Evolving Simpler Constructed Features for Clustering Problems with Genetic Programming

Finn Schofield
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
schofifinn@ecs.vuw.ac.nz

Andrew Lensen
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
andrew.lensen@ecs.vuw.ac.nz

***Abstract*—Clustering is a widely used unsupervised learning technique. However, as the size and complexity of data increases, the performance of clustering algorithms diminishes, as well as the interpretability of the clustering partition. Genetic programming has been used to perform feature construction on data to increase clustering performance. However, existing work has not focused on encouraging simpler constructed features. In this paper, existing techniques are further developed to include parsimony pressure—a method to encourage evolution towards simpler solutions. With simpler solutions, the constructed features become easier to understand and interpret. The results of experiments using the proposed method show that parsimony pressure is an effective method for producing significantly simpler constructed features without any reduction on the performance of *k*-means++ clustering. Evolved individuals are also analysed to demonstrate the effect of parsimony pressure on interpretability, showing the power of parsimony pressure for avoiding redundancies in individuals, and thus increasing the interpretability.**

***Index Terms*—Clustering, Genetic Programming, Feature Construction, Parsimony Pressure, Feature Selection, k-means.**

I. Introduction

Clustering is an important unsupervised machine learning technique that can be used to understand the structure of *unlabelled* data by partitioning the instances into separate groups, or *clusters* [1]. A clustering algorithm seeks to find a cluster partition, whereby instances in the same cluster are similar, and instances in different clusters are different. However, as the dimensionality and complexity of data increases, clustering techniques become less effective and efficient, and the results become difficult to interpret [2]. This problem is related to the "curse of dimensionality", where data becomes more and more sparse as the dimensionality increases. [3].

To address the problem of clustering complex data, dimensionality reduction techniques can be used to reduce the number of features (attributes) of the data. Feature manipulation is a powerful method in this regard, and can either work by eliminating redundant features (feature selection), or by constructing new, higher-level constructed features from the original feature set (feature construction (FC)) [4]. Feature manipulation can help to reduce noise in the data, and highlight higher-level interactions between features. For this reason, there is often a desire to be able to interpret the constructed features themselves to better understand the data [5]. When used for clustering, understanding the constructed features can also complement the partition produced by the clustering algorithm. By understanding which features are important to find a good partition and which are not, the meaning of the clusters can become better understood.

Genetic programming (GP) is a technique that has show promising use for FC [6], [7]. GP is an evolutionary computation method that generates solutions in the form of simple computer programs, known as individuals. GP starts with a population of random individuals, which are then refined generation by generation to produce high-quality solutions. As GP individuals are dynamic in size, a common problem encountered is *bloat*. Bloat occurs when individuals continue to grow in size through the evolutionary process without any significant change to their fitness, causing individuals to develop redundant elements [8]. This results in individuals being larger and more complex than is required by the problem.

Various bloat-control techniques have been proposed, such as depth limiting [9] or parsimony pressure [10]. Depth limiting ensures individuals do not grow beyond a certain depth, while parsimony pressure works by penalising individuals in the evolution process based on a a measure of size or complexity.

Existing GP-based FC has placed little focus on the simplicity and interpretability of results, instead emphasising performance. Understanding the constructed features can therefore become difficult—potentially even impossible—with the large size of individuals generated by existing GP-based FC methods. To address this, we will explore incorporating the use of parsimony pressure into existing GP-based FC for clustering methods in this work. We will:

- Propose a revised GP algorithm for FC in clustering that uses parsimony pressure to encourage the evolutionary process towards smaller individuals;
- Evaluate the proposed method against existing GP for FC methods, as well as against a clustering baseline using no feature manipulation; and
- Analyse selected GP individuals to better understand the complexity of individuals when not using parsimony pressure, and the impact the parsimony pressure has when applied.

II. Background

A. Genetic Programming

GP works by starting with a population of randomly generated individuals. A new generation is then selected based on a selection operator which considers their fitness as determined by the fitness function. This new generation has genetic operators applied, producing the next generation. This repeats for a set amount of generations, or until an acceptable individual is found.

The most common representation of a GP individual is a tree-like computer program [9]. This representation is not fixed in size, as its depth can be increased or decreased through genetic operations. As GP traditionally considers only the fitness of an individual, an individual could continually grow without any change in fitness, or be generated unnecessarily large and have no reason to become smaller—resulting in bloat.

Parsimony pressure is a technique that penalises larger programs. In doing so, parsimony pressure drives the evolution process towards smaller solutions. The parsimony pressure algorithm that this work will use is lexicographic parsimony pressure [11]. This pressure works by assigning the individuals in each generation a rank based on their fitness. Then, for the purposes of any comparisons between any individuals in that generation—for example in tournament selection—individuals of differing ranks are compared by rank, while individuals of the same rank are compared by a measure of complexity (i.e. tree depth, total nodes). This allows the complexity to be factored into the selection, penalising selecting complex representations and favouring simpler solutions.

An important component of lexicographic parsimony pressure is the algorithm chosen to assign the fitness rankings. How the rankings are assigned determines how "aggressive" the parsimony pressure is. The least aggressive of these is simply assigning the same rank to all individuals with the exact same fitness. This algorithm leads the parsimony pressure to simply act as a "tie breaker", and only considers the complexity of an individual when comparing two individuals with the same fitness.

There are other ranking methods such as *direct-bucketing* and *ratio-bucketing* which divide the population more aggressively into fewer ranks [11]. Direct-bucketing works by dividing the population into b roughly-equally sized ranks, while ratio-bucketing works by assigning the lowest performing $1/r$ individuals to the lowest rank, then the lowest performing $1/r$ of the remaining population to the second-lowest rank etc. until all individuals have been assigned a rank.

B. Feature Construction

FC is a technique used to reduce the dimensionality of data by selecting and combining features of the original data to create a smaller set of constructed features [4]. This is a highly useful technique, as reducing the dimensionality of data can allow machine learning and Artificial Intelligence (AI) algorithms to perform more efficiently and effectively. Additionally, the lower the dimensionality of the data, the more understandable the results can potentially be.

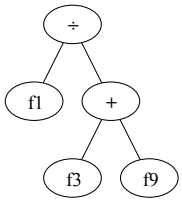

Fig. 1. An example of a GP individual representing a single constructed feature.

Using GP for FC has been explored for both supervised and unsupervised techniques [6], [12]. The tree-based structure of GP makes it a strong candidate for representing constructed features. A constructed feature can be represented as a tree where the leaves are features of the original data and ephemeral random constants (ERCs)[1], while the internal nodes are mathematical and logical functions. From this, we can feed the original data in to the tree and produce a single numerical value representing a feature constructed from the original data. An example of this can be seen in Fig. 1. In this example, the individual can be expressed as $\frac{f1}{(f3+f9)}$ where $f1$, $f3$, and $f9$ are original features of the data. This represents a single constructed feature comprised of three of the original features.

The quality of a given set of constructed features is dependent on the problem that they are being applied to. In the case of FC for clustering, partitions produced using the set of constructed features can be compared with partitions produced using all of the original features. Other ways the quality can be measured include the number of constructed features, or the size of the subset of original features used. Just as reducing the dimensionality of the data can help make the data more understandable, so too can having simpler constructed features.

C. Clustering

Clustering is an important technique for understanding the structure of data. Clustering algorithms work by assigning each instance of a dataset to a cluster. As clustering is an unsupervised learning task, evaluating a clustering solution is more subjective than in supervised contexts, as ground-truth labels are unavailable. Hence, there are many different measures to evaluate clustering performance. These can be divided into two categories: internal and external measures.

Internal measures evaluate a partition by assessing the structure of clusters. They can consider aspects such as similarity within a cluster and/or dissimilarity between clusters. These measures are subjective, and can evaluate a partition according to a number of different criteria.

External measures test a given partition against a ground-truth partition. External measures are an effective way of objectively evaluating a partition (e.g. for benchmarking),

[1]ERCs are values that are randomly initialised when inserted into the tree and then remain constant thereafter.

however they can only be used when ground-truth labels are available.

A plethora of clustering algorithms have been proposed in the literature. In this work, we use the popular clustering algorithm, k-means; specifically the more powerful k-means++ variant [13]. k-means++ is an efficient and simple algorithm for producing cluster partitions. The key difference between k-means and k-means++ is in the initialisation of clusters. In normal k-means, the initial centroids are selected at random. k-means++, however, attempts to select initial centroids that are well-separated. This leads to performance which is at least as good as k-means, and often better. Beyond this, the two algorithms are functionally identical, and operate as follows once the initial centroids have been chosen:

- Each instance of the data is assigned to the nearest centroid according to a distance measure, forming the clusters;
- The mean of each cluster is computed;
- The means of the clusters are set as centroids for the next iteration.

This repeats for a preset amount of iterations or until the centroids stabilise and no longer change between iterations.

D. Related Work

There has been limited work in the area of GP-based FC in unsupervised learning. The research that this paper significantly extends [12] showed promising results in using GP-based FC for clustering. Two representations were proposed: a multi-tree and vector-based approach [12]. In the multi-tree approach, each GP individual is a set of trees, which each represent a constructed feature. In the vector-based approach, each individual is a single tree that uses concatenation and pairwise set operators to produce a single vector of constructed features. The work in [12] showed promising results using the different representations with a few different fitness functions, and will serve as a foundation for the method this work proposes.

While there has presently been no work on reducing the complexity of constructed features in GP-based FC for clustering, there has been research into using bloat control with GP for symbolic regression [14], [15]. Even though this is not directly FC, in symbolic regression FC is performed implicitly by selecting a combination of features and functions to produce a solution. The results in [14] demonstrate that bloat control techniques—including parsimony pressure—can be used to significantly reduce the complexity of GP-based FC individuals while still retaining high-quality solutions.

III. Proposed Method

Existing GP methods for FC in clustering do not consider the interpretability of constructed features in their evolutionary process. This work proposes a revised approach that includes the use of lexicographic parsimony pressure to encourage smaller—and therefore more interpretable—individuals.

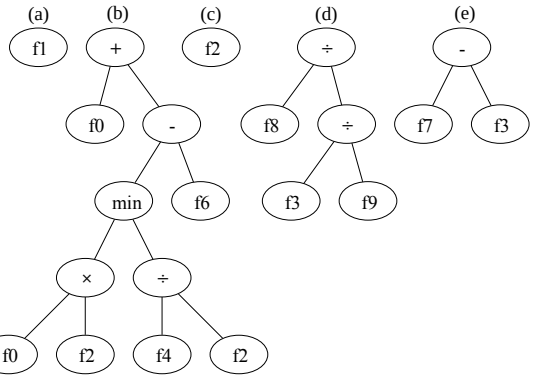

Fig. 2. An example of a set of five constructed features represented as a multi-tree GP individual. Trees (a) and (c) are performing simple feature selection, while (b), (d) and (e) represent higher-level constructed features.

A. Representation

We use the multi-tree representation presented in [12], where each tree corresponds to a single constructed feature. This representation was selected over the vector-tree representation due to its syntax and semantics being more closely related, allowing for more meaningful crossover. An example of the multi-tree representation is provided in Fig. 2. The function set used contains basic arithmetic operators ($+,-,\times,\div$), max and min functions, and an if function. Every operator other than the if function takes two floating point numbers as inputs and produces a single floating point output. The $\div$ is protected division, which protects against division-by-zero errors by simply returning 1 whenever an operation is attempted with 0 as the divisor. The if function takes three floating point numbers as input. If the first input is positive, the second input is returned; otherwise the third input is returned.

Using a multi-tree approach requires special evolutionary operators to be designed. Mutation is done simply by performing standard mutation on a randomly selected tree of the individual. For crossover, as in [12], *random index crossover* (RIC) is used. Using RIC, when two individuals are selected for crossover, a single random tree is selected from each individual, and standard crossover is applied to these two trees. RIC is a less aggressive form of multi-tree crossover compared to others (e.g. crossover between all pairs of trees). This leads to smaller changes across generations, potentially requiring more generations to reach a given level of fitness. However, due to its less destructive nature, RIC can perform finer optimisations between individuals.

B. Parsimony Pressure

The parsimony pressure is introduced to the algorithm through the selection operator. Selection of individuals is done with t-tournament selection using lexicographic parsimony pressure. Before selection, all individuals are assigned a fitness rank. This ranking is assigned simply by fitness, where all individuals with exactly the same fitness are in the same rank. Then, t individuals are selected at random from the current population. From these individuals, only those with the highest

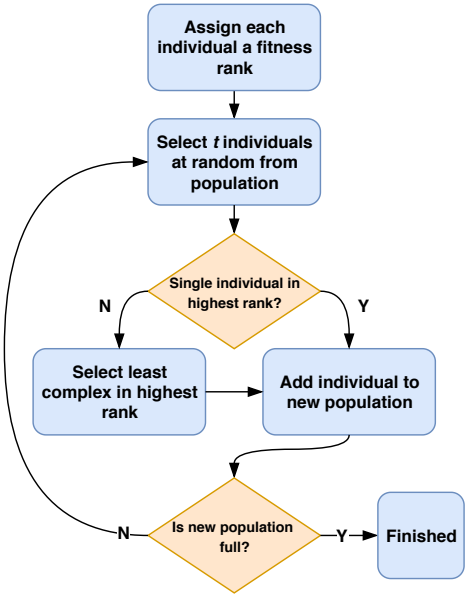

Fig. 3. The selection process with lexicographic parsimony pressure.

ranking are considered. If there is more than one individual of that ranking, the least complex is selected. This is repeated until the new generation is full. A diagram illustrating this process can be seen in Fig. 3

This form of parsimony pressure is an elegant and effective way of introducing penalisation for overly-complex individuals through tie-breaking. A large advantage of lexicographic parsimony pressure is its low computational cost. As individuals' complexity is only considered when tie-breaking between individuals, complexity only needs to be calculated when required, as opposed to computing it for the whole population each generation. In GP, individuals can vary in size vastly, but produce an identical result. Without the lexicographic parsimony pressure, deciding between such individuals is done arbitrarily. It is hoped that this simple change will lead to a significant change in the complexity of the highest-quality individuals produced by the evolutionary process.

C. Fitness Function

The fitness function used is the silhouette score. The silhouette score is an internal measure that accounts for both the similarity of instances within the same cluster, and the dissimilarity between instances in different clusters.This makes it an effective single measure for assessing cluster quality while considering both of these two important criteria.

The silhouette score of a clustering is the mean of the silhouette coefficient of each instance [16]. The silhouette coefficient of each instance is comprised of two functions: $a(i)$, the distance from the instance i to all instances within the same cluster, and $b(i)$, the distance from i to all instances in the nearest other cluster. Given instance i, where C_i is the cluster containing i, and $d(i,j)$ is the distance between instances i and j, $a(i)$ and $b(i)$ are defined as:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{1}$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j) \tag{2}$$

The silhouette coefficient of an instance can then be defined as:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{3}$$

Eq. (3) will produce a value in the range $[-1, 1]$, where 1 is a perfect dense, well-separated clustering, while -1 would indicate a random partition of overlapping clusters. Therefore, this is a measure we seek to *maximise*.

The distance function, $d(i,j)$, is chosen to be Euclidean distance between the instances in the original feature space, as opposed to the constructed feature space. When using the constructed feature space, it was found that the GP tended towards extremely simplistic partitions that yielded perfect silhouette scores, but had little to no relation to the actual ground truth partitions. By using the original feature space, elements of the original structure is preserved, and GP is encouraged towards sensible solutions.

The fitness of an n-tree individual is calculated as follows:

- Each instance of the dataset is evaluated by the individual, producing a new dataset with each instance having n features;
- k-means++ is run with the new dataset, producing a partition;
- The silhouette score of the partition is calculated, giving the fitness of the individual.

IV. Experiment Design

The proposed method was tested on a range of synthetic datasets, using k-means++ as the clustering algorithm. To compare, a method without parsimony pressure [12] was also tested to act as a baseline without any complexity control. This method is functionally identical to the proposed method, except that a normal t-tournament selection is used **without** any parsimony pressure. Additionally, k-means++ using all of the original features is also used as a baseline. Due to GP and k-means++ being non-deterministic, each method will be run 30 times with 30 different random seeds. The mean of each measure is then computed for each method. The GP parameters used for both GP methods for all datasets are presented in Table I. These are standard GP parameters as used in previous work [12].

The complexity of individuals is measured by the total number of nodes. This is a measure we seek to *minimise*, so individuals with a lower value will be favoured. Although this measure is not sensitive to the complexity of individual operations in the individual, it is suitable for favouring smaller individuals where less operations are used in total. Early tests found that this measure was the most effective in reducing complexity across total nodes and unique features. For example, using unique features as the complexity measure had

TABLE I
GP PARAMETER SETTINGS

Parameter	Value
Generations	100
Pop. Size	1024
Initial Pop.	Half-and-half
Mutation Rate	20%
Crossover Rate	80%
Elitism	Top-10
Tournament Size	7
Min. Tree Depth	1
Max. Tree Depth	8

little to no effect on the total nodes compared to GP without parsimony pressure, while the reduction in unique features was not significantly different to when using total nodes as the complexity measure.

One challenge posed by the multi-tree representation is that the number of trees, n, needs to be decided *a priori*, and must be sufficiently high enough to be able to capture the complexity of the data, while also remaining as low as possible to ensure the best reduction of dimensionality. To remedy this, a heuristic is used to determine n based on the parameters of the data. Let D be the number of dimensions of the data, and C be the number of clusters. The heuristic is:

$$n = min\{\frac{D}{r}, \frac{C}{r}\} \tag{4}$$

This heuristic ensures that as the total number clusters in a dataset grows, n can increase to adequately capture the number of features needed to separate the data. By also accounting for the number of original features, the dimensionality of the data will be always be at least reduced by a ratio of r.

For the experiments, r was set as 2, as this was found to provide a good balance in initial tests between performance and the total number of constructed features.

A. Datasets

It is much easier to fairly compare clustering algorithms when ground-truth labels are available. However, it is challenging to source high-dimensional high-cluster real-world datasets with reliable labels. Due to this, our experiments use synthetic data from the GA package HAWKS [17]. HAWKS generates datasets with clusters of dynamic shapes and sizes, evolving datasets towards a predefined silhouette score. Using synthesised data with ground-truth labels allows for clustering methods to be objectively evaluated and compared to each other.

The datasets used in our experiments contain 10, 20, 50, 100, 500, or 1000 dimensions, with 10, 20, or 40 clusters. All datasets contained 500 instances. By using datasets with a large range of dimensionalities, we are able to gain a stronger understanding of the performance of the method on higher dimensional data where feature construction can have a more notable effect. In addition to a range of dimensions, the range of clusters allowed the method to be tested on another measure of data complexity and clustering difficulty.

B. Cluster Quality Evaluation Measures

We evaluated each method's performance on the datasets through a selection of measures. Two measures are used to evaluate clustering performance. We used silhouette score (previously defined in Section III-C) as an internal clustering measure, and as the measure that is optimised by the evolutionary process.

The Adjusted Rand Index (ARI) is used as an external measure, to evaluate methods' performance against ground truth cluster labelling [18]. As ARI is an external measure, it is not a suitable measure to be used during evolution where we want to find a clustering without knowing the labels. Therefore, we use ARI to evaluate the best individual found after the evolutionary process is complete, to compare and contrast the GP methods.

Let m be the total number of instances, C be a cluster partition given by a clustering algorithm, and G be the ground truth partition. First, a contingency table is calculated where each entry m_{ij} is the total number of instances shared by C_i and G_j. Let the sum of each row and column be denoted a_i and b_i, respectively. ARI can then be calculated as:

$$ARI = \frac{\sum_{ij}\binom{m_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{m}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{m}{2}} \tag{5}$$

Eq. (5) calculates the frequency of similarity between the two partitions, adjusting for chance groupings. The value will be in $[0, 1]$, where 1 is a perfect partition with every instance in the correct cluster, and 0 is an *exactly* wrong partition.

C. Complexity Evaluation Measures

Two measures are used to evaluate the complexity of a GP solution. These are the total nodes (the summation of the nodes contained in each tree of the individual) and the unique features (the size of the subset of original features used in the constructed features).

The number of total nodes can serve as a measure of the general size of an individual, and it is hoped that the proposed method will be able to produce significantly smaller individuals with similar clustering performance. The number of unique features indicates the size of the subset of original features is used in the individual. The smaller this subset is, the fewer features are being considered to produce the result. This results in a more simple and interpretable set of constructed features.

V. RESULTS

A. Proposed Method Against k-means++ With All Features

The results in Table II compare the performance of the proposed GP for FC method with parsimony pressure against the performance of k-means++ using all the original features (AF). For each dataset and method, the mean Adjusted Rand Index (ARI) and mean Silhouette Score (SIL) are presented

TABLE II
k-MEANS++ PERFORMANCE ON SYNTHETIC DATA: ALL FEATURES (AF) AGAINST GP WITH PARSIMONY PRESSURE (GP-PP).

		ARI	SIL
10d-10c	AF	0.773	0.398
	GP-PP	0.956+	0.584+
10d-20c	AF	0.812	0.451
	GP-PP	0.981+	0.586+
10d-40c	AF	0.847	0.397
	GP-PP	0.849	0.395
20d-10c	AF	0.932	0.469
	GP-PP	0.999+	0.533+
20d-20c	AF	0.895	0.448
	GP-PP	0.998+	0.541+
20d-40c	AF	0.799	0.305
	GP-PP	0.872+	0.325+
50d-10c	AF	0.732	0.338
	GP-PP	0.984+	0.487+
50d-20c	AF	0.742	0.298
	GP-PP	0.982+	0.455+
50d-40c	AF	0.858	0.224
	GP-PP	0.876+	0.230+
100d-10c	AF	0.825	0.314
	GP-PP	0.988+	0.431+
100d-20c	AF	0.776	0.230
	GP-PP	0.956+	0.303+
100d-40c	AF	0.841	0.136
	GP-PP	0.736−	0.117−
500d-10c	AF	0.867	0.169
	GP-PP	0.956+	0.218+
500d-20c	AF	0.784	0.118
	GP-PP	0.752	0.105−
500d-40c	AF	0.861	0.097
	GP-PP	0.403−	0.014−
1000d-10c	AF	0.894	0.122
	GP-PP	0.915+	0.149+
1000d-20c	AF	0.758	0.079
	GP-PP	0.709	0.068−
1000d-40c	AF	0.821	0.081
	GP-PP	0.308−	-0.003−

TABLE III
k-MEANS++ PERFORMANCE ON SYNTHETIC DATA: GP BASELINE (GP-NOPP) AGAINST GP WITH PARSIMONY PRESSURE (GP-PP)

		ARI	SIL	Nodes	Unq. Fts
10d-10c	GP-NoPP	0.953	0.601	977	10.0
	GP-PP	0.956	0.594	350+	9.80+
10d-20c	GP-NoPP	0.979	0.533	680	9.97
	GP-PP	0.981	0.532	400+	9.80
10d-40c	GP-NoPP	0.851	0.408	700	9.80
	GP-PP	0.849	0.404	350	9.93
20d-10c	GP-NoPP	0.999	0.552	152	13.4
	GP-PP	0.999	0.554	60.8+	11.4
20d-20c	GP-NoPP	0.997	0.493	245	17.0
	GP-PP	0.998	0.477	75.0+	16.0
20d-40c	GP-NoPP	0.856	0.300	532	18.3
	GP-PP	0.872	0.310	280	17.7
50d-10c	GP-NoPP	0.988	0.535	949	41.5
	GP-PP	0.984	0.533	227+	32.0+
50d-20c	GP-NoPP	0.979	0.432	583	43.0
	GP-PP	0.982	0.426	313+	38.1+
100d-10c	GP-NoPP	0.979	0.517	2399	95.7
	GP-PP	0.988+	0.531+	167+	36.5+
100d-20c	GP-NoPP	0.957	0.352	933	76.3
	GP-PP	0.956	0.349	400+	64+
100d-40c	GP-NoPP	0.738	0.145	775	80.1
	GP-PP	0.736	0.144	462+	72+
500d-10c	GP-NoPP	0.950	0.477	761	216
	GP-PP	0.956	0.466	342+	121+
500d-20c	GP-NoPP	0.758	0.243	820	235
	GP-PP	0.752	0.240	563+	169+
500d-40c	GP-NoPP	0.406	0.071	654	208
	GP-PP	0.403	0.070	277+	115+
50d-40c	GP-NoPP	0.874	0.208	615	44.7
	GP-PP	0.876	0.204	293+	41.0+
1000d-10c	GP-NoPP	0.920	0.452	1181	375
	GP-PP	0.915	0.463	406+	143+
1000d-20c	GP-NoPP	0.671	0.234	955	337
	GP-PP	0.709	0.240	634	255
1000d-40c	GP-NoPP	0.294	0.058	608	244
	GP-PP	0.308	0.059	296+	139+

from 30 experiments. Mann-Whitney significance testing was performed with a 95% confidence interval. A "+" indicates the GP-based FC performing significantly better than AF in a measure, while a "−" indicates a significantly worse performance. No symbol indicates no significant difference was found. Both measures should be maximized.

On all datasets except 10d-40c (10 dimensions, 40 clusters), 100d-40c, 500d-20c, 500d-40c, 1000d-20c and 1000d-40c, the proposed method performs significantly better than the k-means++ baseline by both measures. Of these datasets where performance does not improve, ARI performance is significantly lower for 100d-40c and 500d-40c and 1000d,40c. Silhouette performance is significantly worse for 100d-40c, 500d-20c and 500d-40, c1000d-20c and 1000d-40c.

Across twelve of the eighteen datasets tested, there was a significant improvement in clustering performance by both measures, while performance was significantly worse by either measure on only five datasets. This demonstrates the effectiveness of the proposed method at increasing clustering performance through FC in most cases tested. The results show the value of the GP-based FC with parsimony pressure in improving clustering performance.

On all of the 10-cluster datasets, performance is increased, while on the high-dimensional data (100, 500, 1000), high cluster (20, 40) data, performance across both measures was always worse with the exception of 100d-20c. This could potentially be due to the fact that there are too few constructed features to adequately capture the underlying complexity of the data and the amount of ground-truth partitions.

B. Proposed Method Against GP Baseline

The results in Table III compare the baseline GP-based FC for clustering method without parsimony pressure (GP-NoPP) [12] against the proposed method using parsimony pressure (GP-PP). Each GP-PP result is labeled with either a "+" or "−" if is better or worse by a statistically significant margin than GP-NoPP according to a Mann-Whitney test with a 95% confidence interval. No symbol indicates that no significant difference was found. As in Table II, ARI and SIL are measures we seek to maximise, while total nodes (Nodes) and the number of unique features (Unq. Fts) are measures we seek to *minimise*.

The GP method with parsimony pressure produced individuals with significantly higher clustering performance according to both the Silhouette and ARI measures for the 100d-10c

dataset. For all other datasets, there was no significant difference in clustering performance. As the parsimony pressure never causes a performance decrease, the value of the proposed method for reducing complexity with no performance trade off is apparent. This also highlights that the poor performance observed on some datasets by GP-PP against AF in Table II is not a result of the introduction of parsimony pressure, but rather the GP method.

On fourteen of the eighteen datasets, the GP method with parsimony pressure produced individuals with significantly fewer total nodes than the GP baseline. This result highlights the effectiveness of this method at producing smaller, and thus more interpretable, individuals in most cases tested.

For all high dimensional datasets (50d, 100d, 500d, 1000d) except 1000d-20c, GP-PP produce individuals that use significantly fewer unique features than GP-NoPP. This result highlights the effect of the proposed method on reducing unique features where it is most valuable; a set of constructed features with fewer features to consider is intuitively easier to interpret and understand when the original set is large.

Of the datasets with lower dimensionality (10d, 20d), only on one dataset (10d-10c) was there a reduction in the number of unique features. As these datasets already consist of a small set of features, reducing the number of unique features used is more difficult—and also less likely to benefit interpretability.

These results clearly demonstrate the effectiveness of the proposed method at encouraging simpler individuals. The parsimony performance is demonstrated as being very effective in reducing the total number nodes, especially in the 100d-10c dataset where the mean total nodes was 167 compared to 2399 without parsimony pressure. Being able to understand and interpret an individual of only 167 nodes is intuitively a significantly easier task than understanding and interpreting one of 2399. Clustering performance remained similar to the baseline while producing significantly smaller individuals across all datasets.

VI. Evolved Program Analysis

When using GP, it is useful to analyse some individuals produced by the evolutionary process. As two GP methods (with and without parsimony pressure) have been compared, we can analyse individuals produced by each of the methods on the same dataset to understand the difference and effectiveness of the proposed method more clearly.

We will analyse the smallest individuals produced by both GP methods from 30 runs on the 1000d-10c dataset. This was a high-dimensional dataset, on which the GP methods had a significantly better clustering performance than the all-features baseline, and the GP-PP method was able to significantly reduce the complexity of individuals produced.

Fig. 4 shows the smallest individual produced for the 1000d-10c dataset from 30 independent runs using parsimony pressure. This individual represents the following a set of 5 constructed features: $f378, f48, f995, (f891 \times (f945 \div f84)) \times f273$ and $min\{max\{f715, f268\}, f523\}$. Of these, the first three are essentially performing basic feature selection,

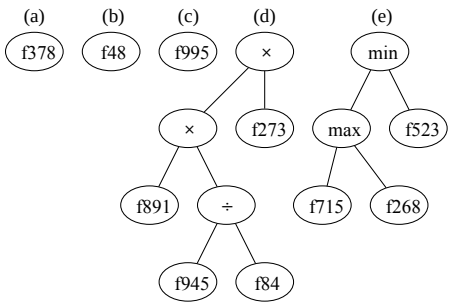

Fig. 4. Smallest individual produced for 1000d-10c dataset across 30 runs with parsimony pressure. In terms of clustering performance, this individual achieved an ARI of 0.964, and a silhouette score of 0.524.

while the last two are higher-level constructed features. This individual demonstrates the flexibility and power of GP-based FC, as simple FS is used when sufficient along with constructed features. In total, this individual is comprised of 15 nodes and ten unique features, compared to a mean of 405 and 142, respectively. This is a good example of a high performing (ARI of 0.964 against 0.894 AF baseline), simple individual on a high-dimensional dataset.

Fig. 4 is a good demonstration of the dimensionality reduction that can be achieved by the GP for FC technique. Using only 1% of the original features, and with some relatively simple mathematical equations, k-means++ performance is increased significantly with a 99.5% reduction in dimensionality. Additionally, this is achieved using less than half the total nodes of the smallest solution produced when not using parsimony pressure. This results in a notably more interpretable set of constructed features.

In contrast, Fig. 5 presents the smallest individual produced by GP without parsimony pressure. In total, this individual has total size of 38—over twice as large as the smallest with parsimony pressure—and uses 11 unique features. Analysing tree (a) of this individual, we can observe some redundancy. The right most sub-tree of (a) corresponds to the equation $min\{f164, min\{f164, f980\}\}$. This is mathematically equivalent to the simpler $min\{f164, f980\}$, resulting in two redundant nodes. With examples of redundancy even in the smaller solution produced for this dataset, the value of introducing parsimony pressure is clear. Additionally, there are original features that are used across multiple different trees. For example, $f205$ is present in both (d) and (e), and $f104$ is present in both (c) and (e). This does not occur in the smallest individual from GP with parsimony pressure in Fig. 4, where each original feature is used only once.

VII. Conclusion

This work has shown that lexicographic parsimony pressure is a very effective method for encouraging simpler constructed features in GP-based FC for k-means++ clustering. The proposed method was able to produce similar results in clustering performance to existing methods, while significantly reducing the total nodes in the individuals and the size of the subset

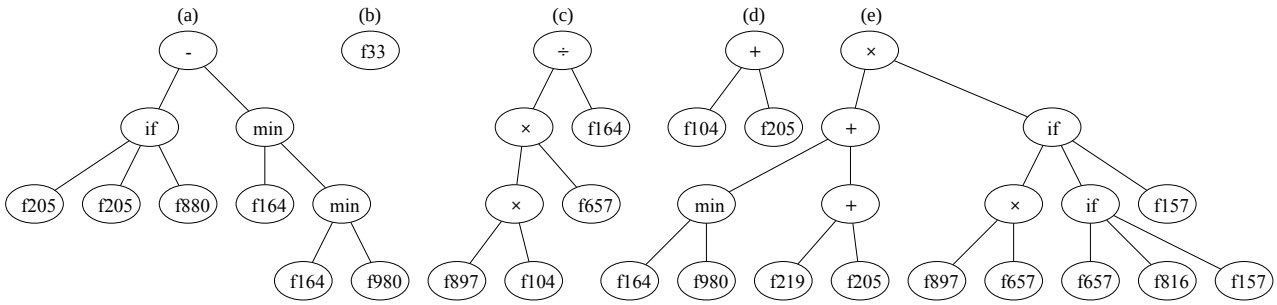

Fig. 5. Smallest individual produced for 1000d-10c dataset across 30 runs *without* parsimony pressure. In terms of clustering performance this individual achieved an ARI of 0.968 and a silhouette of 0.502.

of original features used across most of the synthetic datasets tested.

The method was shown to be very effective at producing more interpretable sets of constructed features that can allow for a deeper understanding of the feature interactions, and the partitions they produce.

Individuals were simpler both in size and in the number of unique original features used. An especially promising finding was the fact that in the vast majority of the higher dimensionality data tested, there was a significant decrease in the unique features used. This is valuable, as the fewer features used, the easier the feature interactions in the constructed feature set are to understand.

Individuals produced with and without parsimony pressure from the same dataset were analyzed, and the effect of the parsimony pressure was highlighted. It was seen that without parsimony pressure, there was redundancy in the GP individuals, even in relatively small examples.

A. Future Work

Further research could investigate more aggressive ranking methods for lexicographic parsimony pressure. It is possible that this could lead to a further complexity reduction than the relatively passive tie-breaking method used in this work. Additionally, alternative forms of parsimony pressure can be investigated, such as parametric parsimony pressure [10]. Further research could explore this method on a range of clustering algorithms, as this work has focused solely on k-means++. Further heuristics for determining the amount of trees per individual could be designed, potentially to address the performance issues with high-dimensional high-cluster data.

REFERENCES

[1] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[2] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, Mar. 2009.

[3] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering," in *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, 1999, pp. 506–517.

[4] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, ser. The Springer International Series in Engineering and Computer Science. Kluwer, 1998, vol. 454.

[5] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," *J. Mach. Learn. Res.*, vol. 5, p. 845–889, Dec. 2004.

[6] A. Bishop, V. Ciesielski, and K. Trist, "Feature construction using genetic programming for classification of images by aesthetic value," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design - Third European Conference, Revised Selected Papers*, 2014, pp. 62–73.

[7] B. Tran, B. Xue, and M. Zhang, "Genetic programming for multiple-feature construction on high-dimensional classification," *Pattern Recognition*, vol. 93, pp. 404–417, 2019.

[8] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*, 2008.

[9] J. Koza, J. Koza, and J. Rice, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, ser. A Bradford book. Bradford, 1992.

[10] R. Poli and N. F. McPhee, "Parsimony pressure made easy: Solving the problem of bloat in GP," in *Theory and Principled Methods for the Design of Metaheuristics*, 2014, pp. 181–204.

[11] S. Luke and L. Panait, "Lexicographic parsimony pressure," in *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO)*, 2002, pp. 829–836.

[12] A. Lensen, B. Xue, and M. Zhang, "New representations in genetic programming for feature construction in k-means clustering," in *Proceedings of the Simulated Evolution and Learning - 11th International Conference, (SEAL)*, 2017, pp. 543–555.

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2007, pp. 1027–1035.

[14] S. Gelly, O. Teytaud, N. Bredèche, and M. Schoenauer, "A statistical learning theory approach of bloat," in *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO)*, 2005, pp. 1783–1784.

[15] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Trans. Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[16] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, p. 53–65, 1987.

[17] C. Shand, R. Allmendinger, J. Handl, A. M. Webb, and J. Keane, "Evolving controllably difficult datasets for clustering," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO*, 2019, pp. 463–471.

[18] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?" in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. Association for Computing Machinery, 2009, p. 1073–1080.