Gene-pool Optimal Mixing in Cartesian Genetic Programming

Harrison, Joe; Alderliesten, Tanja; Bosman, Peter A.N.

DOI
10.1007/978-3-031-14721-0_2

Publication date
2022

Document Version
Final published version

Published in
Parallel Problem Solving from Nature – PPSN XVII - 17th International Conference, PPSN 2022, Proceedings

Gene-pool Optimal Mixing in Cartesian Genetic Programming

Joe Harrison[1,2](✉), Tanja Alderliesten[3], and Peter A. N. Bosman[1,2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{Joe,Peter.Bosman}@cwi.nl
[2] Delft University of Technology, Delft, The Netherlands
[3] Leiden University Medical Center, Leiden, The Netherlands
T.Alderliesten@lumc.nl

Abstract. Genetic Programming (GP) can make an important contribution to explainable artificial intelligence because it can create symbolic expressions as machine learning models. Nevertheless, to be explainable, the expressions must not become too large. This may, however, limit their potential to be accurate. The re-use of subexpressions has the unique potential to mitigate this issue. The Genetic Programming Gene-pool Optimal Mixing Evolutionary Algorithm (GP-GOMEA) is a recent model-based GP approach that has been found particularly capable of evolving small expressions. However, its tree representation offers no explicit mechanisms to re-use subexpressions. By contrast, the graph representation in Cartesian GP (CGP) is natively capable of re-use. For this reason, we introduce CGP-GOMEA, a variant of GP-GOMEA that uses graphs instead of trees. We experimentally compare various configurations of CGP-GOMEA with GP-GOMEA and find that CGP-GOMEA performs on par with GP-GOMEA on three common datasets. Moreover, CGP-GOMEA is found to produce models that re-use subexpressions more often than GP-GOMEA uses duplicate subexpressions. This indicates that CGP-GOMEA has unique added potential, allowing to find even smaller expressions than GP-GOMEA with similar accuracy.

Keywords: Cartesian genetic programming · Gene-pool Optimal Mixing · Subexpression re-use · Evolutionary computation · Symbolic regression

1 Introduction

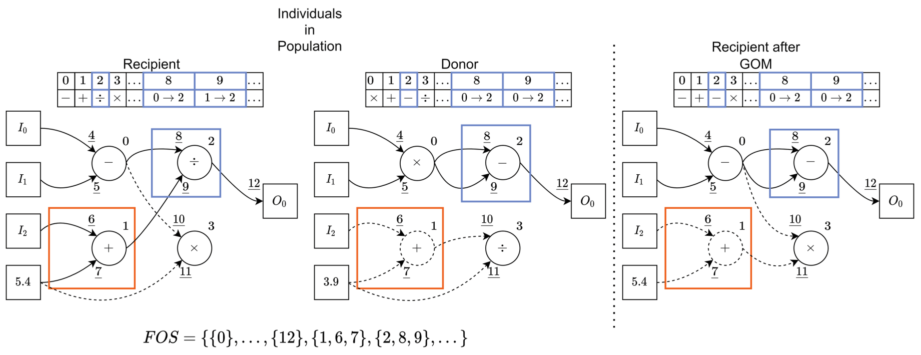
Automated decision-making using Machine Learning (ML) is becoming more prevalent in domains where interpretability is critical such as medicine or law [9]. Unfortunately, many common ML techniques currently used are based on opaque black-box models. Interpretable models are increasingly desired and sometimes even required by law [15].

Symbolic Regression (SR) is the task of finding an expression of a function that fits the samples of a dataset. Typically, SR techniques are used in the hope of obtaining an interpretable expression. Expressions consists of operators, variables, and constants. Genetic Programming (GP) [7] is a popular tree-based technique used for SR. The resulting expressions from the classic version of GP are, however, often too large to comprehend [18], even when its subexpressions are easy to understand by themselves. This is due to a phenomenon called bloat [8]. Generally, the smaller the expression, the higher the likelihood that it will be interpretable. However, smaller expressions may also be less accurate.

A key reason why classic GP results in large expressions, is because it is easier to represent accurate function estimates with larger trees. One way to combat bloat, is to use a fixed-size tree template. However, enforcing a small tree this way makes the search for high quality solutions more difficult, necessitating more sophisticated evolutionary search. One such approach is the Gene-pool Optimal Mixing Evolutionary Algorithm [2] (GOMEA), of which several variants have been developed for different domains, including tree-based GP (GP-GOMEA) [17,18]. GP-GOMEA is particularly adept at finding small expressions while retaining high accuracy. GOMEA attempts to leverage linkage among problem variables to prevent important building blocks from being disrupted during variation while mixing them well. Linkage information can be prespecified if the optimisation problem is sufficiently understood or can be attempted to be learned during evolution by analysing emerging patterns in the population.

It was suggested that GP-GOMEA may benefit from including repeating subexpressions [18]. In GP-trees subexpression re-use only occurs when the same subexpression is evolved multiple times independently. In Cartesian GP (CGP) [10] expressions are represented by an acyclic feedforward graph rather than a tree. This opens up the opportunity for subexpression re-use. The re-use of subexpressions is interesting because it contributes to the decomposability and interpretability of an expression. Subexpression re-use does not directly decrease the expression length, but rather decreases the number of subexpressions that need to be independently understood. In CGP, these subexpressions can be automatically found during the evolutionary process. However, these subexpressions are not considered Automatically Defined Functions (ADFs) [8], but rather Automatic Re-used Outputs (AROs). AROs require the function in its entirety to remain the same whereas ADFs have dummy arguments where different inputs can be instantiated [12,20]. Nevertheless, the two are closely related. Given that the problem is of sufficient complexity, GP can find smaller expressions using ADFs for some problems [8].

CGP has the ability to produce expressions that re-use subexpressions natively without the need to evolve the same subexpression multiple times. Given the observed advantages brought by GOMEA for GP, it is therefore interesting and potentially of added value to see whether CGP can also benefit from an integration with concepts from GOMEA. Vertices in subexpressions that are re-used can possibly benefit from the simultaneous swaps of genes that happen during linkage-based variation in GOMEA as to not disrupt the salient subexpression.



Individuals in Population
Recipient
Donor
Recipient after GOM
I_0
I_1
I_2
5.4
3.9
O_0
$FOS = \{\{0\},\ldots,\{12\},\{1,6,7\},\{2,8,9\},\ldots\}$

Fig. 1. Illustration of how GOM works in CGP-GOMEA. Operator vertices in the 2×2 grid have a problem variable index on the right diagonally above the operator vertex and two underlined problem variable indices to the left representing the location of the incoming vertex. Problem variables in the orange rectangle are an example of variables with high linkage and appear together in the FOS. Problem variables in the blue rectangle are swapped simultaneously from donor to recipient (i.e. clone). Above each graph is a corresponding string representation. Intron vertices and arcs are indicated by dashed lines, and the active graph by filled lines. (Color figure online)

The main contribution of this paper is realising and studying the integration of GOMEA principles in CGP, which we will call CGP-GOMEA[1]. We will compare and contrast CGP-GOMEA with GP-GOMEA and CGP and investigate performance in terms of accuracy, expression length, and subexpression re-use.

2 Methods

Below we outline the relevant details on GOMEA, CGP, and their integration. Special attention is brought to the differences between GP-GOMEA and CGP-GOMEA since these are both GP variants combined with GOMEA. When discussing CGP, the terms vertices and arcs are used, while for GP-trees and GP and CGP in general, the terms nodes and connections are used.

2.1 GOMEA

GOMEA operates on a fixed-length string representation of the problem variables in a genotype. Any mapping from genotype to string can be used as long as the mapping is unique. For instance in GP-GOMEA, nodes in fixed height trees are mapped to a fixed-length string using the pre-order traversal of the tree [17]. Once a mapping is defined, a model describing the linkage between string indices is learned in the form of a Family Of Subsets (FOS), which is a set of subsets of all string indices. Alternatively, the FOS can be provided exogenously.

[1] Code and data can be found at https://github.com/matigekunstintelligentie/CGP-GOMEA.

The FOS in this paper is learned each generation and is a hierarchical cluster tree, called a Linkage Tree (LT), where string indices with strong linkage are grouped together in a hierarchical fashion. We used Normalised Mutual Information (NMI) as a proxy for linkage. NMI is used because it is a measure of mutual dependence among variables (in this case string indices). For indices with strong mutual dependence it might be beneficial if the genetic material associated with these indices, is varied in a joint fashion. The algorithm Unweighted Pair Group Method with Arithmetic mean (UPGMA) [5] is used to build the LT. UPGMA only needs the NMI between pairs of problem variables as input, represented by an NMI matrix, to build an LT. The application of UPGMA results in an FOS of size $2l - 1$, where l is the number of string indices. The subset containing all string indices is removed as to not swap entire individuals. The effective FOS size is $\boldsymbol{2l - 2}$. A randomly initialised population is expected to have no linkage, but due to the NMI matrix being estimated using finite samples some linkage is measured, especially in the case of GP [18]. To combat this, [18] introduced a linear bias correction measured from the initial population such that the NMI matrix is identity at the start of the evolutionary process. This correction is measured once and used throughout the evolutionary process.

Variation in GOMEA happens by means of Gene-pool Optimal Mixing (GOM). Each generation, each individual of the population is first cloned and then undergoes GOM. For each subset in the FOS, a random donor is sampled and then each problem variable instantiation indicated by the subset is copied from the donor to the clone. If the expression of the clone has changed, its fitness is evaluated. If the fitness is equal or better than its original, the change is kept and otherwise it is discarded. The clones replace the entire original population.

2.2 CGP

In CGP, an expression is encoded using a Cartesian grid. Each vertex in the grid has incoming arcs that can potentially come from any preceding column in the grid, making it an acylic feedforward graph. Note that this makes skip connections and vertex re-use possible (see Fig. 1). By limiting to which preceding column in the grid a vertex can connect, the number of subexpression re-uses can be influenced. This parameter is called Levels-Back (LB). A CGP graph consists of four types of vertices:

1. Ephemeral Random Constants (ERCs) - vertices that output a constant value sampled at the start of the evolutionary process.
2. Inputs (I_i) - vertices that return an input feature of a dataset.
3. Outputs (O_j) - vertices that return the output of an expression.
4. Operators - vertices that apply operations to its incoming arcs.

Only operator vertices are part of the CGP grid. For each operator vertex, the number of incoming arcs is equal to the maximum arity of all operators used. Unary operators only use the first input and ignore other inputs. For the remainder of the paper, the maximum input arity of each operator is two (as in [18]). A vertex in the CGP grid can always connect to an input or ERC vertex regardless of what the value of the LB parameter is. The grid size and number of ERC vertices are (manually) determined a priori and highly depend on the problem and desired shape of the resulting expressions. The number of input vertices depends on the number of inputs in the dataset. Note that even though a vertex appears in the grid it might not be connected to an output vertex, see for example the vertex with string index 3 in Fig. 1. The part of the graph consisting of all vertices and arcs that are connected to a particular output vertex will be referred to as the active graph for that output. Other vertices are considered introns. In CGP it is possible to have multiple outputs or recurrent connections, which enables interesting use-cases. However, in this paper, only feedforward graphs are used for the CGP experiments and only problems with a single output are experimented with in order to compare with GP-GOMEA.

In classic CGP variation happens by means of point mutation [11]. An individual is mutated through point mutation of the operators and arcs until the active graph has changed. A notable difference in our implementation is that ERCs are not mutated in order to be able to fairly compare to the GOMEA algorithms. Originally, selection happens in a $1+\lambda$ scheme [10]. However, tournament selection is also common for larger population sizes [11].

2.3 Adapting GOMEA for CGP

In trees, the location of a problem variable explicitly encodes the location of the incoming child nodes and arcs too, whereas this is not the case in feedforward graphs. To adapt GOMEA for CGP, the incoming arcs in the graph must be added as problem variables in addition to the operator problem variables in the grid. Additionally, a string index is needed for the arc from the grid to the output vertex. When an LT is used, the number of problem variables, and consequently the FOS size, required for a template that can accommodate a similar tree as in GP-GOMEA, is larger. The formula for the number of problem variables in CGP used to build the LT is $3rc + 1$ (for maximum arity of two), where r and c are the number of rows and columns in the CGP grid respectively. An important distinction is that the ERCs and input vertices, as opposed to the original GP-GOMEA implementation [19], are not part of the LT FOS because they are encoded at a fixed position in the grid in CGP-GOMEA. This also means that there is no need for converting continuous ERCs to discrete values (bins) as is needed in GP-GOMEA [18]. To mix ERCs in the population, ERCs are added as unary subsets to the FOS after building the LT. Note that this means that the FOS size increases by the number of ERCs used.

Any unique mapping from vertex and arc to problem variable index can be used. Here, a mapping is used where, starting from the nodes in the first column, each vertex is given three problem variable indices, one for the operator and two for the incoming arcs. The mapping used in this paper is illustrated in Fig. 1.

A larger population size positively impacts the accuracy of the NMI estimation [18]. Typically, there are more inputs and ERCs than operators. This means that there are more possible arcs than operators, especially for the output which can connect to any of the grid vertices. This makes the NMI estimate less accurate for the same population size compared to GP-GOMEA because the cardinality of the variables is higher. Hence, for small population sizes, GP-GOMEA is expected to lead to better results. This, together with the larger FOS size, increases the run-time of GOM as it depends on both factors. GOM is the most costly part for both GP- and CGP-GOMEA due to the many fitness evaluations performed inside GOM. One way to make CGP-GOMEA more efficient is by shrinking the FOS size. We here consider two ways to do this: truncate the FOS or trade expressivity for speed by making the grid smaller.

Table 1. Information about the datasets used in the experiments.

Dataset	#Features	#Samples	Variance y
Boston Housing	13	506	84.59
Yacht Hydrodynamics	6	308	229.84
Tower	25	4999	7703.36

3 Experimental Setup

3.1 General Setup

Each experiment is repeated 30 times using a different random seed for each repetition, but equal random seeds across different experiments to create identical dataset splits for each experiment. Significance is tested using the Wilcoxon signed-rank test using the Pratt tie handling procedure [13] with $\alpha = 0.05/\beta$, where β is the Bonferroni correction coefficient [3,18].

Initialisation. In GP-GOMEA, ERC and input nodes are sampled with probabilities $\frac{1}{1+\#\text{inputs}}$ and $\frac{\#\text{inputs}}{1+\#\text{inputs}}$ respectively. ERC nodes, therefore, occur much less often as a terminal node, especially when there are many inputs. In CGP-GOMEA and CGP-Classic, the number of ERCs needs to be defined beforehand. The number of ERCs is set to half the number of terminal locations in a full GP tree. For example, a GP tree of height 4 has 16 terminal nodes, in this case, 8 ERCs are instantiated for CGP. The probability of connecting to an ERC or input vertex in CGP is equal. The values for ERCs are sampled uniformly between the minimum and maximum target value in the training set.

In this paper, we focus on small expressions with a total number of symbols smaller than or equal to 32, a limitation posed on the expression length based on findings by [18]. This corresponds with a GP tree of height 4 and arity of 2 with an additional output node. In GP-GOMEA trees are initialised half-and-half as in [16,18]. For CGP models with a grid with many columns, the full initialisation method [12] often creates large graphs that exceed the 32 node limit. Therefore, only the grow method will be used for all CGP algorithms. Graphs that exceed 32 nodes are penalized in their fitness with a fitness penalty of $10e^6$, severely limiting the chance of selection in the tournament selection of CGP-classic. In GP-GOMEA and CGP-GOMEA, changes due to subset swaps during GOM resulting in a penalty are likely to be discarded.

Operators. The following operators are used: $\{+, -, \div, \times, min, max, exp, pow, log, sqrt, sin, cos, asin, acos\}$. Note that no protected operators are used. This is done to enhance interpretability as protected operators add complexity to each operator. Expressions that return an error on samples in the training set are penalised with a high fitness offset of $10e^6$.

Linear Scaling. To improve performance while keeping an expression small, Linear Scaling (LS) [4,6] is applied to each solution during fitness evaluation unless stated otherwise. LS effectively adds four symbols to each expression. These symbols are however not counted towards the total expression length.

Grid Sizes. For the CGP-GOMEA experiments, four different grid sizes are experimented with: 16×4 (rows $\times$ columns), 8×8, 1×10, and 1×64. The 16×4 grid serves as a comparison to trees of height 4. This grid size is chosen because it is the minimum size that can accommodate any tree of height 4 evolved by GP-GOMEA. An 8×8 grid is used to test what happens if the grid is more flexible in terms of graph depth. A 1×64 grid, which can represent more graph configurations than the 16×4 grid, is tested as a suggestion from literature [11]. A 16×4 grid with $\text{LB} = 1$ is also tried. All other experiments have the LB parameter set equal to the number of columns of their respective grid. The 16×4, 8×8, and 1×64 grids all have an FOS size of 384. A grid of 1×10 is tested because it has the same FOS size as a tree of height 4 in GP-GOMEA. Further, truncation of the FOS of a 16×4 CGP grid is investigated. After shuffling the FOS during GOM, only the first k subsets of the FOS are considered, where k is the truncation value. With a truncation of 61, the same FOS size as a tree of height 4 is reached.

Performance Metrics. The training and test coefficients of determination (R^2) and expression length are reported. The expression length is counted as the total number of nodes used in the active graph including the output node. The mean squared error of the training set is optimised instead of optimising the R^2 directly. In particular, we are interested in the re-use of nodes. GP trees can evolve the same subexpression multiple times, whereas CGP has the native ability to re-use vertices. Subexpressions can have the same semantic outcome while differing syntactically. To test whether CGP-Classic and CGP-GOMEA re-use subexpressions more often than that GP-GOMEA evolves duplicate subexpressions, we therefore count the number of re-uses by comparing the output of each connection in the graph or tree with all other connection outputs, except connections to terminal-nodes. Outputs are generated by using the training set augmented with 1000 samples from a normal distribution as input. The re-use count is incremented when two outputs are within a $10e^{-6}$ range of each other.

Computational Budget. The number of evaluations made in GOM is the most time-consuming part of GOMEA [18]. Since CGP-GOMEA has a larger FOS size, the number of evaluations per generation is also much higher. We have therefore opted for a time-based comparison where each run gets a budget of 5000 s. We empirically found that 5000 s leaves enough time for populations of most sizes to converge. A run is terminated when one or more of the following conditions is met: the run reaches 5000 s, the mean fitness and best fitness are equal, the best fitness remains unchanged for 100 generations, or, the mean fitness remains unchanged for 5 generations.

3.2 Setup Main Experiment

Three commonly used datasets will be used in our main experiment: Boston Housing, Yacht Hydrodynamics, and Tower (see Table 1) [1]. The datasets are split into a training and test set of 75% and 25% of the samples respectively. Two sets of experiments are done. One where only inputs are used as terminal nodes and one where both inputs and ERCs can appear as terminal nodes. These sets of experiments are done because there is a difference in how ERCs are handled between CGP- and GP-GOMEA. GP-GOMEA needs to convert continuous ERCs to discrete problem variables. This is done in GP-GOMEA by binning ERCs into 100 bins, the most successful method from [18].

Due to the relatively large size of populations used in this paper, tournament selection is used with a tournament size of 4 for classic CGP to select the parents of the new population that will be mutated to create offspring. The individual with the best fitness is directly copied into the new population.

3.3 Population Size Study

The grid size influences the population size that is needed to ensure the variety of subexpressions in the initial population is large enough, which is important for the success of GOMEA variants. For the main experiments we chose a fixed population size of 1000 as in [15], but this choice is not necessarily optimal. To show the influence of choosing a population size, we do a study to find the optimal population size for GP-GOMEA, CGP-GOMEA 16×4, and CGP-Classic on the Boston Housing dataset without ERCs under the time constraint of 5000 s. In Table 4 experimental results are reported using the found optimal population sizes for the Boston Housing dataset on the Yacht and Tower dataset.

3.4 Setup Known Ground Truth Experiment

The optimal formula for the three datasets in Table 1 and the required grid size or tree height is unknown. It is equally unknown whether the datasets have a bias for solutions with less or no subexpression re-use. We want to know whether a known expression with multiple re-used subexpressions is more easily found by CGP-GOMEA compared to GP-GOMEA and CGP-Classic. To this end, we devised a synthetic dataset with a specific known expression that re-uses subexpressions: $I_0^4 - I_1^4 + \frac{I_2^4}{I_3^4}$. To search for this expression, we only allow operators $\{+, -, \div, \times\}$ to be used. The 4th powers in the expression can thus only be created by re-using sub-expressions with the $\times$ operator multiple times. The synthetic dataset has 1000 samples each with 4 input variables, each sampled from a normal distribution with $\sigma = 0.25$ and $\mu = \{0, 1, 2, 3\}$ respectively as to generate slightly overlapping yet mostly distinct input samples. The grid size was chosen so that it is possible to evolve the formula exactly. Only GP-GOMEA, CGP-GOMEA 16×4, 8×8, 16×4 LB $= 1$, and CGP-Classic are tested. In this experiment, LS is not used, because finding the formula rather than optimising for accuracy is what matters here. Nor are ERCs used.

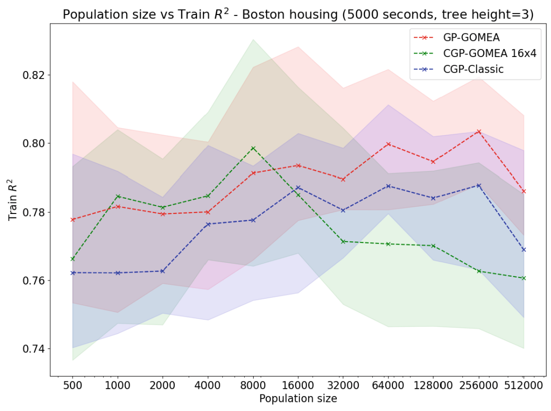


Fig. 2. Results population size study. Shaded area between 10th and 90th percentile.

4 Results

Main Experiment. The results of the main experiment are shown in Table 2. The best training R^2 on the experiments both with and without the use of ERCs is achieved with the CGP-GOMEA 8×8 configuration on Boston Housing and Yacht, and with GP-GOMEA on Tower. The Tower dataset has more variables than available terminal-nodes, which makes it difficult to re-use subexpressions. This is because subexpression re-use means that fewer variables can be used, since re-used subexpressions still count towards the 32 node expression limit. The Yacht Hydrodynamics dataset has a much smaller number of variables and much more re-use is observed for this dataset. A notable difference between the experiments with and without ERCs is that less subexpressions are re-used when ERCs are used, with some CGP configurations even re-using zero subexpressions.

ERCs are used in favour of repeating subexpressions. This could be due to the way ERCs are mixed. In an experiment where ERCs are not added to the FOS and therefore remain unmixed, the re-use of subexpressions was higher for each experiment with similar training R^2.

Truncation, as described earlier, is not a viable method of reducing the FOS size. It consistently ranks among the worst R^2 for all experiments. Trading expressivity for speed is also detrimental to the R^2. A small grid such as 10×1 forces re-use, while as mentioned re-use may not be part of the optimal expression. The configuration from literature, one row with multiple columns, similarly results in low R^2. This is because many individuals in the initial population are penalised for having an expression over 32 nodes, which makes it difficult to create better offspring during GOM without a dedicated constraint handling mechanism in GOMEA, which is currently lacking.

Table 2. Experiment results of various algorithms with and without ERCs as terminal nodes. Median R^2 values are reported due to high variance in test and train R^2. Numbers in bold are best performing for the respective parameter and dataset. Underlined numbers significantly outperform all other algorithms. tr is short for truncation. Test R^2 values filtered from outliers due to unprotected functions are indicated with *.

Algorithm	Without ERCs				With ERCs			
	Median train R^2	Median test R^2	Mean expression length	Mean subexpression re-use	Median train R^2	Median test R^2	Mean expression length	Mean subexpression re-use
Boston Housing								
GP-G	$0.803 \pm 1.71e^{-2}$	$0.761 \pm 5.67e^{-2}$	19.1 ± 3.53	$0.1 \pm 3.00e^{-1}$	$\mathbf{0.83} \pm 1.77e^{-2}$	$0.758 \pm 1.15e^{-1}$	21.6 ± 3.90	$0.1 \pm 3.00e^{-1}$
CGP-G 16×4	$0.81 \pm 2.24e^{-2}$	$0.756 \pm 5.60e^{-2}*$	18.1 ± 4.13	$0.1 \pm 3.00e^{-1}$	$0.806 \pm 2.67e^{-2}$	$0.783 \pm 4.79e^{-2}$	18.3 ± 5.46	0.0
CGP-G tr	$0.768 \pm 2.29e^{-2}$	$0.729 \pm 5.81e^{-2}$	$\underline{\mathbf{11.5}} \pm 4.54$	$0.133 \pm 5.62e^{-1}$	$0.788 \pm 2.23e^{-2}$	$0.733 \pm 5.60e^{-2}$	12.2 ± 3.90	0.0
CGP-G 8×8	$\underline{\mathbf{0.846}} \pm 2.05e^{-2}$	$\mathbf{0.807} \pm 4.53e^{-2}$	27.7 ± 4.64	$0.433 \pm 6.16e^{-1}$	$0.830 \pm 2.94e^{-2}$	$0.787 \pm 7.45e^{-2}$	25.0 ± 6.42	$\mathbf{0.333} \pm 6.50e^{-1}$
CGP-G 1×10	$0.785 \pm 2.29e^{-2}$	$0.743 \pm 4.48e^{-2}$	15.7 ± 5.63	$\mathbf{1.03} \pm 2.12$	$0.772 \pm 2.96e^{-2}$	$0.75 \pm 6.30e^{-2}$	$\mathbf{11.1} \pm 3.90$	$0.1 \pm 3.96e^{-1}$
CGP-G LB$=1$	$0.824 \pm 2.01e^{-2}$	$0.779 \pm 4.43e^{-2}$	21.3 ± 4.08	$0.133 \pm 4.27e^{-1}$	$0.824 \pm 2.25e^{-2}$	$\mathbf{0.789} \pm 5.56e^{-2}*$	19.6 ± 4.38	$0.0333 \pm 1.80e^{-1}$
CGP-G 1×64	$0.807 \pm 2.01e^{-2}$	$0.78 \pm 6.91e^{-2}$	$15.7 \pm 4.58e^{-1}$	$0.133 \pm 3.40e^{-1}$	$0.810 \pm 1.90e^{-2}$	$0.767 \pm 2.08e^{-1}$	$15.4 \pm 9.12e^{-1}$	$0.0333 \pm 1.80e^{-1}$
CGP-C	$0.789 \pm 2.88e^{-2}$	$0.767 \pm 7.35e^{-2}$	16.9 ± 4.94	$0.367 \pm 7.06e^{-1}$	$0.801 \pm 2.26e^{-2}$	$0.762 \pm 4.50e^{-2}$	18.1 ± 4.29	$0.0333 \pm 1.80e^{-1}$
Yacht Hydrodynamics								
GP-G	$0.995 \pm 7.52e^{-4}$	$0.992 \pm 1.71e^{-3}$	17.7 ± 4.30	0.367 ± 1.28	$0.995 \pm 7.59e^{-4}$	$0.994 \pm 1.78e^{-3}$	17.5 ± 4.30	$0.1 \pm 3.00e^{-1}$
CGP-G 16×4	$0.995 \pm 8.75e^{-4}$	$0.994 \pm 2.08e^{-3}$	21.2 ± 6.05	1.2 ± 1.56	$0.995 \pm 7.61e^{-4}$	$0.993 \pm 2.07e^{-3}$	18.5 ± 4.51	$0.3 \pm 7.37e^{-1}$
CGP-G tr	$0.994 \pm 1.02e^{-3}$	$0.992 \pm 2.20e^{-3}$	18.7 ± 5.05	1.53 ± 2.26	$0.995 \pm 9.51e^{-4}$	$0.993 \pm 1.91e^{-3}$	$\mathbf{15.0} \pm 3.81$	$0.167 \pm 5.82e^{-1}$
CGP-G 8×8	$\underline{\mathbf{0.996}} \pm 8.89e^{-4}$	$\mathbf{0.994} \pm 1.62e^{-3}$	28.7 ± 3.38	1.83 ± 2.25	$\underline{\mathbf{0.997}} \pm 8.66e^{-4}$	$\mathbf{0.995} \pm 1.71e^{-3}$	26.6 ± 4.92	0.667 ± 1.07
CGP-G 1×10	$0.994 \pm 6.85e^{-4}$	$0.992 \pm 1.80e^{-3}$	26.4 ± 4.07	$\underline{\mathbf{12.6}} \pm 1.11e^{1}$	$0.995 \pm 6.21e^{-4}$	$0.993 \pm 2.10e^{-3}$	18.2 ± 3.90	$\mathbf{0.733} \pm 1.46$
CGP-G LB$=1$	$0.995 \pm 5.24e^{-4}$	$0.994 \pm 1.81e^{-3}$	25.1 ± 3.72	1.77 ± 1.91	$0.996 \pm 6.38e^{-4}$	$0.994 \pm 2.25e^{-3}$	21.0 ± 4.31	0.467 ± 1.98
CGP-G 1×64	$0.995 \pm 4.92e^{-4}$	$0.994 \pm 1.63e^{-3}$	$\mathbf{15.4} \pm 9.87e^{-1}$	$0.1 \pm 3.00e^{-1}$	$0.995 \pm 5.79e^{-4}$	$0.994 \pm 1.88e^{-3}$	$15.8 \pm 4.96e^{-1}$	$0.1 \pm 3.00e^{-1}$
CGP-C	$0.994 \pm 1.21e^{-3}$	$0.992 \pm 2.19e^{-3}$	22.7 ± 3.96	0.967 ± 1.43	$0.995 \pm 9.49e^{-4}$	$0.994 \pm 2.08e^{-3}$	19.1 ± 3.87	$0.433 \pm 8.44e^{-1}$
Tower								
GP-G	$\underline{\mathbf{0.873}} \pm 8.08e^{-3}$	$\mathbf{0.878} \pm 1.10e^{-2}$	28.2 ± 3.91	$0.0667 \pm 2.49e^{-1}$	$\underline{\mathbf{0.877}} \pm 6.43e^{-3}$	$\underline{\mathbf{0.874}} \pm 1.09e^{-2}$	29.3 ± 2.58	$0.0667 \pm 2.49e^{-1}$
CGP-G 16×4	$0.846 \pm 1.29e^{-2}$	$0.853 \pm 1.52e^{-2}$	17.2 ± 3.89	$0.0667 \pm 2.49e^{-1}$	$0.84 \pm 3.14e^{-2}$	$0.847 \pm 3.26e^{-2}$	16.0 ± 4.01	$0.0333 \pm 1.80e^{-1}$
CGP-G tr	$0.817 \pm 3.78e^{-2}$	$0.821 \pm 3.99e^{-2}$	$\mathbf{13.0} \pm 4.45$	$0.1 \pm 3.00e^{-1}$	$0.764 \pm 4.15e^{-2}$	$0.789 \pm 4.55e^{-2}$	11.2 ± 2.86	0.0
CGP-G 8×8	$0.868 \pm 1.09e^{-2}$	$0.872 \pm 1.67e^{-2}$	22.6 ± 6.28	$0.2 \pm 6.00e^{-1}$	$0.864 \pm 1.50e^{-2}$	$0.866 \pm 1.31e^{-2}$	21.7 ± 6.05	$\mathbf{0.233} \pm 6.16e^{-1}$
CGP-G 1×10	$0.816 \pm 3.34e^{-2}$	$0.827 \pm 3.63e^{-2}$	13.9 ± 3.97	$0.3 \pm 9.00e^{-1}$	$0.769 \pm 3.39e^{-2}$	$0.767 \pm 3.57e^{-2}$	$\mathbf{9.7} \pm 2.79$	0.0
CGP-G LB$=1$	$0.861 \pm 1.81e^{-2}$	$0.861 \pm 1.87e^{-2}$	21.0 ± 3.89	$0.167 \pm 3.73e^{-1}$	$0.85 \pm 2.33e^{-2}$	$0.851 \pm 2.51e^{-2}$	17.0 ± 3.62	$0.0667 \pm 3.59e^{-1}$
CGP-G 1×64	$0.851 \pm 1.46e^{-2}$	$0.845 \pm 1.58e^{-2}$	$15.9 \pm 3.00e^{-1}$	$0.1 \pm 3.00e^{-1}$	$0.847 \pm 1.59e^{-2}$	$0.85 \pm 1.70e^{-2}$	$15.9 \pm 3.40e^{-1}$	$0.0333 \pm 1.80e^{-1}$
CGP-C	$0.844 \pm 2.69e^{-2}$	$0.837 \pm 2.84e^{-2}$	19.3 ± 3.70	$\mathbf{0.333} \pm 5.96e^{-1}$	$0.823 \pm 3.35e^{-2}$	$0.835 \pm 3.38e^{-2}$	16.1 ± 4.75	$0.1 \pm 3.00e^{-1}$

Population Sizing Study. The results of the population sizing study (see Fig. 2) show that all three algorithms initially have a positive trend upwards in terms of R^2 as the population size increases. For CGP-GOMEA 16×4 this trend declines for population sizes above 8000, a smaller population size than observed for the onset of decline in GP-GOMEA and CGP-Classic. This is because although the larger population size positively impacts the quality of the linkage information it also severely limits the number of generations that can be achieved within the maximum time budget, because the run-time of the GOM procedure depends on the FOS size which is larger for CGP-GOMEA.

This exemplifies the importance of using the right parameters in population-based search such as GP. Moreover, what is key to notice from the graph is that the best performance of GP-GOMEA is equal to that of CGP-GOMEA. As these algorithms can represent similar solutions, this was to be expected. However, this search-space-based expectation only holds if the search algorithm is capable of finding high-quality solutions in that space effectively. CGP using classic point-based mutation is not capable of performing equally well. This reconfirms the potential of GOMEA for the GP domain and also confirms that our integration of GOMEA to CGP is essentially successful.

This result also shows that the population size of 1000 used in the main experiment, while congruent with much of literature, can potentially lead to wrong conclusions about the maximum performance of the algorithms tested. Still, the conclusions are valid within the assumed limits. Moreover, the most important comparison, between CGP-GOMEA and GP-GOMEA holds, as from the population sizing experiment we expect these algorithms to perform similarly.

Known Ground Truth Experiment. As mentioned, the expressions found for some datasets have more subexpression re-use than others. If re-use can be found then CGP-GOMEA is a good option. In Table 3 it can be seen that CGP-GOMEA algorithms have a better training R^2 and re-use more subexpressions than GP-GOMEA and CGP-Classic. The CGP-GOMEA 16×4 LB = 1 configuration is the only algorithm that can find the exact expression (twice).

Table 3. Results on synthetic dataset. Numbers marked in bold are best performing for the respective parameter. GP-G, CGP-G and CGP-C are short for GP-GOMEA, CGP-GOMEA and CGP-Classic respectively. The value after ± is the standard deviation.

Algorithm	Median train R^2	Median test R^2	Mean expression length	Times expression found
GP-G	$0.995 \pm 3.33e^{-3}$	$0.995 \pm 4.65e^{-3}$	$0.63 \pm 7.52e^{-1}$	0
CGP-G 16×4	$0.997 \pm 2.88e^{-3}$	$0.997 \pm 3.01e^{-3}$	1.77 ± 1.36	0
CGP-G 8×8	$0.999 \pm 2.15e^{-3}$	$\mathbf{0.999} \pm 2.44e^{-3}$	3.27 ± 2.43	0
CGP-G LB = 1	$\mathbf{0.999} \pm 6.98e^{-4}$	$0.998 \pm 1.30e^{-3}$	$\mathbf{3.87} \pm 3.19$	**2**
CGP-C	$0.998 \pm 4.27e^{-3}$	0.998 ± 4.08^{-3}	2.0 ± 1.37	0

5 Discussion

In CGP-GOMEA a grid size still needs to be defined a priori. A large enough grid could be defined such that it could accommodate any possible tree with 32 nodes, but this would lead to a very large FOS size and subsequently very long run-times. This effectively means that the expressions in CGP-GOMEA are always bounded by a predefined grid size. Potentially a technique akin to NeuroEvolution of Augmenting Topologies (NEAT) [14] could be used to evolve unbounded graphs while still being able to swap homologous blocks using a GOMEA-like approach.

Table 4. Training R^2 of various algorithms trained on Yacht and Tower dataset with population size found in population sizing study for the Boston Housing dataset. Median R^2 values are reported due to the high variance in test and train R^2 values. Numbers marked in bold are best performing for the respective parameter and dataset. Underlined numbers significantly outperform all other algorithms.

	Boston Housing	Yacht Hydrodynamics	Tower
GP-G	**0.803** $\pm$ 1.27e^{-2}	0.994 $\pm$ 5.85e^{-4}	0.769 $\pm$ 1.60e^{-2}
CGP-G	0.791 $\pm$ 2.80e^{-2}	**0.994** $\pm$ 3.93e^{-4}	$\underline{\mathbf{0.844}}$ $\pm$ 2.66e^{-2}
CGP-C	0.788 $\pm$ 1.70e^{-2}	0.993 $\pm$ 9.49e^{-4}	0.780 $\pm$ 1.98e^{-2}

The R^2, expression length, and potentially the number of re-used subexpressions are of interest to optimise. In this paper, however, only the training R^2 is optimised. No pressure is applied on evolving short expressions or expressions with subexpression re-use. Instead, these are just attributes that resulted from single-objective training. As a result, less re-use may have been observed than what is possible. A multi-objective setting may overcome this as well as give more insight into just how much re-use is possible.

Further, a potentially interesting line of research is using the subgraphs with multiple re-uses found by CGP-GOMEA as building blocks for other algorithms. Since these re-uses clearly have value [7,17]. Re-used subexpressions are easily found in CGP graphs without using exogenous processes.

Limitations of this work are the use of only one population size in the main experiment, a restricted number of datasets, and a fixed runtime. Ideally, the optimal population size is used for each configuration and dataset. This would however quickly exceed our computational budget. Of high interest are approaches that adaptively set the population size, increasing resources over time so that an anytime algorithm is obtained. More research is needed to identify in more detail what datasets can benefit from models with native re-use. Finally, only one configuration of CGP-Classic is compared against. While we believe the comparison was fair, showcasing the potential of GOMEA within a basic representation space of CGP, more versions of CGP exist [11] and should be compared against in future work, with similar augmentations on the GOMEA side.

6 Conclusion

In this paper, we showed how GOMEA principles can be applied to CGP and we thereby introduced CGP-GOMEA. We find that CGP-GOMEA with a grid-size of 8×8 strikes a good balance between CGP grid depth and breath and obtains similar training R^2 compared to GP-GOMEA and superior training R^2 compared CGP-Classic on three common datasets while re-using more subexpressions. On a synthetic dataset, where the expression to regress to is known, that has multiple subexpression re-uses, CGP-GOMEA is better able to find expressions that are close to optimal compared to GP-GOMEA and CGP-Classic. We therefore conclude that CGP-GOMEA can successfully leverage the advantageous properties of GOMEA within the CGP representation, enabling re-use integrated within the search procedure, opening up interesting avenues of research.

Acknowledgement. This research is part of the research programme Open Competition Domain Science-KLEIN with project number OCENW.KLEIN.111, which is financed by the Dutch Research Council (NWO). We further thank the Maurits en Anna de Kock Foundation for financing a high-performance computing system. We also thank Marco Virgolin aiding in implementing GP-GOMEA, and Dazhuang Liu and Evi Sijben for their fruitful discussions and reviews.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
2. Bosman, P.A.N., Thierens, D.: On measures to build linkage trees in LTGA. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7491, pp. 276–285. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32937-1_28
3. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
4. Dick, G., Owen, C.A., Whigham, P.A.: Feature standardisation and coefficient optimisation for effective symbolic regression. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 306–314 (2020)
5. Gronau, I., Moran, S.: Optimal implementations of UPGMA and other common clustering algorithms. Inf. Process. Lett. **104**(6), 205–210 (2007)
6. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36599-0_7
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge (1992)

8. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs, vol. 17. MIT Press, Cambridge (1994)
9. Lipton, Z.C.: The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. Queue **16**(3), 31–57 (2018)
10. Miller, J.F., et al.: An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1135–1142 (1999)
11. Miller, J.F.: Cartesian genetic programming: its status and future. Genet. Program Evolvable Mach. **21**, 1–40 (2019). https://doi.org/10.1007/s10710-019-09360-6
12. Poli, R., Banzhaf, W., Langdon, W.B., Miller, J.F., Nordin, P., Fogarty, T.C.: Genetic Programming. Springer (2004)
13. Pratt, J.W.: Remarks on zeros and ties in the Wilcoxon signed rank procedures. J. Am. Stat. Assoc. **54**(287), 655–667 (1959)
14. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
15. Vilone, G., Longo, L.: Explainable artificial intelligence: a systematic review. arXiv preprint arXiv:2006.00093 (2020)
16. Virgolin, M., Alderliesten, T., Bel, A., Witteveen, C., Bosman, P.A.: Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1395–1402 (2018)
17. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.: Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1041–1048 (2017)
18. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.: Improving model-based genetic programming for symbolic regression of small expressions. Evol. Comput. **29**(2), 211–237 (2021)
19. Virgolin, M., De Lorenzo, A., Medvet, E., Randone, F.: Learning a formula of interpretability to learn interpretable formulas. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12270, pp. 79–93. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58115-2_6
20. Woodward, J.R.: Complexity and cartesian genetic programming. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 260–269. Springer, Heidelberg (2006). https://doi.org/10.1007/11729976_23