4

Feature Extraction Using Multi-Objective Genetic Programming

Yang Zhang and Peter I Rockett

Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, S1 3JD, UK
yang.zhang,p.rockett@shef.ac.uk

Summary. A generic, optimal feature extraction method using multi-objective genetic programming (MOGP) is presented. This methodology has been applied to the well-known edge detection problem in image processing and detailed comparisons made with the Canny edge detector. We show that the superior performance from MOGP in terms of minimizing the misclassification is due to its effective *optimal* feature extraction. Furthermore, to compare different evolutionary approaches, two popular techniques - PCGA and SPGA - have been extended to genetic programming as PCGP and SPGP, and applied to five datasets from the UCI database. Both of these evolutionary approaches provide comparable misclassification errors within the present framework but PCGP produces more compact transformations.

4.1 Introduction

4.1.1 Feature Extraction

Over recent decades, the effort made on designing ever-more sophisticated classifiers has almost eclipsed the importance of feature extraction, typically the pre-processing step in a pattern classification system - see Figure 4.1. Indeed many elegant results have been obtained in the area of classification since the 1970s. Nonetheless, feature extraction maintains a key position in the field since it is well-known that feature extraction can enhance the performance of a pattern classifier via appropriate pre-processing of the raw measurement data [2].

To utilize the potential information in a dataset to its maximum extent has always been highly desirable and subset selection, dimensionality reduction and transformation of features (feature extraction) have all been applied to patterns in the past before they are labeled by a classifier. Often though, the feature selection and/or extraction stages are omitted or are implicit in the recognition paradigm - a multi-layer perceptron (MLP) is a good example, where a distinct feature selection/extraction stages are not readily identifiable. In many application domains, such as medical diagnosis or credit scoring, however, the *interpretability* of the features used in classification may contain much important information.

A generic and domain-independent methodology for generating feature extraction stages has hitherto been an unattained goal in pattern recognition. Normally, feature extraction approaches are hand-crafted using domain-specific knowledge and optimality is hard to guarantee - in fact the subject of optimality is rarely even addressed. Indeed, much of image processing research, for example, has been devising what are really feature extraction algorithms to detect edges, corners, *etc.* Ideally, we require some measure of class separability in the transformed decision space to be maximized but with hand-crafted methods, this is usually hard to assess.

Most importantly, no generic and domain-independent methodology exists to *automate* the process of creating or searching for good feature extractors for classification tasks where domain knowledge either does not exist or is incomplete.

Generally speaking, there are two main streams of feature extractions: linear and non-linear. The former is exemplified by principal component analysis (PCA) and related methods and are mainly applied to reduce the dimensionality of the original input space by projecting the data down into the sub-space with the greatest amount of data variability - this does not necessarily equate to optimal *class* separability. To obtain the optimal (possibly non-linear) transformation $\mathbf{x} \rightarrow \mathbf{y}$ from input vector, $\mathbf{x}$ to the decision space vector, $\mathbf{y}$ where:

$$\mathbf{y} = f(\mathbf{x}) \tag{4.1}$$

is a more challenging task. Popular supervised learning network systems such as the multi-layer perceptrons, radial basis functions and LVQs, coupled with powerful training algorithms, can provide model-free or semi-parametric methods to design non-linear mappings between the input and classification spaces from a set of training examples. In such paradigms, the three processing steps shown in Figure 4.1 are merged into a single, indivisible stage. Nonetheless, optimality is hard to guarantee with such methods. For example, with multi-layer perceptrons, critical quantities such the *best* number of hidden neurons are often determined empirically using cross-validation or other techniques.

Following the above insight that the feature extraction pre-processing stage is a mapping from input space to a decision space which can be represented as a *sequence* of transformations, we seek the mapping which maximizes the class separability in decision space and hence the classification performance. Thus the problem reduces to one of finding an optimal sequence of operations, subject to some criterion.

4.1.2 Genetic Programming

Genetic programming (GP) is an evolutionary problem-solving method which has been extensively used to evolve programs, or sequences of operations [1].

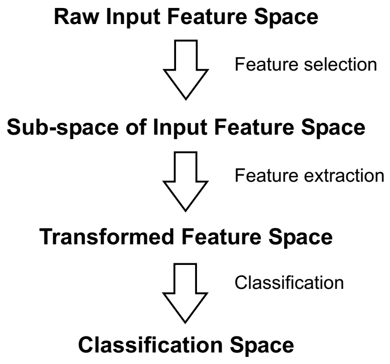

Fig. 4.1. Prototypical pattern recognition system

Typically, a prospective solution in GP is represented as a parse tree which can be interpreted straightforwardly as a sequence of operations. Indeed, GP has been used before to design feature extraction.

Koza [1] has evolved character detectors using genetic programming while Tackett [9] evolved a symbolic expression for image classification based on image features. Bot [4] has used GP to evolve new features in a decision space, adding these one-at-a-time to a k-nearest neighbor (k-NN) classifier until the newly added feature fails to improve the classification performance by more than a predefined amount; in fact, Bot's method is a greedy algorithm and therefore sub-optimal.

Pipelined image processing operations to transform multi-spectral input synthetic aperture radar (SAR) image planes into a new set of image planes were evolved by Harvey *et al.* [29]. A conventional supervised classifier was employed to classify the transformed features. Training data were used to derive a Fisher linear discriminant and GP was applied to find a threshold to reduce the output to a binary image. The discriminability, however, is constrained in the discriminant-finding phase and the GP only used as a one-dimensional search tool to find a threshold.

Sherrah *et al.* [14] proposed an Evolutionary Pre-Processor (EPrep) system which used GP to evolve a good feature (or feature vector) by minimizing misclassification error. Generalized linear machine, k-nearest neighbor (k-NN) and maximum likelihood classifiers were selected randomly and trained in conjunction with the search for feature extractors. The misclassification error over the training set was used as a raw fitness for the individuals in the evolutionary population. This approach not only has a large search space in which to work, but also depends on the classifiers in an opaque way such that there is a potential risk that the evolved pre-processing can be excellent but the classifier can be ill-matched to the task in hand, giving a poor overall performance or *vice versa*.

Kotani *et al.* [23] used GP to determine the polynomial combination of raw features to pass to a k-NN classifier. Krawiec [33] proposed a new method of using GP to design a fixed-length decision-space vector which protects 'useful' blocks during the evolution. Unfortunately, the protection mechanism actually contributes to the over-fitting which is evident in his experiments.

Significantly, all previous work on GP feature extraction has used a single objective comprising either a raw misclassification score or an aggregation of this with a tree complexity measure - see also [25]. The single objective function used in this previous work has a number of shortcomings; indeed, many real-world problems naturally involve the simultaneous optimization of multiple, often conflicting objectives. As far as genetic algorithms are concerned, many methods have been proposed to apply multi-objective optimization: *e.g.* SPEA-II[7, 17], MOGA [19], PCGA [20]. Also see [39] for a detailed review of multi-objective evolutionary algorithms.

4.1.3 Multi-objective Optimization

As we have noted, there are often competing multiple objectives in real world design problems. For a pattern classification system, the obvious design objective is to minimize the misclassification error (over some finite size training set). Unfortunately, unless specific measures are taken to prevent it, the trees in a GP optimization tend to carry-on growing in complexity, a phenomenon known as *bloat*. This results in excessive computational demands, typically accompanied by a degradation in the generalization performance of the trained classifiers. Effectively, bloat is an *over-parameterization* of the problem. To counter this, various heuristic techniques have been tried to suppress bloat, for example, adding a further objective to penalize more complex individuals. Such strategies stem from Occam's Razor - for a given error, the simpler individual is always preferred. Ekárt and Németh [18] have shown that embedding tree complexity within a multi-objective GP tends to prevent bloat by exerting selective pressure in favor of trees of smaller size [7]. Thus a multi-objective framework based on Pareto optimality [16] is presented here.

Both Strength Pareto GP (SPGP) and Pareto Converging GP (PCGP) are applied in the present framework to find the Pareto set for our multi-objective optimization problem - identifying the (near-)optimal sequence of transformations which map input patterns to decision space with maximized separability between the classes. As far as the authors are aware, this is the first report of the comparison between these two evolutionary techniques (generational *vs.* steady-state) applied to multi-objective genetic programming.

The rest of this chapter is organized as follows: Our generic framework for evolving optimal feature extractors will be presented in Section 4.2 In Section 4.3, we will describe the principal application domain to which the methodology will be applied - the edge detection problem in image processing. In Section 4.4, quantitative comparisons are made with the 'gold-standard' Canny edge detector to investigate the effectiveness of the present method on the USF real world dataset [24]. In addition, comparisons between two multi-objective evolutionary strategies - SPGP and PCGP - have been made on five datasets from UCI Machine Learning database [8]. We offer conclusions and thoughts on future work in Section 4.5.

4.2 Methodology

There are two ways of applying evolutionary optimization techniques in the pattern classification domain - one could be regarded as putting the whole system inside the evolutionary learning loop to directly output class labels. This is a tempting route due to its 'simplicity'. Unfortunately, it not as simple as it first appears since the search space is very large, comprising the space of all feature extractors *and* the space of all classifiers. The other option - and the one we favor - is that treating the feature extraction as a stage distinct from classification. Our target is to invest all available computational effort into evolving the feature extraction and to utilize established knowledge from the relatively well-understood area of classifier design. Consequently, we adopt the approach of evolving the optimal feature extraction and performing the classification task using a standard, simple and fast-to-train classifier. The "fast-to-train" requirement on the classifier comes from the fact that we still have to use classification performance as an individual's fitness and thus the classifier has to be trained anew within the evolutionary loop for every fitness evaluation. In addition, evolving a distinct feature extraction stage retains the possibility of human interpretation of the evolved meta-features and therefore the use of our method as a data exploration technique.

Our target is to obtain a generic, domain-independent method to fully automate the process of producing optimal feature transformations. Hence we make no assumptions about the statistical distributions of the original data, nor indeed about the dimensionality of the input space.

4.2.1 Multi-objective Genetic Programming

In order to implement the method, multi-objective genetic programming (MOGP) is used as the search tool driven by Pareto optimality to search for the optimal sequence of transformations. If the optimization is effective, features in the transformed space should not only be much easier to separate than in the original pattern space but should also be *optimal* in the decision space with the classifier employed. In the context of *optimality*, we conjecture that our method should yield a classification performance which is at least as good as the best of the set of all classifiers on a particular dataset [6].

The two evolutionary strategies mentioned above - SPGP and PCGP - are both implemented here on five UCI datasets. The SPGP approach uses a generational strategy with tournament selection to maintain two sets of individuals during evolution, one representing the current population and the other containing the current approximation to the Pareto set. Ranking is done by calculating the strength, or fitness, of each individual in both sets - when calculating an individual's strength, we use the method proposed in SPEA-II [7]. In our modified binary tournament selection method, two individuals from the union of the population and the non-dominated set are randomly selected. If both trees have been drawn from the same set we compare the normalized fitness to determine the winner; if not, the raw fitness vector is used to decide which should be chosen.

For the steady-state PCGP strategy, we followed the method designed in [20] as PCGA with straightforward modifications for use with GP trees. In comparing the two evolutionary strategies, we are aiming to focus on the issue of the search convergence. The Pareto Converging Genetic Algorithm (PCGA) ensures population advancement towards the Pareto-front by naturally sampling the solution space. The motivation for this approach is aimed particularly at solving difficult (real-world) problems with local minima. For detailed information see [20].

We have used a fairly standard chromosomal tree structure for the MOGP implementation. There are two types of nodes in each individual - function nodes and terminal nodes. Details of node types are summarized in Table 4.1.

Non-destructive, depth-dependent crossover [13] and a depth-dependent mutation operator were used to avoid the breaking of building blocks. Here a set of candidate sub-trees is chosen to mutate based on their depth in the tree using the depth-fair operator [13]. Then, one particular sub-tree from this set is selected biased in its complexity (*i.e.* the number of nodes) using roulette wheel selection; that is, we favor mutating more complex trees. Detailed information can be found in [5].

The stopping criterion for SPGP is any of the following being is met:

- The maximum number of generations is exceeded (500 in present case)
- The evolution process stops as adjudged by the Bayes error of the best individual failing to improve for $0.04 \times$ the maximum number of generations (the 0.04 value is an empirical value determined by experiment)
- The misclassification error $= 0$.

For PCGP, the termination condition was set simply as exceeding the maximum number of generations, 2000 in present work. The population size for SPGP in all experiments presented here was 500 while a population of only 200 individuals was used for PCGP. Note that when *fully* converged, all the non-dominated individuals in the PCGP population comprise the Pareto-front. For more detailed information see [20].

4.2.2 Multiple Objectives

Within the multi-objective framework, our three-dimensional fitness vector of objectives comprises: tree complexity, misclassification error and Bayes error, as follows:

Table 4.1. MOGP Settings

Terminal set	Input pattern vector elements
Constant	10 floating point numbers 0.0, . . . , 1.0
Raw fitness vector	Bayes error, misclassification error, number of nodes
Standardized fitness	Strength-based fitness
Original population	Half full-sized trees, half random trees
Original tree depth	5
Probabilities	0.3 mutation, 0.7 crossover

Tree Complexity Measurement

As pointed-out above, tree bloat in GP can produce trees with extremely small classification errors over the training set but a very poor error estimated over an independent validation set. This is clearly an example of the familiar over-fitting phenomenon and grounded on the principle of Occam's Razor, we use the node count of a tree as a straightforward measure of tree complexity. Within the concept of Pareto optimality in which all objectives are weighted equally, this complexity measure exerts a selective pressure which favors small trees.

Misclassification Error

In addition to the tree complexity measure, the fraction of misclassified patterns counted over the training set is used as a second objective [33]. Since the GP tree chromosomes used here naturally lead to an n-to-1 mapping into the one-dimensional decision space, a simple threshold in this decision space is adapted during the determination of fitness value to obtain the minimum misclassification error. This means we are trying to evolve an optimal feature extractor *conditioned* on a thresholding classifier operating in the decision space.

Under Pareto optimality [16], all objectives are treated as equally important since we are aiming to explore the trade-off between the complexity and the misclassification error (over the training set). With the aid of the two competing objectives, we are aiming to maximize the class separability in the decision space using the simplest possible mapping. Nonetheless, during our early experiments, we found that learning is often very slow and sometimes the optimization fails to converge at all. After detailed investigation into the searching process, we found that in the initial stages, when all the randomly-created individuals possessed roughly the same (very high) misclassification error, there was insufficient selective pressure to pick individuals with slightly more promise than the irredeemably poor performers. Thus the search stagnated. Consequently, the third objective of Bayes error was investigated as an additional measure of inter-class separability.

Bayes Error

The search performance of all evolutionary algorithms is critically dependent on the use of appropriate fitness functions. Our motive for choosing the Bayes error as an objective is because it is the fundamental lower bound on classification performance, independent of the class distributions and classifier. In a two class problem, we map the n-dimensional input pattern space into the 1D decision space forming two class-conditioned probability density functions (PDFs) in the decision space and the overlapping region(s) of these two class-conditioned PDFs can be used to estimate the Bayes error with a simple histogramming procedure.

If the Bayes error is used as a direct replacement for the misclassification error, the optimization converges rapidly - clearly the Bayes error is exerts more sensitive selective pressure in the the situation where the population is far from convergence. Unfortunately, the subsequently estimated validation error was disappointingly high. Further investigation revealed that in minimizing the overlap between two class-conditioned PDFs, the GP often achieved this goal by producing two PDFs with non-coincident, *'comb'-like* features as illustrated in Figure 4.2. The Bayes error is indeed small when calculated over the *training set* but this does not generalize to an independent validation set. Clearly what was desired were two well-separated PDFs although the GP was meeting this goal in an unintended and unhelpful way - such opportunistic behavior has been observed previously in evolutionary algorithms. As a consequence, we employed both misclassification error *and* the Bayes error estimate in a three-dimensional fitness vector.

The Bayes error allows the evolutionary search to make rapid initial progress after which the misclassification error objective appears provide selective pressure which separates the transformed distributions. We have also observed that on those occasions when the optimization has got temporarily stuck in a local minimum, it is the Bayes error which improves first and appears to 'lead' algorithm out of the local minimum.

4.3 Application Domain

4.3.1 Edge Detection in Image Processing

Edge detection is an important image processing technique in many applications, such as object segmentation, shape recognition, *etc.* and so we chose this well-understood problem as the starting point to demonstrate our methodology. Edge detection is a well-researched field which should be suitable for assessing the effectiveness of the evolved feature extraction method in comparison to an established and conventional method *e.g.* the Canny edge detector; the Canny algorithm is widely held to be a 'gold standard' among edge detection algorithms [21, 22, 24]. Harris [34] used GP with a single objective

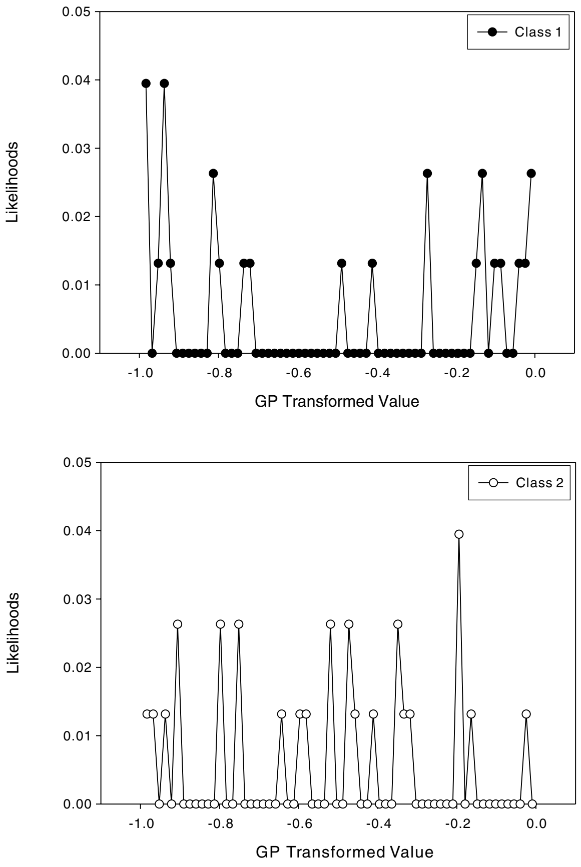

Fig. 4.2. Example of *'comb'-like* class-conditioned densities evolved using the Bayes error metric alone

to evolve an 'optimal' edge detector but terminated the evolution when he obtained a performance comparable to the Canny detector See also [10, 11].

We used a synthetic dataset for training on the edge detection task because Chen *et al.* [30] concluded that hand-labeling real image training set did not adequately sample the pattern space, leading to deficient learning. We have followed a very similar approach of synthesizing a training set from a physically realistic model of the edge imaging process. Three distinct types of patterns are identified: edges, non-obvious-non edges and uniform patches. Further details can be found in [30, 31, 32]. We have employed an image patch size of 13 $\times$ 13 in the generated training set - probably larger than is needed - deliberately to investigate whether the most useful features would be selected by the objective of minimizing the tree size, and by implication, the number of raw input features used. The training set comprised 10,000 samples and the realistic figure of 0.05 is chosen for the prior probability of the edge class. See also [6].

The Canny algorithm includes a significant number of post-processing steps, such as non-maximal suppression (NMS) which appear to be responsible in large measure, if not completely, for its superiority over other conventional edge detectors [32]. The NMS step is a heuristic sequential and spatially-localized classification stage, which serves to reduce the fraction of false positives (*FPs*) with a slight attendant sacrifice in the fraction of true positives (*TPs*). Our interest here is a domain-independent methodology and a step such as NMS is very much a heuristic specific to the image processing domain; nonetheless we have included NMS in our comparisons since it is held to be an integral part of the Canny algorithm.

Whereas the our GP detector has no adjustable parameters (once trained), the edge labelling threshold is a user-defined tuning parameter in the Canny algorithm. Thus fair comparison is not completely straightforward. The principled basis we have chosen for comparison between the GP-generated and Canny edge detectors is the operating points which minimize the Bayes risk for both detectors. The operating point which naturally emerges from our GP algorithm is that which minimizes the misclassification error over the training set which was constructed with edge prior of 0.05. For the Canny algorithm we can locate the corresponding decision threshold by plotting the Bayes risk *versus* threshold to locate the optimal operating point (*i.e.* the minimum Bayes risk) [15]. The Bayes risk can be written as:

$$R = P \times (1 - TP) + (1 - P) \times FP \tag{4.2}$$

where, TP and FP are the fractions of true and false positives, respectively. P is the prior of edge. Generally, both TP and FP will vary with threshold and other operating conditions. Through locating the minimum of the risk-threshold plot we can identify the 'optimal' operating point of the detector at the given prior.

Note we have used the assumption of equal costs: In fact, cost ratios are always subjectively chosen and vary from application to application. For example, in medical image processing, the cost of a false negative may be unacceptably high and so a suitable cost would be used, biasing the classifier operating point. In other applications, false negatives resulting in line fragmentation, may be tolerable in order to keep processing times below some limit. Here we adopt a neutral position of using a cost ratio of unity since we have no basis for regarding one sort of error as more or less important than any other.

Detailed comparison using synthetic datasets between the MOGP edge detector and the Canny algorithm had been made elsewhere [5]. In order to investigate the labeling performance on real image data, we have applied the GP-generated edge detector to images taken from the ground-truth labeled USF dataset [24] and drawn comparison with the Canny edge detector with and without NMS; the results are presented in Section 4.4.

4.3.2 UCI data

In addition to the edge detection task, we have applied our method to five other datasets from the UCI Machine Learning databases [35] in Section 4.4.2 where we make comparison between the two evolutionary strategies - SPGP and PCGP - to investigate the relative merits of generational and steady-state evolutionary techniques. The datasets used in the current work are:

- BUPA Liver Disorders (BUPA): To predict whether a patient has a liver disorder. There are two classes, six numerical attributes and 345 records.
- Wisconsin Diagnostic Breast Cancer (WDBC): This dataset has been discussed before by Mangasarian *et al.* [26]. 569 examples with thirty numerical attributes.
- Pima Indians Diabetes (PID): All records with missing attributes were removed. This dataset comprises 532 complete examples with seven attributes.
- Wisconsin Breast Cancer (WBC): Sixteen instances with missing values were removed; 683 out of original 699 instances have been used here. Each record comprises ten attributes. This dataset has been used previously in [27].
- Thyroid (THY): This dataset includes 7200 instances with 21 attributes (15 are binary, 6 are continuous) from 3 classes. It has been reconfigured as a two-class problem with 166 instances from class1 and the remaining, 7034 instances from non-class1. This dataset has been discussed by [36].

For convenience, the details of the datasets used in the current work are summarized in Table 4.2.

Table 4.2. Five UCI Datasets

Name	Number of Features	Size and Distributions
BUPA	6	345 = 200 (Benign) + 145 (Malignant)
WDBC	30	569 = 357 (Benign) + 212 (Malignant)
PID	7	532 = 355 + 177 (Diabetic)
WBC	10	699 = 458 (Benign) + 241 (Malignant)
THY	21	7200 = 166 (Class1) + 7034 (Others)

4.4 Results

4.4.1 Comparisons on USF Datasets

In order to examine the performance of the GP edge detector as well as make fair comparison with the Canny algorithm, we have assessed the performance of both detectors with and without the NMS post-processing step.

As pointed out in our previous work [6], fair comparisons turn-out to be somewhat harder than first appear. We are trying to compare the GP edge detector (with or without NMS) to the Canny edge detector (with or without NMS) at the detector operating points which minimize the Bayes risk. The principal complication here is that the USF dataset has been subjectively censored since the non-obvious non-edge patterns [24],[32] - which are the patterns most likely to be confused by any classifier - have been hand-labeled as belonging to a distinct "don't care" class. Hence we have used the following methodology: Quantifying the performance of the Canny detector over the USF images is straightforward. We have used the same optimal threshold as determined over the synthetic data [5],[6] which assumes an edge prior of 0.05 - the GP detector assumes this same prior therefore neither detector is comparatively disadvantaged. The labeling performance is summarized in Table 4.3 for each of the USF images shown in Figs 4.3(a) - 4.6(a).

Table 4.3. Canny [TP, FP] Operating Points for USF Test Images

Figure	Edge Prior	Without NMS		With NMS	
		TP	FP	TP	FP
4.3	0.087	0.0003	0.0001	0.3821	0.0466
4.4	0.080	0.0204	0.0003	0.4326	0.0695
4.5	0.211	0.0142	0.0063	0.3886	0.0061
4.6	0.066	0.0048	0.0003	0.3580	0.0049

Table 4.4. GP [TP, FP] Operating Points for USF Test Images

Figure	Edge Prior	Without NMS		With NMS	
		TP	FP	TP	FP
4.3	0.087	0.5045	0.0278	0.3657	0.0083
4.4	0.080	0.4151	0.0298	0.3388	0.0052
4.5	0.211	0.6228	0.0129	0.4433	0.00036
4.6	0.066	0.5581	0.0246	0.3415	0.0047

It is again straightforward to obtain the labeling performance of the GP detector *without* NMS and these results are shown in Table 4.4. To determine the GP performance *with* NMS we have devised a special method of carrying-out non-maximal suppression on the output of the GP detector. First, we estimate the orientation of an edgel using the familiar difference-of-boxes operator. We then quantize the edge direction into one of the eight principal directions of the compass and examine the decision variable responses of the GP detector for the three pixels centered on the edge and (approximately) normal to the edge direction. The 'distance' of a given edge pixel's response from the decision threshold can be taken as a measure of its edge strength and we perform non-maximal suppression on this quantity. The results are again summarized in Table 4.4.

It is apparent from Table 4.3 that the Canny algorithm without NMS performs very poorly, an observation consistent with the preceding results on synthetic edge data reported in [6]. In particular, the *TP* values are very low. Consistent to the conclusion made in [32] that there is no clear minimum risk operating point for the Canny algorithm without NMS. Hence the success of this algorithm owes very little to the feature extraction (pre-processing) step, a finding which will be a surprise to many in the image processing community.

Table 4.5. Bayes Risk Comparisons for USF Test Images

Figure	GP		Canny	
	Without NMS	With NMS	Without NMS	With NMS
4.3	0.06848	0.06276	0.0871	0.09630
4.4	0.07420	0.05768	0.0786	0.10933
4.5	0.08977	0.11774	0.2129	0.13381
4.6	0.05214	0.04785	0.06596	0.04694

With NMS, the Canny algorithm has a much higher *TP* fraction although the *FP* fraction also increases. This is again consistent with the result reported in [32] that after the NMS step, there is a clearer optimal operating point in terms of minimization of Bayes risk. Indeed, after NMS the (increased) *FP* fraction becomes the principal contributor to the Bayes risk.

The received wisdom in the image processing community would suggest that the difference between the detector output with and without NMS can be explained by the thinning of otherwise thick edges - this turns-out not to be the case since NMS significantly changes the ROC of the Canny edge detector and hence its optimal operating point.

From the comparisons we can see that before NMS, GP gives larger *TP* values while after NMS, the two algorithms give comparable *TP* values. The Canny algorithm, however, gives much bigger *FP* values. Comparisons of the Bayes risk figures are shown in Table 4.5 from which it can be seen that the risk values of the Canny algorithm are higher than for the GP, with or without NMS. The only exception to this is the image in Fig. 4.6 where, after NMS, the GP has a slightly higher risk. The reason seems to be that the noise level in this image is much lower than the others, allowing the Canny algorithm to yield a higher *TP* value while having an *FP* value roughly equal to the GP detector. In fact, it is not statistically significantly different using t-test under 95% confidence level. The final labeled images in Fig 4.6 (e, f) for Canny and GP (with NMS) look much more similar than for any of the other images from the USF dataset.

In the USF dataset, a number of regions have been labeled as “don’t care” - the white regions in Fig. 4.3b, 4.4b, 4.5b, and 4.6b are the distinct “don’t care” class. For the edge detection problem we have concluded that it is these non-obvious non-edge (NONE) patterns labeled as “don’t care”in the USF images which make the classification task difficult. This is consistent with the observation of Konishi *et al.* [22] that the USF images are easier to label than the Sowerby dataset which these authors also considered and which has no ”don’t care” regions.

Whole image labeling results for four, typical USF images using the Canny and GP detectors (with and without NMS) are shown in Fig. 4.3 to Fig. 4.6. In these figures, (a) denotes the original image from the USF dataset, (b) shows the ground truth data, (c) shows the labeling results from the Canny detector without NMS and (d) shows images labeled with the GP feature extractor; (e) shows images from the Canny edge detector with NMS, (f) shows the output from the GP feature extractor with the gradient-direction based NMS.

In these labeling results, it is striking that Fig. 4.3(c) contains only six labeled pixels in the top right hand corner of the image and the same poor performance in other corresponding (c) figures are consistent with the results from Table 4.3. These results further confirm our conclusion that the Canny edge detector’s performance is not due to the feature extraction stage but to the sophisticated post-processing steps coupled with subjectively - and implicitly - set cost ratios.

Discussion

The evolution described here, driven by multiple objectives is able to generate separated class-conditioned distributions in the 1D decision space. In contrast to the hand-crafted, heuristic post-processing steps of the Canny edge detector [21], we concentrate our computational resource on optimizing the feature extraction stage to yield greater separability in the mapped feature space. Further, compared to the Canny detector which is based on extensive domain knowledge - see Canny’s original work on deducing the ‘optimal’ filter kernel [21] - we did not supply *any* domain-dependent knowledge to MOGP apart from the carefully constructed training set. This lends evidence to support our conjecture that our method is able to automatically produce (near-)optimal feature extraction stages of a classification system.

Although feature selection was not explicitly intended in our approach, we have deliberately employed an overly large image patch (13×13) to investigate how the GP selects input features within this patch given that one of our fitness objectives is to minimize tree size and, therefore, the number of leaf nodes (*i.e.* individual pixel intensity values). A histogram of the number of times each pixel was used by the trees in a typical converged Pareto set is shown in Fig. 4.7, where the central pixel of the 13 × 13 patch has the row and column indices of (0, 0). This figure illustrates that the MOGP optimization has a strong tendency to select pixels from around the center of the image patch, a fact which is intuitively pleasing because most of the edge/non-edge discriminatory information can be considered to come from this region. We reiterate that we have not embedded any domain knowledge in this optimization. Thus we believe that feature selection is occuring as a beneficial by-product of the feature extraction process due to the way we have incorporated parsimony into the optimization.

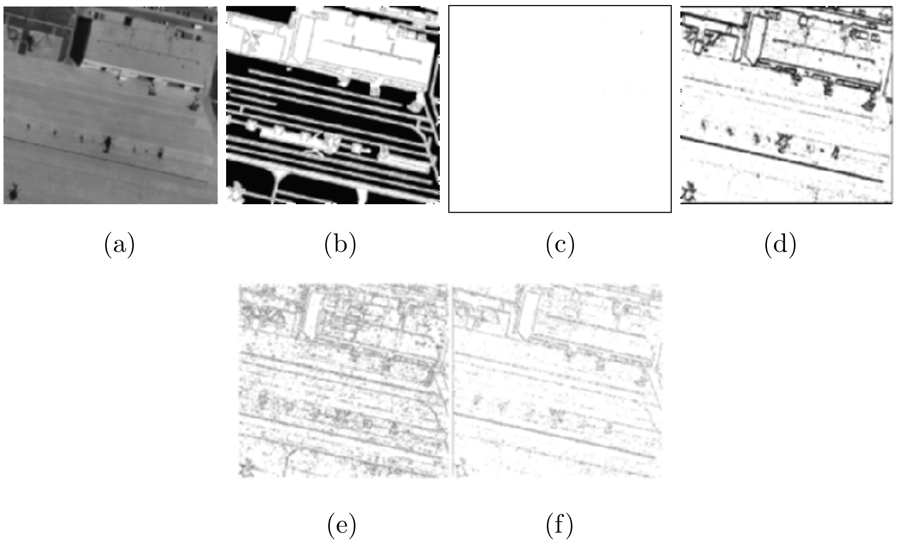

Fig. 4.3. a to f illustrate the comparisons from the Canny edge detector and GP, details refer to the text.

4.4.2 Comparison of Generational and Steady-state Evolutionary Algorithms on UCI Datasets

As mentioned in the introduction, evidence to substantiate the generic property of our method has been demonstrated in previous work [6] where we have shown that our methodology, implemented using the SPGP algorithm, exhibited either superior or equivalent performance to nine conventional classifiers. Here we report the results of a preliminary investigation into the efficacy and performance of different evolutionary techniques within the proposed framework. SPGP and PCGP have been compared on five typical problems from the

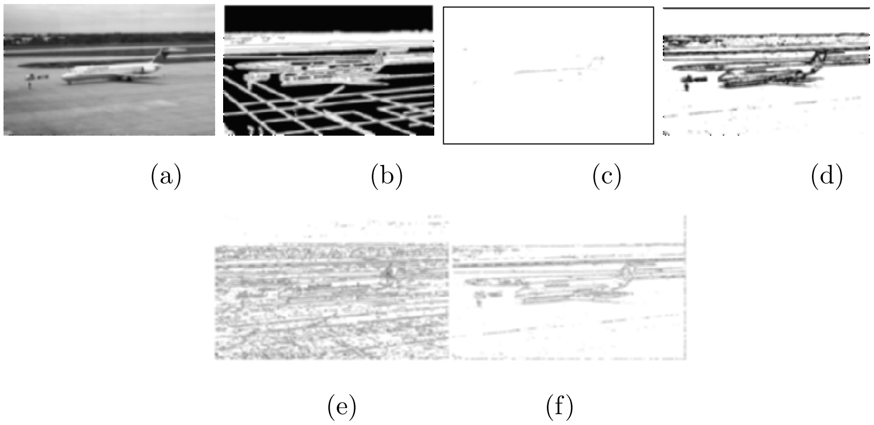

Fig. 4.4. a to f illustrate the comparisons from the Canny edge detector and GP, details refer to the text.

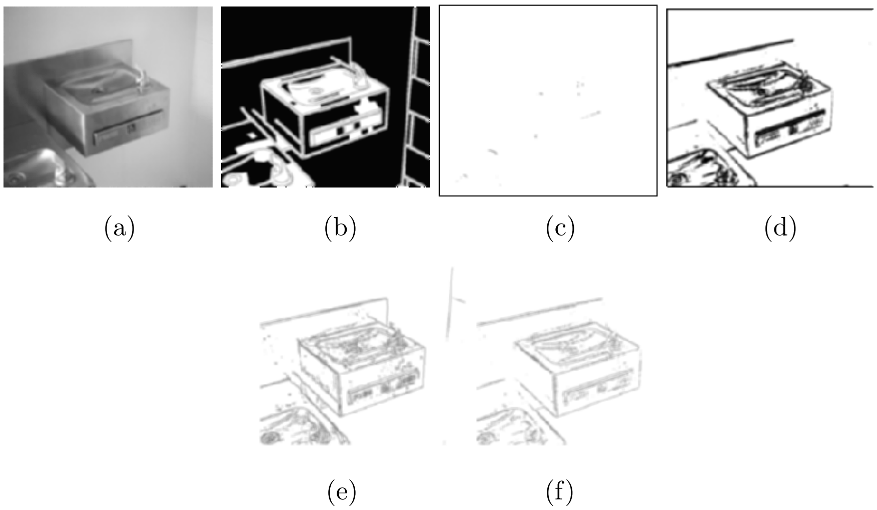

Fig. 4.5. a to f illustrate the comparisons from the Canny edge detector and GP, details refer to the text.

UCI database [35]. As far as we are aware, there has been little detailed analysis and comparison of multi-objective evolutionary strategies on real world GP problems.

We have applied SPGP and PCGP to each of the problems listed in Table 4.2. The GP settings are the same as those listed in Table 4.1 except that the terminal nodes are now the elements in the pattern vectors of the five UCI datasets rather than image pixel values. For the generational SPGP algorithm,

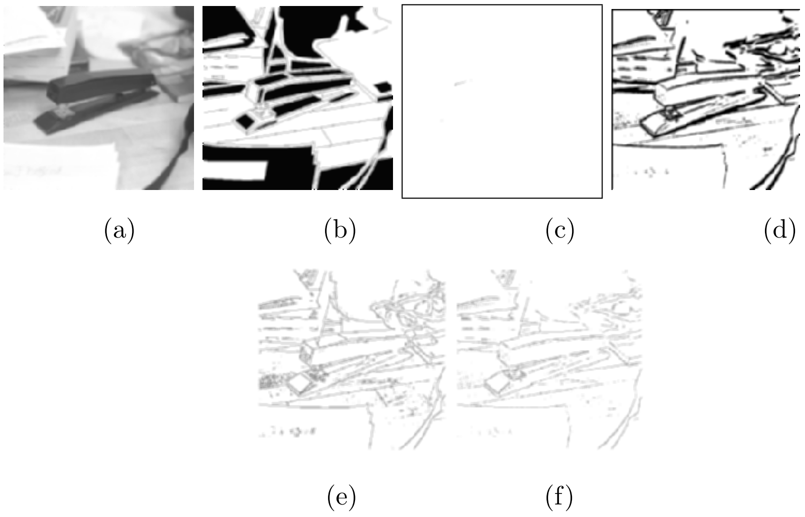

Fig. 4.6. a to f illustrate the comparisons from the Canny edge detector and GP, details refer to the text.

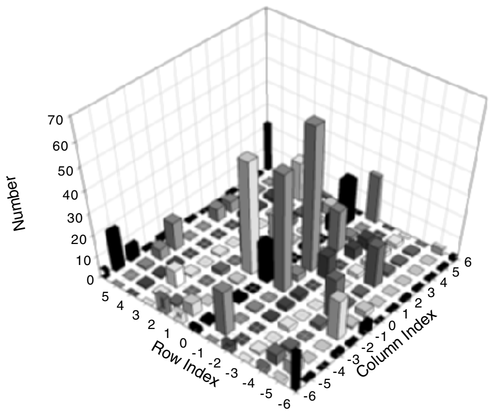

Fig. 4.7. Histogram of numbers of pixels used in the GP trees in a typical Pareto set, relative to the 13 × 13 input patch.

we used a population size of 500 and a maximum of 500 generations; the three possible stopping criteria listed in Table 4.1 are reused. For the steady-state PCGP algorithm, only 200 individuals were used in the population, while for simplicity, the stopping criterion used was to run for a fixed number of 2000 generations. This means that PCGP is performing around one-tenth the number of fitness evaluations of the SPGP algorithm and is thus comparatively disadvantaged; this was an intentional feature of this preliminary study to determine if approximately equivalent solutions could be obtained with PCGP but with much less computing resource. A sophisticated method of gauging actual convergence of PCGA have been discussed by Kumar & Rockett [20].

Table 4.6. Mean Error Comparisons Between SPGP and PCGP on Five UCI Datasets

Datasets	SPGP			PCGP		
	BE	ME	Mean Nodes	BE	ME	Mean Nodes
BUPA	0.185	0.272	58.928 ± 54.636	0.219	0.278	7.830 ± 5.242
PID	0.160	0.203	32.633 ± 40.133	0.173	0.200	3.870 ± 1.401
WBC	0.019	0.025	24.071 ± 46.320	0.014	0.022	6.280 ± 2.647
WDBC	0.025	0.028	36.941 ± 43.417	0.028	0.028	7.785 ± 5.343
THY	0.0044	0.0061	37.964 ± 43.299	0.0044	0.0058	11.220 ± 5.492

The mean error comparisons between the two evolutionary algorithms over 10 repetitions are summarized in Table 4.6, where ME stands for misclassification error from optimal thresholding in the decision space after feature extraction. BE, the corresponding Bayes error estimates are listed to give an indication of the degree to which the misclassification error approaches its fundamental lower bound. Table 4.6 contains only data from the *non-dominated* solutions from each of the evolutionary paradigms. We have applied Alpaydin's F-test [28] to these data from which we conclude that none of these differences in error is statistically significant. What *is* a notable and significant difference between these two algorithms is the numbers of mean nodes required to obtain 'identical' misclassification errors. The mean numbers of nodes for the steady-state PCGP approach are very much smaller than for the generational SPGP algorithm. We have observed that some inactive/redundant sub-trees exist in solutions evolved by SPGP [6] and the fact that we generate significantly smaller trees using PCGP implies that PCGP is much more effective in controlling tree bloat than SPGP. This result saves a lot of computational effort during evolution as well as having a practical implication.

To further explore the differences between the 'quality' of the solutions produced by these two evolutionary paradigms, we have plotted the misclassification error *versus* the number of nodes for the final solutions produced by the two algorithms on each of the datasets in Figures 4.8 to 4.12. Although the SPGP results shown are this algorithm's best approximation to the Pareto front, for PCGP we have plotted the *whole* population which in practice, contains a number of *dominated* solutions. Our aim here is to compare the coverage and sampling of the Pareto front produced by the two algorithms, albeit at an intermediate stage for PCGP. Note that these figures do *not* depict Pareto-optimal sets *per se* since these plots are a two-dimensional representation of a three-dimensional Pareto front. This in part explains why a number of solutions appear at first glance to be dominated (in the Pareto

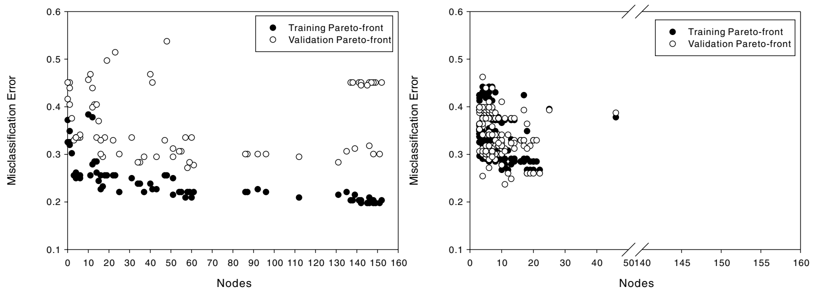

Fig. 4.8. Misclassification error *vs.* Number of nodes for the members of the Pareto sets generated by SPGP (left) and PCGP (right) on the BUPA dataset

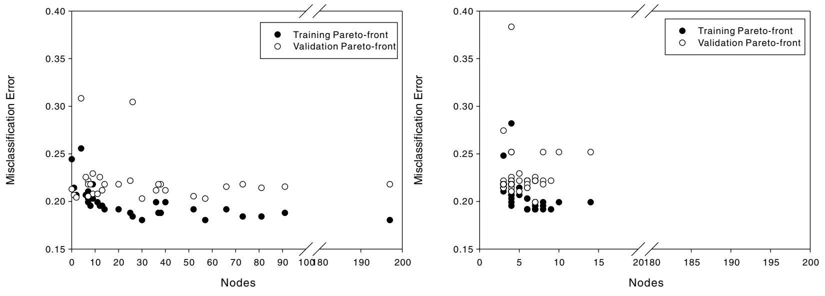

Fig. 4.9. Misclassification error *vs.* Number of nodes for the members of the Pareto sets generated by SPGP (left) and PCGP (right) on the PID dataset

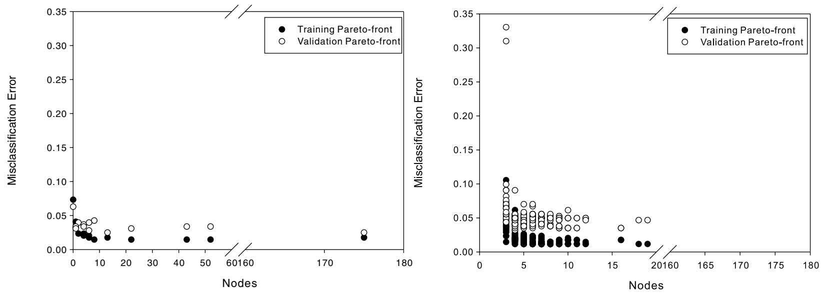

Fig. 4.10. Misclassification error *vs.* Number of nodes for the members of the Pareto sets generated by SPGP (left) and PCGP (right) on the WBC dataset

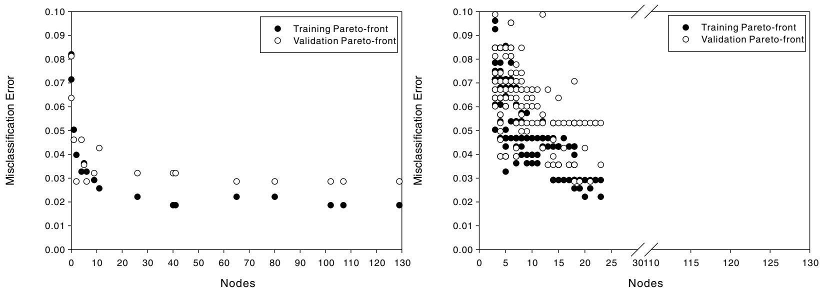

Fig. 4.11. Misclassification error *vs.* Number of nodes for the members of the Pareto sets generated by SPGP (left) and PCGP (right) on the WDBC dataset

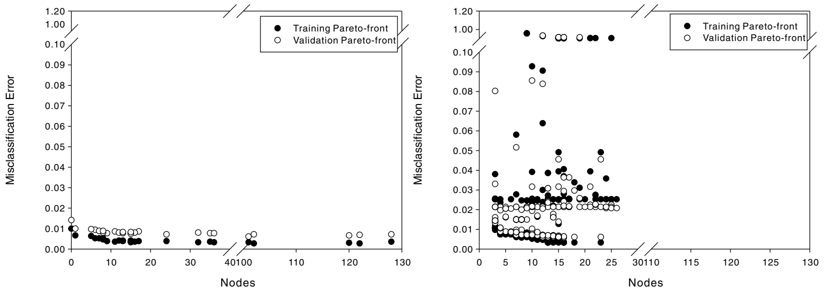

Fig. 4.12. Misclassification error *vs.* Number of nodes for the members of the Pareto sets generated by SPGP and PCGP on the THY dataset

sense) whereas in fact, they are actually non-dominated if one takes the third (Bayes error) objective into account. (Indeed, many of the PCGP solutions actually are dominated within their population.)

From Figures 4.8 to 4.12, it is apparent that PCGP produces a solution set which is much more strongly centered on lower node numbers. This preliminary comparison of PCGP is thus extremely promising since even in the state of not being fully converged and only having had around one-tenth the computing resources invested in it, PCGP is able to produce solutions which are significantly smaller than those from SPGP but with indistinguishable error rates.

Tree Interpretation

By way of example, we show a typical GP-generated tree in Figure 4.13. This individual was a non-dominated population member obtained from a PCGP run on the THY dataset; we present the tree as it was generated without pruning. The leaf nodes in Figure 4.13 are labeled as Xn, for $\in [1...N]$, where N is the number of raw attributes in the input pattern vector; for the THY dataset here, $N = 21$ (see Table 4.2). All the function nodes in the graph are elementary functions except *pow2* which calculates the child value raised to the power of two and $ifelse$ which returns the second child value if the first child value is 0, otherwise it returns the third child value. Actually, in the example tree shown in Figure 4.13, the sub-tree containing $ifelse$ could be simplified as $X19$. Although it is trivial to remove this obvious redundancy by hand (or even as a non-evolutionary post-processing stage), it appears that even the PCGP evolutionary strategy has the propensity to generate inactive code.

The value returned from the tree root - $-(minus)$ - is the transformed feature value in the one-dimensional decision space in which the separability between classes is maximized. Each input pattern will be fed into the tree and will return a scalar value as the extracted feature. Again, the set of raw input features has been selected during the optimization process which seeks to minimize the number of tree nodes. Surprisingly, with this dataset only two original elements can be used to extract the *optimal* new feature to obtain the desirable classification performance.

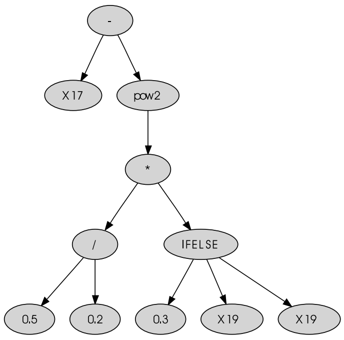

Fig. 4.13. One typical evolved GP feature extractor on THY dataset

4.5 Conclusions and Future Research Work

In the present work we conjecture that efficient feature extraction can yield a classification system, the performance of which is *at least* as good as the best of all available classifiers on any given problem. In essence, the evolutionary algorithm is 'inventing' the optimal classifier for a given labeling problem/dataset. It is, of course, entirely feasible that the genetic optimization is re-inventing an existing classifier rather than devising a novel algorithm but this is of little practical consequence. When confronted with a new dataset, the common approach among pattern recognition practitioners is to empirically try a range of classification paradigms since there is no principled method for predicting which classifier will perform best on a given problem; there is always the chance that better performance could be obtained with another, but untried classifier. Our conjecture that multi-objective evolutionary optimization produces the best possible classifier effectively eliminates the risk from this trial-and-select approach. Needless to say, although the results presented here and in previous work [6] support our conjecture, by definition, we cannot offer a definitive proof.

Also of considerable importance is the fact that we do require any domain knowledge of the application. The feature extraction transformation sequence is identified automatically, driven solely by the optimality criteria of the multi-objective algorithm. We have presented the application to edge detection, for example, in which our method yields superior results to the very well-established and mature Canny algorithm. The generic property of the presented method enables it to be straightforwardly applied to other application domains.

Although the principle of evolutionary optimization is straightforward, exact *implementation* is still an open research issue. Here we present the preliminary results of applying two different evolutionary paradigms, one generational (SPGP) and one steady-state (PCGP). The differences between the misclassification errors attained with the two types of genetic search are not statistically significant although the *complexity* of the generated feature transformation sequences is markedly different. Even where the PCGP algorithm has not fully converged, we find that the steady-state method produces much smaller trees which implies that PCGP is more responsive to the objective trying to minimize tree size. The concentration of the solutions at low node numbers also means that the sampling of the Pareto front is better with PCGP. (Previous comparisons between PCGA and other multi-objective GAs imply a similar, fundamental superiority in the steady-state approach [20].) Faster convergence and better control of the tree-bloating make the PCGP approach very promising for practical applications.

We have demonstrated the use of multi-objective genetic programming (MOGP) to evolve an "optimal" feature extractor which transforms the input patterns into a decision space such that pattern separability in this decision space in is maximized. In the present work we have projected the input pattern to a *one-dimensional* decision space since this transformation naturally arises from a genetic programming tree although potentially, superior classification performance could be obtained by projecting into a multi-dimensional decision space [14] - this is currently an area of active research.

Finally, as with all multi-objective optimizations, what arises from our method is a family of equivalent solutions - the Pareto set - which presents the system designer with the trade-off surface between classification performance and the complexity of the feature extractor. Exactly which pre-processing solution is selected for the final application will depend on the *generalization* properties of the solutions - invariably one would like the solution which is best able to predict class of as-yet unseen patterns with the greatest accuracy. Predicting generalization performance is very much a major and continuing issue in statistical pattern recognition - ultimately we would like to formulate a measure of classifier generalization which potentially, could be optimized along with the other multiple objectives. This remains an area of open research.

References

[1] J.R. Koza. Genetic Programming II, Automatic Discovery of Reusable Programs. The MIT Press, Cambridge, Massachusetts, 1994

[2] R. O. Duda and P. E. Hart and D. G. Stork. Pattern Classification (2nd Edition), Wiley-Interscience, 2000

[3] D. Addison, S. Wermter and G. Arevian. A comparison of feature extraction and selection techniques. In: *Proceedings of the International Conference on Artificial Neural Networks*, Istanbul, Turkey, Supplementary Proceedings, pp. 212–215, 2003

[4] M. C. J. Bot. Feature extraction for the k-Nearest neighbor classifier with genetic programming. In: *Genetic Programming, Proceedings of EuroGP'2001*, Lake Como, Italy, pp. 256-267, 2001

[5] Y. Zhang and P.I. Rockett. Evolving optimal feature extraction using multi-objective genetic programming: A methodology and preliminary study on edge detection. In: *GECCO 2005*, pp. 795-802, 2005

[6] Y. Zhang and P.I. Rockett. A Generic Optimal Feature Extraction Method using Multiobjective Genetic Programming: Methodology and Applications. *IEEE Trans. Systems, Man, and Cybernetics*, 2005 (submitted)

[7] S. Bleuler, M.Brack, L. Thiele, and E.Zitzler. Multiobjective genetic programming: Reducing bloat using SPEA2. In: *Congress on Evolutionary Computation (CEC 2001)*, pp. 536-543, 2001

[8] Z.J. Huang, M. Pei, E. Goodman, Y. Huang, and G. Liu. Genetic algorithm optimized feature transformation - A comparison with different classifiers. In: *GECCO 2003*, LNCS 2724, pp. 2121-2133, 2003

[9] W.A. Tackett. Genetic programming for feature discovery and image discrimination. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 303-309, 1993

[10] M. Ebner and A. Zell. Evolving a task specific image operator. In:*Joint Proceedings of the First European Workshops on Evolutionary Image Analysis, Signal Processing and Telecommunications (EvoIASP'99 and EuroEcTel'99)*, Göteborg, Sweden, Springer-Verlag, pp. 74-89, 1999

[11] M. Ebner. On the evolution of interest operators using genetic programming. In: *Late Breaking Papers at EuroGP'98: the First European Workshop on Genetic Programming*, Paris, France, pp. 6-10, 1998

[12] M.D. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. Comparison of edge detectors: A methodology and initial study. In: *Computer Vision and Pattern Recognition, Proceedings CVPR '96*, pp. 143-148, 1996

[13] T. Ito, I. Iba, and S. Sato. Non-destructive depth-dependent crossover for genetic programming. In: *Proceedings of the First European Workshop on Genetic Programming*, LNCS, Paris, pp. 14-15, 1998

[14] J.R. Sherrah, R.E. Bogner, and A. Bouzerdoum. The evolutionary preprocessor: Automatic feature extraction for supervised classification using genetic programming. In: *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Stanford University, CA, USA. pp. 304-312, 1997

[15] T. Kanungo and R.M. Haralick. Receiver operating characteristic curves and optimal Bayesian operating points. In: *International Conference on Image Processing - Proceedings*, vol.3, pp. 256-259, Washington, DC., 1995

[16] C.A.C. Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In: *Congress on Evolutionary Computation*, pp. 3-13, Washington, D.C., 1999

[17] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. *Technical Report*, 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1998

[18] A. Ekárt and S.Z. Németh. Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming. *Genetic Programming and Evolvable Machines*, vol. 2, pp. 61-73, 2001

[19] C.M. Fonseca and P.J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms -Part I: A unified formulation. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, vol. 28, pp.26-37, 1998

[20] R. Kumar and P.I. Rockett. Improved sampling of the Pareto-Front in multiobjective genetic optimization by Steady-State evolution: A Pareto converging genetic algorithm. *Evolutionary Computation*, vol.10, no. 3, pp. 283-314, 2002

[21] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1986

[22] S. Konishi, A.L. Yuille, J.M. Coughlan, and S.C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 57-74, 2003

[23] M. Kotani, M. Nakai, and K. Akazawa. Feature extraction using evolutionary computation. In: *Proceedings of the Congress of Evolutionary Computation*, IEEE Press, pp. 1230-1236, 1999

[24] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical ROC curves. *Computer Vision and Image Understanding*, vol.84, no.1, pp. 77-103, 2001

[25] D. P. Muni, N. R. Pal, and J. Das. A novel approach to design classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp.183-196, 2004

[26] O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, vol. 43, no. 4, pp. 570-577, 1995

[27] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, vol. 23, pp. 1-18, 1990

[28] E. Alpaydin. Combined 5 2 cv F-test for comparing supervised classification learning algorithms. *Neural Computation*, vol. 11, no. 8, pp. 1885-1892, 1999

[29] N.R. Harvey, S.P. Brumby, S. Perkins, J.J. Szymanski, J. Theiler, J.J. Bloch, R.B. Porter, M. Galassi and A.C. Young. Image feature extraction: GENIE vs conventional supervised classification techniques. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 2, pp. 393-404, 2002

[30] W.C. Chen, N.A. Thacker and P.I. Rockett. An adaptive step edge model for self-consistent training of a neural network for probabilistic edge labeling. *IEE Proceedings - Vision, Image and Signal Processing*, vol. 143, no.1, pp. 41-50, 1996

[31] P.I. Rockett. Performance assessment of feature detection algorithms: A methodology and case study on corner detectors. *IEEE Transactions on Image Processing*, vol.12, no.11. pp. 1668-1676, 2003

[32] Y. Zhang and P.I. Rockett. The Bayesian operating point of the Canny edge detector. *IEEE Trans. Image Processing*, 2005 (submitted)

[33] K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, vol.3, no.4, pp.329-343, 2002

[34] C. Harris. An investigation into the application of genetic programming techniques to signal analysis and feature detection. PhD. thesis, Dept. Comp. Science., Univ. College of London, Sep. 1997.

[35] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998

[36] W. Schiffmann, M. Joost, and R. Werner. Synthesis and performance analysis of multilayer neural network architectures. Technical Report 16/1992, University of Koblenz, Institute für Physics, 1992.