smith. 2009. Multiple Network CGP for the Classification of Mammograms. In *Applications of Evolutionary Computing, Proceedings*, M Giacobini, A Brabazon, S Cagnoni, GA DiCaro, A Ekart, AI EsparciaAlcazar, M Farooq, A Fink, P Machado, J McCormack, M ONeill, F Neri, M Preuss, F Rothlauf, E Tarantino, and S Yang (Eds.), Vol. 5484 LNCS. Springer-Verlag Berlin, 405–413.

[17] James Alfred Walker, Yang Liu, Gianluca Tempesti, and Andy M. Tyrrell. 2010. Automatic Code Generation on a MOVE Processor Using Cartesian Genetic Programming. In *Evolvable Systems: From Biology to Hardware: 9th International Conference, ICES 2010, York, UK, September 6-8, 2010. Proceedings*, Gianluca Tempesti, Andy M. Tyrrell, and Julian F. Miller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 238–249.