Feature Selection and Classification Using Age Layered Population Structure Genetic Programming

Anthony Awuley
Department of Computer Science
Brock University
St. Catharines, ON, Canada L2S 3A1
Email: anthony.awuley@gmail.com

Brian J. Ross
Department of Computer Science
Brock University
St. Catharines, ON, Canada L2S 3A1
Email: bross@brocku.com

***Abstract*—This paper presents a new algorithm called Feature Selection Age Layered Population Structure (FSALPS) for feature subset selection and classification of varied supervised learning tasks. FSALPS is a modification of Hornby's ALPS algorithm – an evolutionary algorithm renown for avoiding pre-mature convergence on difficult problems. FSALPS uses a novel frequency count system to rank features in the GP population based on evolved feature frequencies. The ranked features are translated into probabilities, which are used to control evolutionary processes such as terminal–symbol selection for the construction of GP trees/sub-trees. The FSALPS meta–heuristic continuously refines the feature subset selection process whiles simultaneously evolving efficient classifiers through a non–converging evolutionary process that favors selection of features with high discrimination of class labels. We compared the performance of canonical GP, ALPS and FSALPS on some high–dimensional benchmark classification datasets, including a hyperspectral vision problem. Although all algorithms had similar classification accuracy, ALPS and FSALPS usually dominated canonical GP in terms of smaller and efficient trees. Furthermore, FSALPS significantly outperformed canonical GP, ALPS, and other feature selection strategies in the literature in its ability to perform dimensionality reduction.**

I. Introduction

Classification is one of the major tasks in machine learning, in which of features or attributes are used to predict class labels [1]. A feature is "an independent measurable property of a process been observed"[2]. Noisy attributes in data increase the complexity of the learning space and reduce performance of learning algorithms with corresponding high computational requirement in data analysis and machine learning. Feature selection is the process of refining input data by removing irrelevant and/or redundant features [3]. Therefore, to improve the performance of classification algorithms, feature selection is often performed. For a data set D having m features, a feature selection algorithm selects a subset of $n \leq m$ features by eliminating irrelevant and redundant features such that it improves or retains the prediction accuracy of the classifier algorithm. According to [4], relevant feature subset selection for learning systems may improve understanding of the data, leads to the design of better classifier models, eliminate irrelevant data with benefits such as enhanced data visualization, reduced computational cost for constructing learning models and improve generalization of constructed models.

Genetic programming (GP) has been widely used for classification [5]. GP evolves an expression that uses a subset of features from the problem feature vectors to predict class label(s). In fact, feature reduction is said to automatically arise when using GP, since most evolved expressions will normally use reduced semi-optimized sets of features selected from the overall feature set. Feature extraction with GP is also possible [5], although we do not consider it in this paper.

Some examples of using GP for feature reduction are as follows. Smith *et al.* [6] used GP and GA for feature extraction and feature selection respectively as a preprocessor for a $C4.5$ decision tree–learning algorithm. The approach proved more successful than direct application of the standard $C4.5$ decision tree–learning algorithm on the same dataset. Research in [7] and [8] used $C4.5$ decision tree learning algorithm as a classifier for a GP feature construction problem. Oechsle *et al.* [9] separate the feature selection and feature extraction stages, and used GP to evolve them independently on a vision problem. The GP classification methodology involved evolution of a set of partial solutions for a single class. This enables the use of GP for classification of high multi–class data, and is faster than conventional GP. Lin *et al.* [10] used a multi–population layered GP to perform feature selection. Badran *et al.* [11] used a multi-objective GP to perform feature extraction as a preprocessing stage to a multi-class problem. This involved the reduction of the original dimensionality space to a new reduced and optimized multi-dimensional decision space. Both Bedran *et al.* [11] and Lemczyk *et al.* [12] used GP to handle a multi-class dataset by addressing the classification problem through task decomposition.

This paper presents a new evolutionary feature selection and classification strategy called Feature Selection Age Layered Population Structure (FSALPS) [13]. FSALPS extends Hornby's ALPS algorithm [14][15] to evolve a classifier and perform feature subset selection with high classification accuracy. The ALPS algorithm overcomes the problem of premature convergence by the regular introduction of new individuals into the population. This results in evolutionary search that is constantly exploring different parts of the search space. We extend ALPS by introducing the ability to evolve feature subsets with high discrimination of class labels. FSALPS uses a novel frequency–based feature–ranking system that will affect the probability of feature assignment to new randomly created individuals. The intention is that the features used will naturally be refined during evolution, which should improve the overall performance of results over time.

FSALPS is tested on high dimensional benchmark datasets and compared to canonical GP, ALPS GP, and others. Combining ALPS search power with GPs dynamic tree encoding to perform feature subset selection should lead to improvements in aspects of classification problems involving high dimensional datasets. We show that FSALPS GP significantly reduces the number of features whiles maintaining or improving classification accuracy. This reduces the computational effort needed to design a classifier system as compared to other feature selection strategies that rely on external classifiers. We also did this without consideration of the effect of bloat (intron) code within the trees, whose existence makes the frequency estimations less accurate than possible.

Section II reviews the ALPS algorithm. Section III introduces the FSALPS algorithm. In Section IV, the performance of canonical GP, ALPS GP and FSALPS are compared on a number of classification problems. A very-high dimension hyperspectral computer vision problem is explored in Section V. In Section VI, FSALPS is compared to related works. Concluding remarks and future research are in Section VII.

II. Review: ALPS

The Age Layered Population Structure (ALPS) algorithm introduced by Hornby[14][15] seeks to reduce the problem of premature convergence in stochastic algorithms. In ALPS, age is used as a property of individuals to restrict competition and breeding in the population. ALPS segregates the population of individuals into age-layers. Age is measured by how long an individual's genotypic material has been evolving in the population. ALPS outperformed canonical EAs in various test problems [14][15].

An aging scheme is used to separate individuals into age layers (see Table I). These values are multiplied by an age gap parameter to determine the maximum age per layer. Given an exponential aging scheme with an age gap of 10 and 6 layers, the maximum ages for the layers will be $10, 20, 40, 80, 160, 320$. Individuals within a layer are not allowed to outgrow the maximum allowed age for that layer. Rather, an attempt is made to move such individuals to the next higher layer. Unsuccessful migrants are destroyed. The maximum age per layer allows evolution to proceed long enough in the preceding layer before individuals are old enough to migrate to the next available layer. It also allows individuals to improve in fitness before being pushed to the next higher layer. The last layer has no age limit, hence allows for the accumulation of the best individuals. According to [14], an individual in the last layer is only guaranteed to remain there provided it is the global optimum or else it will be replaced by other highly fit individuals from the lower layers.

The layered approach of evolution does not only restrict competition between individuals in the entire population but also serves as a way of transferring genotypic materials from different fitness basins to higher layers [14]. In ALPS, the bottom layer is regularly replaced with randomly generated individuals. The periodic introduction of such individuals in the bottom layer results in an EA that is never completely converged [14]. By using age to restrict breeding, the possibility of highly fit old individuals dominating the population is reduced, which most often leads to early convergence in canonical EA.

TABLE I: Examples of age schemes for ALPS [14]

Ageing-scheme	0	1	2	3	4	5	6	8
Linear	1	2	3	4	5	6	7	8
Fibonacci (n)	1	2	3	5	8	13	21	34
Polynomial (n^2)	1	2	4	9	16	25	49	64
Exponential (2^n)	1	2	4	8	16	32	64	128

III. The FSALPS Algorithm

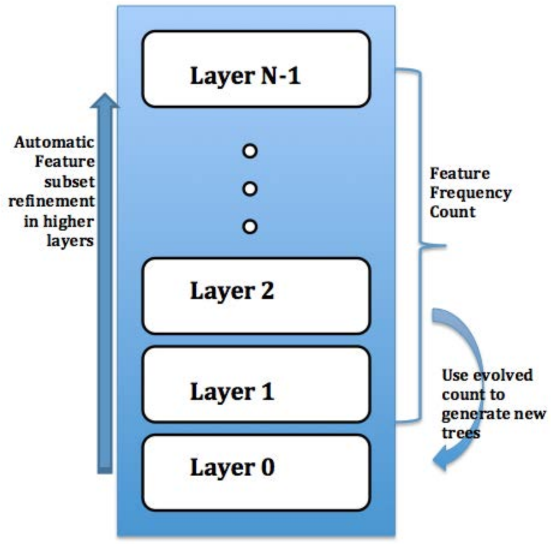

Layer N-1
Layer 2
Layer 1
Layer 0
Automatic Feature subset refinement in higher layers
Feature Frequency Count
Use evolved count to generate new trees

Fig. 1: FSALPS feature selection refinement loop. The frequencies of features in the population are used to determine probabilistic tree generation for layer 0.

FSALPS enhances the ALPS algorithm in Section II, by introducing feature selection mechanisms into the general ALPS strategy. FSALPS performs feature selection by providing a loop back mechanism (see Figure 1) where evolved feature-counts are translated into feature–ranking probabilities. These probabilities are used for selecting terminal symbols in tree construction during initialization or for sub–tree construction during mutation. The FSALPS heuristic performs regular refinement of feature probabilities by reanalyzing the terminal distributions throughout the GP population during a run.

The FSALPS algorithm is shown in Algorithm 1. FSALPS calls which differ from the standard ALPS algorithm are shown in boldface. Lines $2-5$ initialize the FSALPS algorithm. Initially, the system uses a uniform probability for

Algorithm 1 Pseudocode for FSALPS

```
procedure FSALPS_GEN()
    AgeScheme ← SelectAgeingScheme()
    layers ← CreateLayers(AgeScheme)
    i ← SequentialLayerSelection(layers)
    probVector ← **InitialFeatureProbabilities()**
    while not TerminationCondition() do
        if BottomLayer(i) & TooOld(i) then
            probVector ← **ComputeFeatureProbs()**
            j ← **CreateRandomGenome(probVector)**
        else
            if mutation then
                j ← **DoMutation(probVector)**
            else
                if crossover then
                    j ← DoCrossover()
                end if
            end if
        end if
        offspringIndex ← SelectSlotNextGeneration(i)
        j ← CreateChild(offspringIndex)
        EvaluateChild(j)
        TryMoveUp(i,j )
    end while
end procedure
```

Algorithm 2 Pseudocode for Feature Ranking

```
procedure COMPUTEFEATUREPROBS()
    FeatureVector ← $null$
    ProbabilityVector ← $null$
loop:
    for k $in$ Layer(i) do
        for terminal $in$ k do
            FeatureVector[terminal]++;
        end for
    end for
    sum ← Sum(FeatureVector)
    for t $in$ FeatureVector do
        ProbabilityVector ← $t/sum$
    end for
    return ProbabilityVector
end procedure
```

all terminal symbols. Later, when the last age layer becomes active, the uniform terminal probabilities are replaced with computed probabilities, by counting terminals in the trees in the population.

A frequency count of features in a classifier expression is by a direct count of features in the GP tree. Terminal probabilities are directly computed from the frequency counts of n terminals seen throughout the entire population. If feature i has a frequency f_i, then its probability is:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

The frequency count values are then converted into probabilities on line 8 of Algorithm 1 by invoking Algorithm 2. These newly computed probabilities for the features are used in the random creation of new individuals for the first layer when the FSALPS system enters re–initialization mode (line 7). They are also used during tree mutation (line 12) to select terminals during the construction of random sub-trees.

ALPS requires regular inter–layer migration for individuals that are older than the allowed age limit for a layer (line 19). An individual's age is compared to the maximum allowed age of its layer, and if it is older, an attempt is made to move the individual to a higher layer. All layers are scanned for the layer (L_h) with an age range that can accommodate a new individual from a lower layer (L_l). When moving over–aged individuals (I_l) to a higher layer, a *reverse tournament* selection strategy is used to select the weakest individual in a random tournament of size k, which is then replaced by the migrated individual.

IV. Classification Experiments

A. Experimental Design

TABLE II: Dataset summary

Name	Attributes	Examples	Pos	Neg
Pima Indians Diabetes	8	336	111	225
Wisconsin Breast Cancer	30	569	212	357
Ionosphere	34	351	225	126
Sonar	60	208	111	97

1) Datasets: Four datasets with varying number of features and training examples are examined (Table II). The datasets are from the UCI ML database [16]. All use continuous (floating-point) feature values, and all are binary classification problems. The labels Pos and Neg refers to diabetes diagnoses (Pima), tumour malignancy (Wisconsin), good or bad radar prediction (Ionosphere), and metal cylinder or rock detection (Sonar). Incomplete data is removed from all the datasets.

2) Fitness Function: For each feature subset, a classifier is used to test the prediction accuracy of class labels. In a binary classification problem, the fitness of individuals on a test data with P positive instances and N negative instances is translated into metrics such as true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Class labels that were correctly discriminated by the evolved GP classifier make up TP and TN. The sum of TP and FN equals the sum of positive class labels in the test data. We use classification accuracy for fitness measurements:

$$fitness = (TP + TN)/(P + N)$$

Positive and negative examples are not distinguished during sampling, and so their natural distribution in the dataset determines their frequency as fitness cases.

TABLE III: GP Language

Name	Representation	Arity	Definition
Log base 10	log	1	$\begin{cases} log(arg0), & arg0 \geq 0 \\ 1, & arg0 < 0 \end{cases}$
Natural Log	ln	1	$\begin{cases} ln(arg0), & arg0 \geq 0 \\ 1, & arg0 < 0 \end{cases}$
Maximum	max	2	$max(arg0, arg1)$
Minimum	min	2	$min(arg0, arg1)$
Multiplication	*	2	$arg0 * arg1$
Addition	+	2	$arg0 + arg1$
Subtraction	-	2	$arg0 - arg1$
Protected Division	%	2	$\begin{cases} arg1/arg0, & arg0 \neq 0 \\ 1, & arg0 = 0 \end{cases}$
Constants	ERC	0	Ephemerals
Features	$f_0 - f_n$	0	Dataset features

TABLE IV: Canonical GP Parameters

Parameter	Definition
Number of observed Runs	20
Replacement Strategy	Generational
Population size	250
Generations	1000
Selection	Tournament, size = 4
Initial Population	Ramped half-and-half
Grow Minimum	2
Grow Maximum	6
Maximum tree size	17
Crossover	90 %
Mutation	10 %
Termination criteria	100% classification accuracy or end run
k-fold cross validation size	20

Training and testing is done using 20-fold cross validation. Each dataset is shuffled and divided into 20 subsets. We then iteratively use 19 subsets for training, and one subset for testing. We designate the individual with the highest training fitness over a run as being the evolved solution for that run.

3) GP Language: The functions listed in Table III are used. Functions meet closure requirements to avoid illegal operations during tree evaluation. ERCs reside in the range $-1.0 \leq ERC \leq 1.0$.

4) Parameters: The parameters used for canonical GP are in Table IV. See [17] for discussion of basic GP parameters. Each evolutionary run will complete a total of $250,000$ evaluations.

Parameter settings used in the FSALPS and ALPS systems are listed in Table V. Five layers are used in each ALPS system using the polynomial (1, 2, 4, 9 and 16) ageing scheme. When selecting parent(s) for breeding, the selection pressure parameter determines how often a parent is selected from either the current or immediate lower layer. A selection pressure of 0.8 implies 80% of parents are selected from current layer and 20% from the immediate lower layer.

TABLE V: ALPS and FSALPS parameter settings

Parameter	Definition
Number of runs	20
Number of layers	5
Offspring selection pressure	0.8
Population per layer	50
Elite size per layer	3
Ageing scheme	Polynomial (1, 2, 4, 9, 16)
Age gap size	5
Layer replacement	Reverse tournament
FSALPS probability calculation	Normal Frequency

TABLE VI: Testing accuracy (%) over 20 runs.

	Canonical		ALPS		FSALPS	
	Avg	Best	Avg	Best	Avg	Best
Breast cancer	93.12	100.0	92.80	100.0	93.82	100.0
Ionosphere	88.02	100.0	89.71	100.0	90.65	100.0
Pima	73.47	94.12	73.81	88.24	74.03	94.12
Sonar	71.27	100.0	73.68	100.0	74.36	100.0

TABLE VII: Minimum (Min), maximum(Max) and average (Avg) number of unique features in best solution tree for all runs. ERC terminal nodes are included as a single feature category in the tabulations. Best results are in boldface.

Dataset		Canonical	ALPS	FSALPS
Pima: Total = 9	Min	7	8	**3**
	Max	9	9	9
	Avg	8.6	8.9	**6.4**
Breast cancer: Total = 31	Min	12	6	**5**
	Max	26	29	**17**
	Avg	17.85	20.9	**9.3**
Ionosphere: Total = 35	Min	9	13	**7**
	Max	25	31	**20**
	Avg	18.65	23.5	**12.75**
Sonar: Total = 61	Min	16	28	**7**
	Max	42	45	**27**
	Avg	23.65	36.9	**16.85**

5) GP system and Hardware Configuration: All the algorithms tested (Canonical GP, ALPS, FSALPS) are implemented in ECJ [18].

We used a cluster with 9 nodes with 6 cores each, making a total of 54 cores with 2 different hardware configurations:

1) Intel Core i7 920 2.66GHz with 12GB ram
2) AMD Phenom X6 1055 3.6GHz with 8GB ram

The runs were deployed in parallel such that each run was handled by one compute node.

B. Results

1) Classification Accuracy: The prediction accuracy of a classifier is a measure of how the predictor performs on all training examples. The best solution tree in the training phase is tested on a test dataset. Using k–fold cross validation, all instances of the datasets are used for testing. Table VI contains the average testing performance (classification accuracy measured in percentage) for each k–subset of the best-evolved training solution tree. The results recorded for all three strategies were not significantly different when compared at a 95% confidence interval using Tukey's HSD ANOVA test.

2) Feature Analysis: Feature reduction is analyzed for the best solutions in each of the 20 runs. Table VII summarizes feature reduction per dataset, in terms of the average/min/max number of features within best solution trees per experiment. Feature reduction is most evident in the FSALPS strategy.

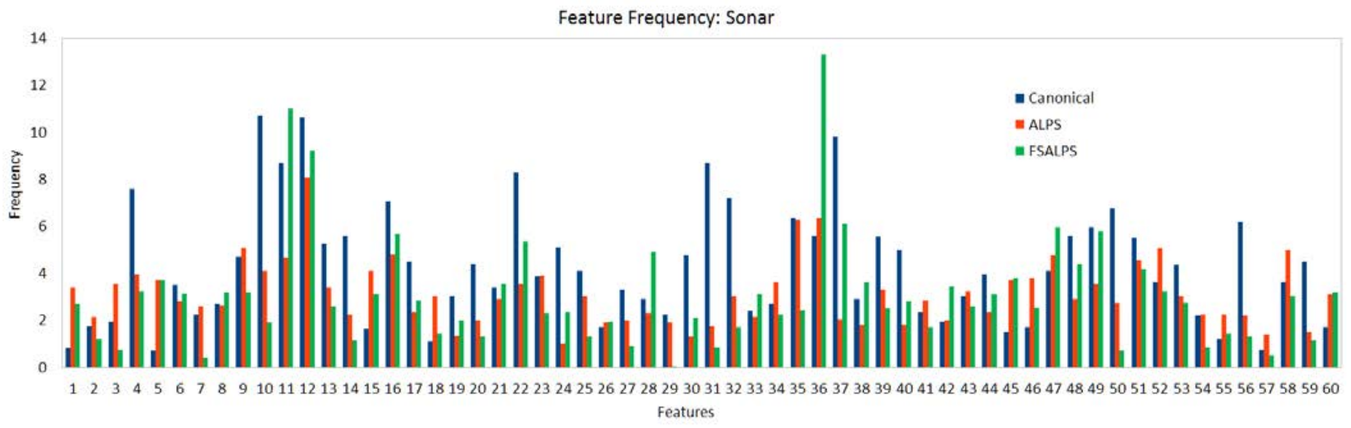


Fig. 2: Sonar: Frequency histogram of features found in solutions (average 20 runs).

TABLE VIII: Pearson correlation coefficients for feature frequency distributions.

	Canonical vs ALPS	Canonical vs FSALPS	ALPS vs FSALPS
Pima	−0.183	−0.254	0.914
Breastcancer	0.545	0.587	0.803
Ionosphere	0.624	0.675	0.786
Sonar	0.333	0.391	0.589

TABLE IX: Comparing mean difference between feature selection performed for each strategy at the 0.05 significance level using Tukey's HSD ANOVA test for Breastcancer(⊗), Ionosphere(⊕), Pima(⊖) and Sonar(⊘) dataset for all 20 runs. An arrow points to the dominant strategy while a — means there is no significant difference between the two strategies.

	ALPS	FSALPS
Canonical	$—^{\otimes\ominus}$ $\longleftarrow^{\oplus\oslash}$	$\uparrow^{\otimes\oplus\ominus\oslash}$
ALPS		$\uparrow^{\otimes\oplus\ominus\oslash}$

TABLE X: Comparing mean difference between best solution tree-size for each strategy at the 0.95 significance level for Breastcancer(⊗), Ionosphere(⊕), Pima(⊖) and Sonar(⊘) dataset for all 20 runs. An arrow points to the dominant strategy while a — means there is no significant difference between the two strategies.

	ALPS	FSALPS
Canonical	$—^{\oplus}$ $\uparrow^{\otimes\ominus\oslash}$	$—^{\oplus}$ $\uparrow^{\otimes\ominus\oslash}$
ALPS		$—^{\otimes\oplus\ominus\oslash}$

Figure 2 show the frequencies of the individual objectives as found within solutions in the Sonar runs, averaged over 20 runs. The frequencies are affected by the overall tree sizes of solutions. Some features were almost completely eliminated for each subsequent run, while others are more commonly found in solutions. This shows that features vary in importance with respect to their value during classification.

We compared the distribution shapes of the objective frequencies by computing Pearson correlation coefficients of the frequencies between the three strategies (Table VIII). ALPS and FSALPS show a high correlation to each other in terms of objective frequencies, while there is less correlation of them with canonical GP. In Table IX, Tukey's HSD ANOVA test shows that FSALPS is the superior algorithm for feature reduction (95% confidence interval).

Figure 3 is a visualization of the feature frequencies in the lowest and highest layers of ALPS and FSALPS. This is measured in one single run of each algorithm.

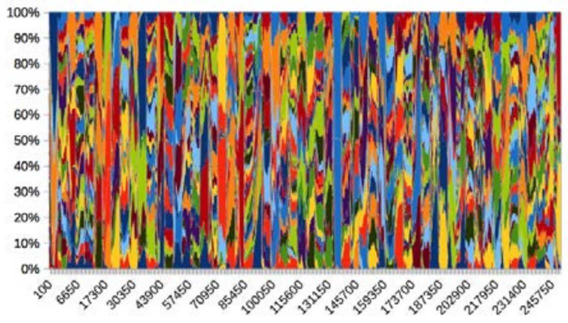
3) Tree Size Analysis: Figure 4 shows the average tree size growth rate per generation for the three strategies, in 2 of the experiments. Tree growth rate is very high for canonical GP compared to ALPS and FSALPS. In Table X, tree size of the best solution tree used for testing was compared at a 95% confidence interval using Tukey's HSD test. The results show a significant size difference between the canonical GP and ALPS variants for all but one dataset.

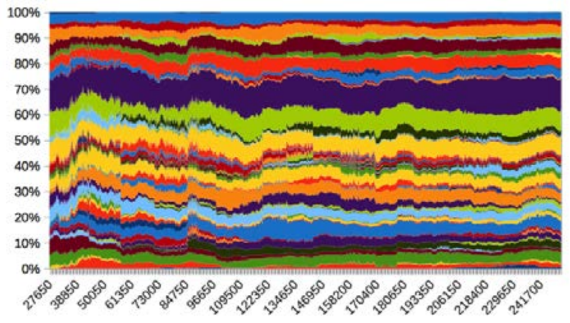
Execution time was significantly lower for ALPS and FSALPS compared to canonical GP. This follows directly from the smaller tree sizes of these algorithms. Details are in [13].

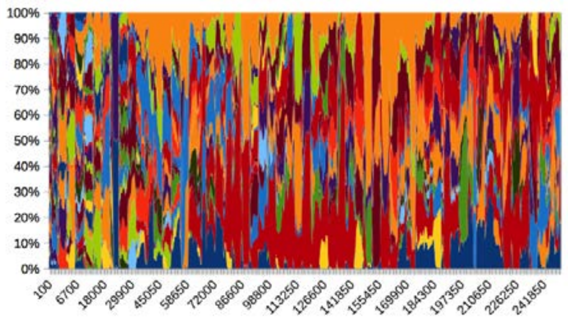
C. Discussion

We were surprised that there is no significant difference in classification accuracy between the 3 strategies. Nevertheless, there is a significant difference in feature reduction. First, canonical GP outperformed ALPS in terms of feature reduction. This is because ALPS continually reintroduces all features into the population at regular intervals, which negatively impacts its feature selection capabilities. FSALPS fixes this issue by selecting a refined set of features to introduce into the population. This resulted in significant reduction of features compared to both canonical GP and ALPS.

ALPS and FSALPS also create smaller trees than canonical GP. This affects the time efficiency for initializing and


(a) ALPS layer 0.


(b) ALPS Layer 4.


(c) FSALPS layer 0.

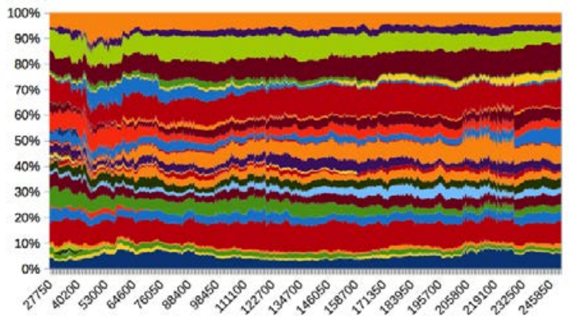
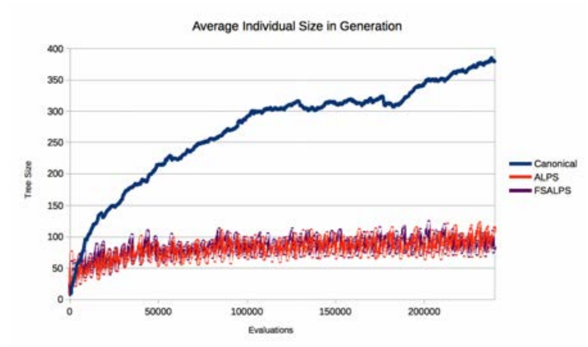
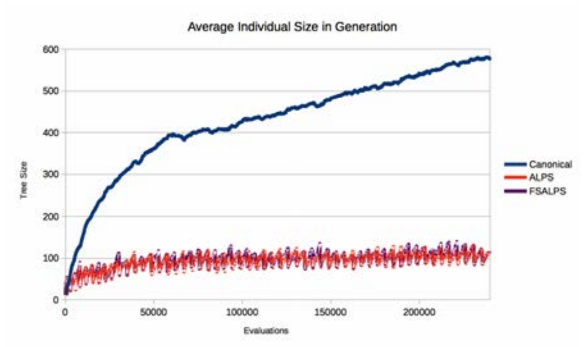

(d) FSALPS Layer 4.

Fig. 3: Sonar experiments: Percentage contribution of each feature in layers 0 and 4 in example ALPS and FSALPS runs. Each of the 60 features is uniquely colour-coded. In (a), ALPS layer 0 shows the regular random introduction of all features through the run, while (c) shows that FSALPS introduces a reduced set of features into level 0. Layer 4 in both algorithms is more stable, although the FSALPS case has a reduced feature set (i.e. fewer colour bands).



(a) Ionosphere



(b) Sonar

Fig. 4: Average population tree size per generation (avg 20 runs).

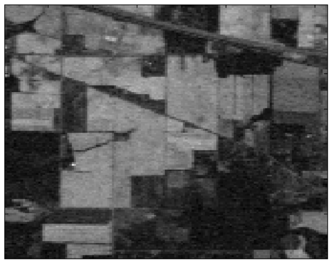
evaluating trees, which results in faster evolutionary runs and classification times in general.

V. Experiment: Hyperspectral Image Classification

A. Experimental Setup

TABLE XI: Number of samples used per training class

Class	Positive samples	Negative samples
Corn–notill	73	102
Soybean–mintill	60	150


(a) Sample band (2.09μm).

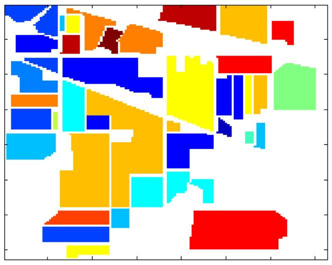

(b) Ground truth of 16 classes.

Fig. 5: A sample spectral band from the Indian pins hyperspectral data and its ground truth image

Hyperspectral imaging allows for simultaneous collection of image data in hundreds of narrow adjacent spectral bands [19] and is achieved using hyperspectral remote sensors, which are usually airborne on planes or satellites. Hyperspectral images are ideal for classifying various physical phenomena such as vegetation or mineral resources [19]. This is possible because organic and inorganic materials produce unique absorption and reflection properties at particular bandwidths. GP has been used to evolve hyperspectral classifiers for a number of applications [20] [21] [22] [23] [24].

We will use GP to evolve classifiers for detecting specific class labels (crops) on the Indian pines hyperspectral data, which was obtained from [25]. The images consist of sets of 145 by 145 pixel data obtained over the Indian Pines test site in northwestern Indiana using the AVIRIS sensor. The site predominantly consists of agriculture and other vegetation. We use 200 image bands with a wavelength range of $0.4 - 2.5\mu$ meters. Since each band is considered a feature, this is a very high dimensional problem to test the capabilities of FSALPS.

Fig 5a shows the reflectance image of the Indian Pines data at the 2.09μm bandwidth. The ground truth image contains a total of 16 classes with known samples (see Fig 5b). Here, we will evolve classifiers for two cases: Corn–notill and Soybean–mintill. Positive and negative examples are obtained from image regions according to the ground truth image in Figure 5b. For each image band, we normalize the RGB color channel between 0.0–1.0. The training examples were manually selected (Table XI). We selected pixels such that a wide range of points of the ground truth image was sampled. At the same time, we did not examine the hyperspectral data during sampling. Hence a similar quality of training examples would likely have arisen if random sampling were performed. 95% of training examples are from regions with 100% intensity of observable reflectance of class labels. The remaining 5% are boundary regions with reduced intensity of resident spectra. The Corn–notill class contains 5.26% of the total samples available on each hyperspectral band while Soybean–mintill covers 8.9% of each band.

As before, we will compare canonical GP, ALPS and FSALPS classifiers on this data. We use the same experimental setup used earlier (Tables III, IV V). The 200 bands are represented as 200 features (terminals), and each returns the band value at that pixel being classified. An image pixel is processed during one tree evaluation, and a positive or negative classification decision for that pixel is determined from the classifier expression.

TABLE XII: Average and best classification testing accuracy (%) for Indian Pines data.

		Canonical	ALPS	FSALPS
Corn–notill	Avg	87.07	88.02	86.64
	Best	92.52	93.69	92.16
Soybean–mintill	Avg	88.63	89.84	89.66
	Best	91.30	92.43	91.40

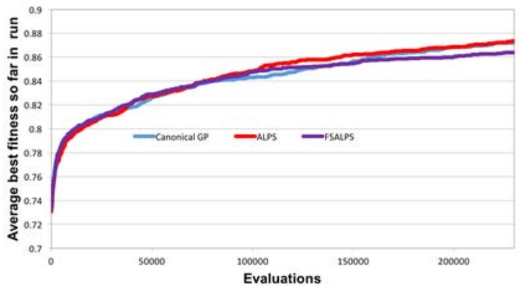
TABLE XIII: Minimum (Min), maximum(Max) , average (Avg) and Standard Deviation (StD) number of unique objectives found in the solution set for each dataset. Total objectives considered for each dataset including ERC is 201.

		Canonical	ALPS	FSALPS
Corn–notill	Min	18	28	8
	Max	47	81	31
	Avg	30.15	51.85	18.25
	StD	7.62	11.95	6.15
Soybean–mintill	Min	17	29	8
	Max	49	78	40
	Avg	30.6	49.55	20.55
	StD	8.26	13.31	8.46

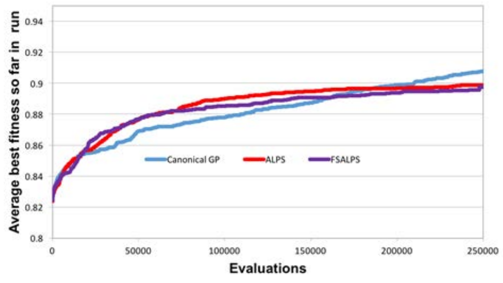
B. Results

1) Classification Accuracy: We found that the 3 strategies did not show signs of premature convergence (Figure 6). The best–evolved classifier obtained from training is applied to the test data. Classification performance of the best classifiers are shown in Figure 7. The raw percentage scores of classification accuracy are recorded in Table XII. All three strategies performed well on both cases. An ANOVA test using (95% confidence interval) did not reveal any statistically significant differences.

2) Feature reduction: Table XIII highlights the feature reduction seen. FSALPS reduced the features the most, and ALPS the least. We used an ANOVA test at 95% significance interval to compare feature reduction between all 20 runs,

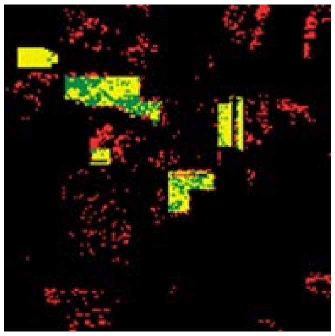


(a) Corn–notill

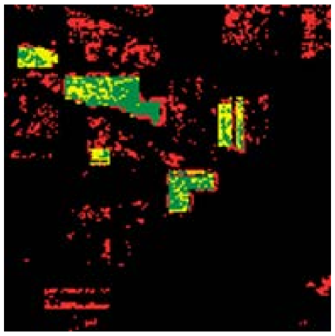


(b) Soybean–mintill

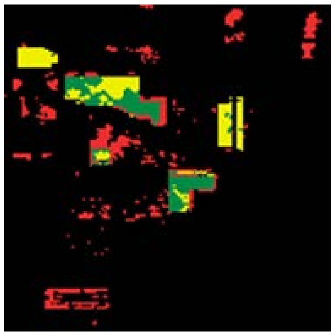
Fig. 6: Training performance plots (avg 20 runs)



(a) Canonical GP: 87.67% (32.73% TP, 95.51% TN)



(b) ALPS: 87.53% (59.95% TP, 92.76% TN)



(c) FSALPS:87.57% (48.01% TP, 95.44% TN)

Fig. 7: Classification accuracy of best classifiers for Corn-notill: TP - green, TN - black, FP - red, FN - yellow.

TABLE XIV: Comparing mean difference between feature selection performed for each strategy at the 0.05 significance level for Corn–notill($\otimes$) and Soybean–mintill ($\oplus$)

	ALPS	FSALPS
Canonical	$\leftarrow^{\otimes\oplus}$	$\uparrow^{\otimes\oplus}$
ALPS		$\uparrow^{\otimes\oplus}$

for all algorithms (see Table XIV). FSALPS significantly outperformed canonical and ALPS for both crop types on feature reduction. Figure 8 compares the frequency of features in layer 0 for ALPS and FSALPS for two hyperspectral runs.

3) Tree Size: Canonical GP has the highest rate of tree growth, while ALPS and FSALPS grew at the same rate (see Figure 9). An ANOVA test (95% significance) confirmed that the tree sizes of ALPS and FSALPS are significantly smaller than canonical GP.

C. Discussion

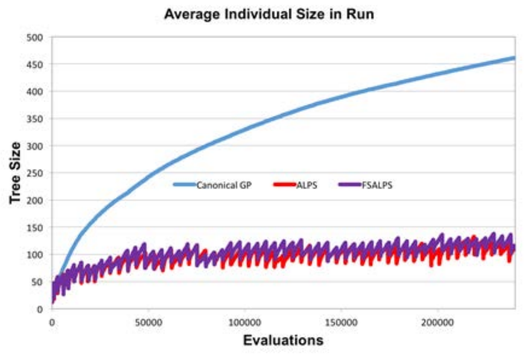
The results for the hyperspectral experiments follow the earlier findings in Section IV. Classification accuracy was equivalent between canonical GP, ALPS, and FSALPS. FSALPS was the best in feature reduction, and FSALPS and ALPS both generated smaller trees than canonical GP.

TABLE XV: Comparing classifier accuracy of FSALPS with others in the literature.

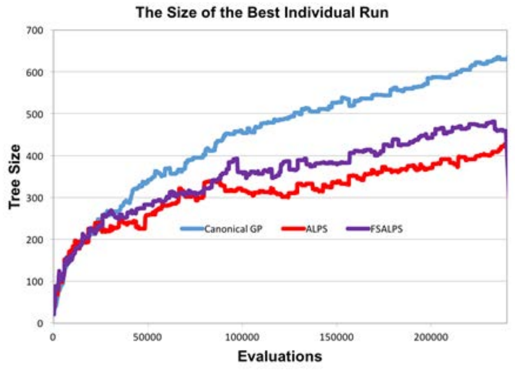
	Pima	WBC New	Ionosphere	Sonar
CHCGA+SVM[2]	80.47	–	94.27	–
CHCGA+RBF[2]	76.82	–	94.29	–
GAP (avg)[26]	75.72	96.14	89.90	85.57
GAP (best)[26]	79.62	98.86	96.17	96.42
Simple Meta[26]	76.04	95.09	89.82	86.64
C4.5[26]	67.94	–	–	69.69
HIDER[26]	74.10	–	–	56.93
XCS[26]	68.62	–	–	53.41
O.F.A[26]	69.80	–	–	79.96
LVSM[26]	78.12	–	–	–
Krawiec[26]	76.41	–	–	–
GAP (J48)[6]	73.64	95.71	90.69	75.89
GAP (IBK)[6]	68.96	94.82	91.38	83.72
GAP (NB)[6]	75.77	96.75	90.60	77.64
C4.5(J48)[6]	73.32	93.88	89.82	73.86
IBK[6]	69.90	95.44	86.95	86.65
N.B.[6]	75.13	93.26	82.37	67.16
FSALPS(avg)	74.03	93.82	90.65	74.36
FSALPS(best)	94.12	100.0	100.0	100.0

VI. Comparisons to related work

Harper [27] used a variation of ALPS called spatial co–evolution ALPS (SCALPS). SCALP was found to produce

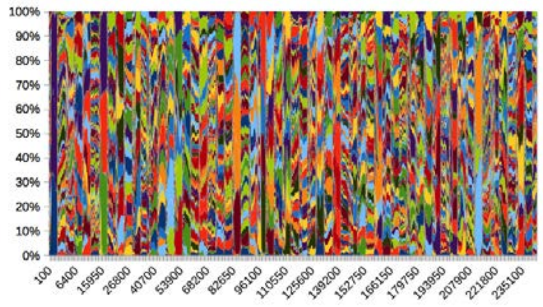


(a) Average individual size.

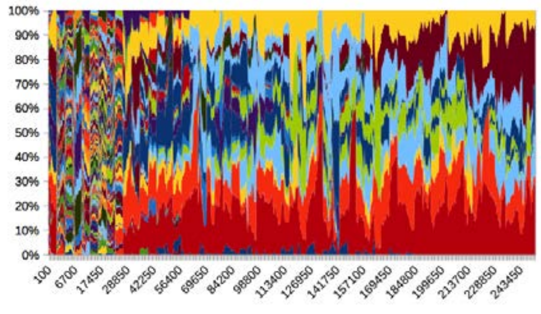


(b) Best individual size.

Fig. 9: Tree growth: Corn–notill. (avg 20 runs)



(a) ALPS Layer 0.



(b) FSALPS Layer 0.

Fig. 8: Percentage contribution of features in Layer 0 of (a) FSALPS and (b) ALPS for two hyperspectral runs. FSALPS is noticeably more stable than ALPS.

smaller and simpler solution trees compared to canonical GP, which is consistent with our results.

Pedro *et al.* [5] identify computational cost and bloat as two main pitfalls associated with the use of GP-based classifiers. The ability of FSALPS and ALPS to produce smaller and more efficient solution trees mitigates these perceived shortcomings.

Table XV compares FSALPS with others in the literature on datasets we used. FSALPS is competitive with the other results in terms of classification accuracy. Our "best" classifiers showed superior performance to others in the table. However, further work using similar testing procedures is needed to confirm these results, and especially using similar experimental methodologies.

TABLE XVI: Comparing feature selection of some published strategies and FSALPS.

Research		Pima	WBC New	Ionosphere	Sonar
CHCGA + SVM [2]	Min	7	–	16	–
CHCGA + RBF [2]	Min	7	–	16	–
[26]	Min	–	–	–	–
	Avg	**6.4**	15.7	19.7	38.0
FSALPS	Min	**3**	**5**	**7**	**7**
	Avg	**6.4**	**9.3**	**12.75**	**16.85**
	Tot:	8	30	34	60

Our primary goal in the hyperspectral experiments is to study a very high-dimensional classification problem, rather than focus on the quality of the hyperspectral classifiers themselves. Preliminary investigation shows that our classifiers are competitive with those in the literature [13]. However, more work is required for accurate comparisons to be made.

VII. Conclusions and Future Work

FSALPS is a new heuristic for GP that performs dynamic feature reduction during evolution. FSALPS emerged dominant over canonical GP and ALPS in terms of feature reduction capabilities. Surprisingly, on the problems investigated here, FSALPS solutions did not score significantly higher classification accuracy than canonical and ALPS GP contrary to the usual expectations that simpler search space means more accurate classifiers. Canonical GP and ALPS are able to compensate and sacrifice feature selection for classification accuracy. FSALPS was also shown to be competitive with other approaches in the literature, in terms of classification accuracy, while often superior in feature reduction capabilities. Therefore, FSALPS is worthy of consideration in classification problems, given that it creates accurate classifiers that are efficient to execute.

There are many directions for future work. First, the feature analysis done by FSALPS could be improved. Because FSALPS uses the entire tree for frequency calculations, bloat (intron) sub-expressions will contribute to inaccurate feature counts. Although bloat will affect our frequencies in our experiments, the results show that feature reduction still arose. Since FSALPS trees are smaller, bloat is probably not as prevalent as with standard GP. More effective feature probabilities could be computed if bloat expressions could be reduced, perhaps via size-fair crossover [28], expression simplification, or linear GP. Since classification accuracy was not affected in our results, another approach might be to take a classifier evolved from standard GP and perform source-code simplification on it. On the other hand, FSALPS performs significant bloat reduction automatically, with minimal overhead.

We used classification accuracy to measure fitness of GP individuals. Additional criteria could be used, such as sensitivity, specificity, information gain, maximum relevance, feature correlation and mutual information [2]. The use of more sophisticated fitness evaluation may result in improved classifier performance and feature reduction.

Other than Figure 2, we did not examine in detail the specific value of particular features in different problems. Future work should investigate this further, especially with respect to information gain measurements.

Feature extraction [2][5] could be incorporated into the FSALPS algorithm. There are situations where the original feature vectors are not representative of the given problem and will require the discovery of interesting hidden relationships between the features. Thus it is possible to exploit GPs representational power to construct a system that performs feature selection and feature extraction as done in [10]. We believe that an FSALPS-like system will facilitate the discovery of new relevant features leading to improved classification accuracy and a more refined feature subset.

In our work on the Indians Pines hyperspectral data, we used a very basic image classification language. Improved results would arise when spatial operators and other spectral operators [24] are added to the GP language. These additions could easily facilitate detection of more difficult regions (e.g. boundaries and edges) on the hyperspectral image.

The problems investigated here involve binary classifications. A new direction would be to investigate multi–classification problems using FSALPS. For example, we could evolve a classifier that simultaneously discriminates multiple classes on hyperspectral images.

Acknowledgements: Thanks to Cale Fairchild for his help with system issues. Research supported by NSERC DG 138467.

References

[1] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms." *IEEE Trans. Evolutionary Computation*, vol. 4, no. 2, pp. 164–171, 2000.

[2] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.

[3] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. Chapman & Hall/CRC, 2007.

[4] W. B. Powell, *Approximate Dynamic Programming - Solving the Curses of Dimensionality.* Wiley, 2007.

[5] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Trans. Sys. Man Cyber Part C*, vol. 40, no. 2, pp. 121–144, Mar. 2010.

[6] M. G. Smith and L. Bull, "Using genetic programming for feature creation with a genetic algorithm feature selector." in *Proc. PPSN VIII*, ser. LNCS, vol. 3242. Springer, 2004, pp. 1163–1171.

[7] F. E. B. Otero, M. M. S. Silva, A. A. Freitas, and J. C. Nievola, "Genetic programming for attribute construction in data mining," in *Proc. EuroGP'03*. Springer-Verlag, 2003, pp. 384–393.

[8] A. Ekárt and A. Márkus, "Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms," *Artif. Intell. Eng. Des. Anal. Manuf.*, vol. 17, no. 3, pp. 205–220, Jun. 2003.

[9] O. Oechsle and A. F. Clark, "Feature extraction and classification by genetic programming," in *Proc. ICVS'08*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 131–140.

[10] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Classifier design with feature selection and feature extraction using layered genetic programming," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 1384–1393, 2008.

[11] K. Badran and P. Rockett, "Multi-class pattern classification using single, multi-dimensional feature-space feature extraction evolved by multi-objective genetic programming and its application to network intrusion detection," *Genetic Programming and Evolvable Machines*, vol. 13, no. 1, pp. 33–63, 2012.

[12] M. Lemczyk and M. Heywood, "Pareto-coevolutionary genetic programming classifier," in *Proc. GECCO '06*. ACM, 2006, pp. 945–946.

[13] A. Awuley, "Feature selection and classification using age layered population structure genetic programming," Master's thesis, Dept of Computer Science, Brock University, 2015.

[14] G. Hornby, "Alps: the age-layered population structure for reducing the problem of premature convergence." in *GECCO '06*. ACM, 2006, pp. 815–822.

[15] G. S. Hornby, "Steady-state alps for real-valued problems." in *Proc. GECCO '09*. ACM, 2009, pp. 795–802.

[16] M. Lichman, "Uci machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[17] J. Koza, *Genetic Programming*. MIT Press., 1992.

[18] S. Luke, "Ecj: A java-based evolutionary computation research system, v22." [Online]. Available: http://www.cs.gmu.edu/~eclab/projects/ecj

[19] R. B. Smith, "Introduction to hyperspectral imaging," 2012. [Online]. Available: http://www.microimages.com/documentation/html/Tutorials/hyprspec.htm

[20] J. dos Santos, A. da Silva, R. da Silva Torres, A. Falco, L. Magalhes, and R. Lamparelli, "Interactive classification of remote sensing images by using optimum-path forest and genetic programming," in *Computer Analysis of Images and Patterns*, ser. LNCS. Springer, 2011, vol. 6855, pp. 300–307.

[21] C. Chion, J.-A. Landry, and L. Da Costa, "A genetic-programming-based method for hyperspectral data information extraction: Agricultural applications," *IEEE Trans. Geoscience and Remote Sensing*, vol. 46, no. 8, pp. 2446–2457, Aug 2008.

[22] P. J. Rauss, J. M. Daida, and S. Chaudhary, "Classification of spectral imagery using genetic programming," in *Proc. GECCO-2000*. Morgan Kaufmann, 10-12 July 2000, pp. 726–733.

[23] M. He, Y. Zhang, Y. Xie, N. Liang, and C. Wen, "Classification of multi-spectral/hyperspectral data using genetic programming and error-correcting output codes," in *Proc. Indust. Elect. and App.* IEEE, May 2006, pp. 1–6.

[24] B. J. Ross, A. G. Gualtieri, F. Fueten, and P. Budkewitsch, "Hyperspectral image analysis using genetic programming." *Appl. Soft Comput.*, vol. 5, no. 2, pp. 147–156, 2005.

[25] M. G. Romay, "Hyperspectral remote sensing scenes." [Online]. Available: http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes

[26] M. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, 2005.

[27] R. Harper, "Spatial co-evolution: quicker, fitter and less bloated." in *Proc. GECCO 2012*. ACM, 2012, pp. 759–766.

[28] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programming*. Lulu Enterprises UK Ltd, 2008.