Noname manuscript No.
(will be inserted by the editor)

Symbiotic Coevolutionary Genetic Programming: A Benchmarking Study under Large Attribute Spaces

John A. Doucette · Andrew R. McIntyre · Peter Lichodzijewski · Malcolm I. Heywood

Received: date / Accepted: date

Abstract Classification under large attribute spaces represents a dual learning problem in which attribute subspaces need to be identified at the same time as the classifier design is established. Embedded as opposed to filter or wrapper methodologies address both tasks simultaneously. The motivation for this work stems from the observation that team based approaches to Genetic Programming (GP) have the potential to design multiple classifiers per class – each with a potentially unique attribute subspace – without recourse to filter or wrapper style preprocessing steps. Specifically, competitive coevolution provides the basis for scaling the algorithm to data sets with large instance counts; whereas cooperative coevolution provides a framework for problem decomposition under a bid-based model for establishing program context. Symbiosis is used to separate the tasks of team / ensemble composition from the design of specific team members. Team composition is specified in terms of a combinatorial search performed by a Genetic Algorithm (GA); whereas the properties of individual team members and therefore subspace identification is established under an independent GP population. Teaming implies that the members of the resulting ensemble of classifiers should have explicitly *non-overlapping* behaviour. Performance evaluation is conducted over data sets taken from the UCI repository with 649 to 102,660 attributes and 2 to 10 classes. The resulting teams identify attribute spaces 1 to 4 orders of magnitude smaller than under the original data set. Moreover, team members generally consist of less than 10 instructions; thus, small attribute subspaces are not being traded for opaque models.

Keywords Feature Subspace Selection, Problem Decomposition, Symbiosis, Coevolution, Model Complexity, Classification, Genetic Programming

J. A. Doucette
David R. Cheriton School of Computer Science, University of Waterloo, ON. Canada
E-mail: j3doucet@cs.uwaterloo.ca

A. R. McIntyre, P. Lichodzijewski and M. I. Heywood
Faculty of Computer Science, Dalhousie University, NS. Canada
E-mail: {armcnty, piotr, mheywood}@cs.dal.ca

1 Introduction

Data mining represents an activity in which the goal is to understand more regarding the properties / structure of the application domain, as embodied in the data, through machine learning. As such, the authors interpret this as a multifaceted requirement. The goal is not merely to provide best case classification accuracy, but also to facilitate the development of 'knowledge' from a solution; hereafter interpreted in terms of a general requirement for model simplicity. The relative importance of the two factors is naturally very much in the eye of the beholder. However, it is apparent that a lot of emphasis has been placed on the former, to the possible decrement of the latter [10, 11]. In addition there are several other factors that place real limits on the practical value of proposed approaches. Specifically, we highlight the issues of scaling to large data sets and recognizing the significance of class imbalance; where we take the view that the two requirements need addressing simultaneously.

The increasing availability of electronic document repositories, bioinformatic sources and network logs are all a testament to this trend. Moreover, the specific focus in this work on large attribute spaces implies that the number of training exemplars that can be retained in cache will only ever be a small fraction of the total training partition. In the case of class imbalance, we note that even a 'balanced' data set will consist of P/C exemplars for each in-class partition and $P-P/C$ out-of class exemplars; where there are P training exemplars and C classes. Thus, as soon as there are more than two classes, an in-class partition represents a minority class relative to an out of class partition *cf.*, the remaining classes. In practice, properties embedded in the class partitions of large data sets are generally not equally represented. For example, some topics in document repositories are less frequent than others. Likewise, fault (abnormal) conditions in network (medical) data sets typically occur at much lower frequency than the 'typical' behaviour. In this work, we take the view that a single holistic approach should be taken to addressing the issues of large data sets, class-imbalance, attribute selection and model simplicity in order to provide the basis for a robust scalable approach to data mining under a supervised learning context.

There have naturally been many developments in data mining / machine learning that address specific components of the above process. For example, from the perspective of attribute selection three generic approaches have been identified [15]: filter, wrapper and embedded. *Filter* methods begin by performing a process of *dimension reduction*[1] before building the classifier relative to the subset of attributes. *Wrapper* methods place this process in an iterative cycle so that dimension reduction can be revisited as a function of classifier performance. Naturally, in both filter and wrapper methods the process of attribute selection remains independent from the process of constructing the classifier. As such, we maintain that the degree of

[1] By 'dimension reduction' we recognize two generic forms for reducing the attribute space as seen by the classification stage: attribute selection or attribute transform. Attribute selection attempts to select a subset of the original attributes using some measure of inter attribute correlation *e.g.*, F-measure, Gini index. Conversely, attribute transforms apply an operator to the original attribute space to transform this to a new coordinate frame such that various orthogonality properties are satisfied *e.g.*, PCA. Naturally, attribute selection maintains an explicit link to the original attribute space, potentially retaining more insight into the application domain as the classifier builds a model relative to the original domain specific attributes.

granularity in the decision making for attribute selection is not as refined as that possible when both attributes and classifier are designed simultaneously, as in the *embedded* paradigm. Generic machine learning techniques supporting the embedded paradigm include decision tree induction (C4.5), maximum-entropy (MaxEnt) classifiers and Genetic Programming (GP). Needless to say, there is nothing to preclude the use of an embedded technique for classification in conjunction with a filter framework for deployment *e.g.*, [39].

Addressing the class-imbalance problem while scaling to large data sets is satisfied in a single step by recognizing that the goal is to construct a *subset* of training exemplars at each training epoch such that: 1) the process of evaluating the fitness function is decoupled from the total number of training exemplars, and 2) the sampling process may change the distribution of class labels from that of the original training partition. The insight of the second observation is that maintaining the original distribution of class labels during subset sampling results in classifiers that are correlated with maximizing performance under the accuracy objective; whereas biasing the sample for equal representation of all classes is correlated with maximizing performance under an area under the curve (AUC) objective [41]. Naturally, a bias in classifier performance relative to an AUC metric is considered more beneficial than assuming performance relative to the accuracy metric [10]. That this appears through adopting a sampling heuristic as opposed to actually estimating the AUC metric directly is an added advantage [41].[2]

In this work we maintain that the combination of competitive and cooperative coevolutionary mechanisms with GP provides a rich repertoire for supporting the above requirements within a single framework. Thus, by assuming GP as the basis for constructing solutions we already have an *embedded* as opposed to wrapper or filter based schemes for building classifiers. This means that both the classifier and supporting set of attributes are identified simultaneously. Conversely, in order to increase the *transparency / simplicity* of solutions we maintain that support for problem *decomposition / modularity* must be provided. In the case of this work, this is achieved through cooperative coevolution under a model of symbiosis. Moreover, there has been a sizeable amount of research in coevolutionary approaches to active learning for scaling Evolutionary Computation to domains with many exemplars. In this work a competitive coevolutionary relationship is assumed between the subset of training exemplars and the classifiers. The resulting combination of these features provides the basis for addressing all the above requirements simultaneously; hereafter referred to as Symbiotic Bid-Based GP or SBB.

Section 5 details how the popular LIBSVM non-linear SVM implementation was deployed under a wrapper methodology. In doing so, an investigation of the significance of attribute support in model complexity and classifier performance is facilitated. Specifically, in considering classification performance alone, the non-linear SVM consistently provides the more accurate solutions. However, the attribute subspaces on which SVM models are based can be orders of magnitude larger than those supporting SBB solutions, while only providing a 1 to 5 percent improvement in accuracy. Moreover, whereas SVM classification performance more or less improves as a monotonic function of the number of attributes, the

[2] Combining a simple subsampling heuristic with estimation of an AUC style fitness function may provide additional reinforcement for this tendency [8].

complexity of SVM models does not. Instead, there is a specific 'sweet spot' for SVM model complexity, after which increases or decreases in attribute count result in significant increases in model complexity. Moreover, expressing SBB and SVM complexity in terms of an equivalent metric demonstrates that the simplicity of non-linear SVM solutions rarely approaches that of SBB.

The paper now develops by introducing the background material for competitive and cooperative coevolution. Specifically, competitive coevolution provides the basis for scaling GP to large data sets whereas cooperative coevolution provides the metaphor for problem decomposition (Section 2). In so doing, we make clear the various design decisions taken to support SBB. Next the symbiotic approach to GP teaming (SBB) is described, thus designing subspace classifiers under an embedded paradigm (Section 3). The methodology adopted for benchmarking is then established (Section 4), before presenting the formulation of the complexity metric and results of the empirical study (Section 5). Finally, a review of related work is made with conclusions and recommendations in Sections 6 and 7 respectively.

2 Background

2.1 Pareto Competitive Coevolution in Training Subset Selection

A Pareto competitive coevolutionary framework will be assumed for defining the interaction between training exemplar subset and candidate classifiers [12,36,7]. Assuming such a framework will provide the basis for defining which training exemplars and candidate solutions get to 'survive' between generations (Section 3.4.2). The following establishes the supporting definitions for the Pareto framework and highlights the resulting design decisions made in the case of SBB.

The outcome of applying two candidate solutions to a common set of training exemplars (points) can be defined in terms of a pair of vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$. Vector $\mathbf{v}_1$ is said to ***dominate*** vector $\mathbf{v}_2$ if the following dominance relation holds:

$$dom(\mathbf{v}_1, \mathbf{v}_2) \Leftrightarrow \forall q : \mathbf{v}_1[q] \geq \mathbf{v}_2[q] \wedge \exists q : \mathbf{v}_1[q] > \mathbf{v}[q] \tag{1}$$

where q indexes vector dimensions (objectives) *cf.*, the subset of training exemplars or 'points'. Individuals with a dominance vector satisfying the above condition are said to be part of the Pareto front, $\mathcal{F}(\cdot)$, or the non-dominated set. Members of the front have the property that favouring one individual over another requires a tradeoff in objective values, implying that some training exemplars are more important than others.

Given the ith candidate solution (in the case of SBB, a team) m_i and the kth point (from the sample of training exemplars) p_k, a domain specific reward function $G(m_i, p_k)$ returns the outcome of applying candidate solution m_i to point p_k. Higher outcomes are assumed to be better. In the case of M_{size} candidate solutions, objective $M_{size} \cdot i + j$ of point p_k is defined by:

$$\begin{cases} 1 \text{ if } G(m_i, p_k) > G(m_j, p_k) \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

for $1 \leq i, j \leq M_{size}$. If this objective is equal to 1, then point p_k is said to ***distinguish*** between candidate solutions m_i and m_j [12,36]. Pareto-coevolution therefore

searches for candidate solutions that achieve high outcomes and points that serve as informative evaluators.

The initial interest in Pareto-based competitive coevolutionary frameworks came from the ability to guarantee monotonic progress towards solutions [7]. However, it was necessary to assume an infinite memory mechanism (archiving) in order to support the monotonic progress i.e., an infeasible computational overhead associated with archive maintenance. Indeed recent studies have demonstrated that even assuming the more efficient LAPCA multi-layer algorithm (*e.g.*, [7]) for archive maintenance still results in a significant computational overhead when compared to an alternative host–parasite coevolutionary algorithm [4]. Instead the approach adopted here for the proposed SBB algorithm separates the population into two layers only (dominated and non-dominated). This places more emphasis on selection within a layer with competitive fitness sharing [38] used to explicitly enforce a bias in favour of unique behaviour (Section 3.4.2).

In assuming a dual mechanism for rewarding diversity – Pareto dominance and competitive fitness sharing – we recognize that comparatively weak overlapping behaviour may be sufficient for legitimately populating the Pareto front [34], resulting in the majority of candidate solutions lying on the Pareto front. The motivation for introducing an explicitly cooperative mechanism in conjunction with the competitive Pareto formulation is to provide a more effective mechanism for evolving teams of individuals with non-overlapping objective values.

2.2 Symbiosis as a metaphor for Cooperative Coevolution

Cooperative models of coevolution assume explicit support for a divide and conquer approach to collaborative problem solving, thus multiple individuals participate to provide a solution. From the perspective of evolving good embedded classifiers, we are now in a position to associate subsets of attributes with individual team members. Thus, different team members potentially respond to different subsets of the overall task. Any model of coevolution needs to address three basic design decisions: 1) how to identify individuals who will cooperate, 2) how to vary team and team member (collaborator) content, and 3) how to assign credit for the respective contributions to a group of individuals comprising a team. In the following, a case is made for assuming Symbiosis as a metaphor for Cooperative Coevolution.[3]

Symbiosis is a biological process that recognizes two or more initially independent individuals as entering into a 'partnership' [29]. The degree of competition versus cooperation defining the partnership will vary as a function of environmental factors. The most frequently studied forms of symbiosis take one of two forms: Ectosymbiosis or Endosymbiosis. Ectosymbiosis assumes that the individuals participating in a relationship do not require a host. Conversely, under *endosymbiosis* one of the individuals represents a host for others. In this work we assume the abstract model *endosymbiosis* proposed by Maynard Smith consisting of three basic stages [30]: i) The coexistence of two or more candidate species; ii) A compartmentalization of a subset of individuals from the different species within a host

[3] Section 6.2 provides a synopsis of previous / current research in cooperative coevolution in general.

entity, and iii) The emergence of a hierarchical process of replication. The latter point implies that hosts reproduce as a single process i.e., without requiring the independent replication of symbionts that were originally required to form the host compartment; thus a higher level of organization is established.

From the specific context of providing a framework for cooperative coevolution, such a model of symbiosis addresses the aforementioned three basic design questions as follows:

1. Identifying candidate individuals for potential cooperative association: is a function of the host compartment. Assuming that each host is represented as a variable length list of indexes, then a 'team' is synonymous with host compartment and the list represents a subset of the available individuals for combining as a team. The natural implication is that there will be a team (host) population and an independent population of potential collaborators (symbionts).
2. Varying team and collaborator content: is addressed by assuming a hierarchy of variation operators. Operators applied to the host compartment (team) perform a combinatorial search for good candidates for cooperation. Operators applied to symbionts (collaborators) maintain symbiont diversity, but are initiated when a new team (host compartment) is introduced, thus avoiding disruption to current teams.
3. Assigning credit for the respective contributions to team versus individual fitness: is essentially performed at the level of the team (host). The underlying goal of symbionts is to survive i.e., continue to be indexed by the hosts. Natural selection will ensure turnover in team complement i.e., selection pressure at the team level promotes phenotypic change in the symbionts [24].

A key point of this abstracted model for symbiosis is that an explicitly serial relationship exists between a population of individuals representing candidate teams (host compartments) and a second population of candidate collaborators (symbionts). Fitness evaluation is only performed at the level of hosts. Variation operators are hierarchical, beginning at a host and propagating down to symbionts. Naturally, this only represents one possible scheme for mapping a generic metaphor for symbiosis into a model for cooperative coevolution (see [18] for a recent survey). The following section will develop this model in detail within the context of evolving teams of GP symbionts.

3 Symbiotic Bid-Based Genetic Programming framework

This work makes widespread use of coevolution to integrate different components of the overall framework, Figure 1. Specifically, candidate solutions are defined across two populations – teams (M) and learners (L) – through a symbiotic relation. The team population conducts a combinatorial search to resolve 'who' appears in each team, whereas the learner population details the corresponding programs and class label. Conversely, competitive coevolution defines the interaction between a subset of training exemplars – as defined by the point population (P) – and the team population. As such there are three distinct search processes represented by the three unique populations:

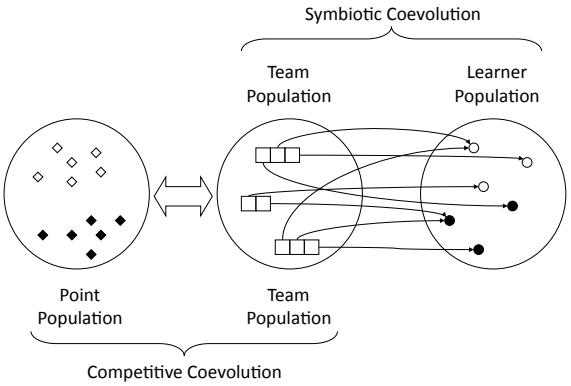


Fig. 1 Architecture of Symbiotic Bid-based GP: Point and team populations are linked via competitive coevolution. Team and learner populations are linked via symbiotic (cooperative) coevolution. Fitness sharing between individuals in the team population defines a second competitive relationship.

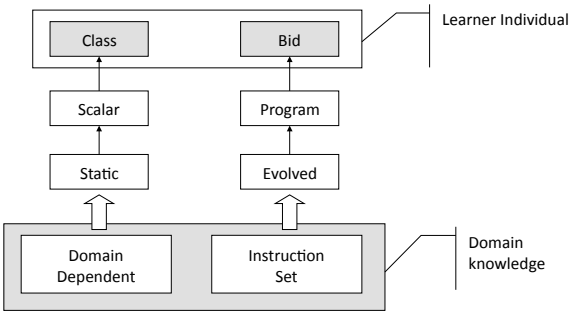


Fig. 2 Learner individual as per bid-based GP. As per canonical GP domain knowledge establishes the instruction set, however, the evolved program provides a *bid* value. Each learner individual is associated with a single class label. At least two learners with dissimilar class labels are necessary to provide a non-trival team.

1. *Points* (P^t): Scaling to potentially large data sets is established through a competitive coevolutionary relationship, Section 2.1. That is to say, a subset of training exemplars P (the 'points'), is identified at each generation, t, against which fitness evaluation of the current population of teams is performed. This is particularly important when each exemplar consists of thousands of attributes.
2. *Learners* (L^t): Each learner associates a 'bidding behaviour' with a single class label (Figure 2). Class labels are assigned during learner creation. This means that there is at least one learner to represent each class. The bidding behaviour, on the other hand, are evolved using GP so that new learners may explore novel behaviour. Fitness evaluation is only performed within the context of a *team* i.e., the host compartment of Symbiosis (Section 2.2). Modification of the learner population is conducted as a top down process relative to the teaming population, thus maintaining context / minimizing disruption to the teaming population.

3. *Teams* (M^t): Teams represent subsets of learners sampled from L^t that together may form complete, non-trivial, solutions. The team population thus searches for combinations of learners that lead to cooperation under the bidding metaphor. Each team must reference at least 1 unique learner per class, but no more than ω learners in total per team i.e., co-operative symbiotic coevolution, Figure 1.

In the following we develop the framework controlling the interaction between each of the three populations relative to a breeder style generational model of evolution summarized by Algorithm 1.

3.1 Representation

Point and team populations index subsets of the training partition and learner population respectively (Figure 1). Individuals from the point population, $p_k \in P^t$, are each limited to indexing a single exemplar from the set of exemplars defined by the original training partition, T, or $\forall p_k \in P^t : p_k \in \{0, ...T-1\}$, albeit under the additional constraint that each class label has equal representation in the point population. Such a balanced sampling heuristic has been demonstrated to result in classifiers that are more robust to artifacts of class imbalance than uniform sampling [41]. Fitness evaluation implies that, each member of the team population is evaluated against each member of the point population under a competitive coevolutionary relationship (Section 2.1). Thereafter, at each generation a fixed number of points are replaced, a process driven by the utility of a point in supporting 'distinctions' between teams (Section 2.1).

In the case of the team population, M^t, each member, m_i, identifies a *subset* of learners, Figure 1, with a minimum of 2 and a maximum of ω learner indexes per (team) individual. Thus, a variable length representation is assumed, where this provides the basis for discovering the level of complexity necessary to solve a task. In both the point and team population, the population size is predefined. However, learner population size may *increase* under the action of **GenTeam**, Section 3.4.1, or *decrease* as learners 'die', Section 3.4.2.

Individuals from the learner population assume the structure of bid-based GP (Figure 2) [25]. Thus, each learner is defined in terms of the tuple $\langle c, b \rangle$; where c is a scalar class label the range of which is defined by the task domain – or for a C class problem $c \in \{0, ..., C-1\}$ – and b is the 'bidding behaviour' as defined by the learner's program. The goal of such a bidding behaviour is to establish the context – i.e., partition of the data – for bidding 'high' or 'low'. When a learner bids sufficiently high, relative to other members of the team, it wins the right to suggest its corresponding class label, c. A learner therefore must act in conjunction with other learners (see Team Evaluation, Section 3.2), hence the role of the team population in explicitly defining a subset of learners to cooperate with *cf.*, a symbiotic relationship as reviewed in Section 2.2.

Naturally, any GP representation could be assumed for evolving a bidding behaviour, b. Without loss of generality we assume linear GP [2]; thus individuals are composed from a predefined set of instructions under a simple register level transfer (RLT) language with program state defined over a set of registers $R[x] : x \in [0, ..., maxRegisters - 1]$. The RLT language has a generic form that

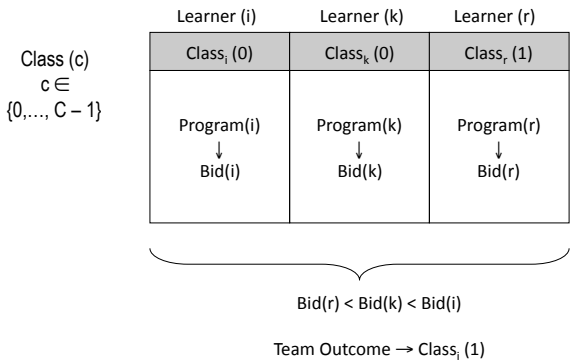


Fig. 3 Resolving the definitive class label from a team of learners. This team is represented by three learners: i, k, r as selected from the learner population. Process begins by presenting an exemplar and running the identified subset of bid-based GP programs, b. The individual with largest bid 'wins' and suggests its corresponding class label, c. Fitness evaluation is conducted relative to these class labels.

follows from the number of arguments an operator takes. Thus, in the case of two argument operators – specifically: $\langle op \rangle \in \{+, -, \times, \div\}$ – then the RLT has the form: $R[x] \leftarrow R[y]\langle op \rangle R[z]$. In the case of one argument operators – specifically: $\langle op \rangle \in \{\cos(\cdot), \exp(\cdot), \log(\cdot)\}$ – then the RLT has the form: $R[x] \leftarrow \langle op \rangle R[y]$. From this generic form, two 'addressing modes' are supported. This results in the last instruction operand being defined purely in terms of the register set, as above, or the last operand being in terms of an index to the attribute space.[4]

3.2 Team Evaluation

Before defining the evolutionary process itself we first introduce the process for evaluating a team under the symbiotic team–learner relation, Figure 3, or establishing the classification performance of the team from the bidding behaviour of learners. The selection of an individual from the team population, $m_i \in M^t$, identifies a corresponding subset of individuals from the learner population, L^t. Evaluation now progresses by first identifying the bid values of each learner associated with a given team. Thus, for all learners in the current team ($l_j \in m_i$), execute their program under the condition defined by the current training exemplar, p_k. In each case the output from each learner's program, b, is normalized to the unit interval for the purposes of establishing a common concept of 'bidding' high or low, or $bid(l_j) = (1 + \exp(-R[0](l_j))^{-1}$; where '$R[0](l_j)$' is the numerical value in register zero of learner j's program after program execution.

With all the bid values on exemplar p_k established for the learners associated with team m_i a scheme is now necessary for defining which of the team's learners gets to suggest the class label. In this work we assume a 'winner-takes-all' policy in which the single learner with maximum bid 'wins' the right to suggest its corresponding class label, Figure 3. Thus, the winning bid is identified as

[4] For example, $R[x] \leftarrow R[y]\langle op \rangle IP(z)$ where $IP(z) : z \in \{0, ..., A-1\}$ denotes an attribute space index associated with the data set.

$l^* = \arg_{l_j \in m_i} \max[bid(l_j)]$. Adopting such an approach is deemed more likely to make clear the role played by a specific learner (within a specific team). Finally, the result from the **Evaluate** function at Step 10 of Algorithm 1 is a binary outcome, $G(\cdot, \cdot)$, reflecting whether the class label from the winning learner matches that of the corresponding desired class label, d_k, as defined by the training partition, or:

$$G(m_i, p_k) = \begin{cases} 1, \text{ IF } c.(l^*) == d_k \\ 0, \text{ otherwise} \end{cases} \quad (3)$$

where $c.(l^*)$ denotes the class label, c, suggested by the learner returning the winning bid (l^*) from the team currently under evaluation (m_i) relative to point p_k. It is this function that supports the definition of distinctions introduced within the context of competitive coevolution, Section 2.1.

Algorithm 1 Overview of the SBB training algorithm. t is the generation index, thus P^t, M^t, L^t are the point, team and learner populations at generation t.

```
1: procedure TRAIN
2:     t = 0
3:     P^t = INITPOINTS()                                  ▷ Section 3.3
4:     (M^t, L^t) = INITTEAMS()
5:     while t ≤ t_max do                                  ▷ Main loop.
6:         P^t = GENPOINTS(P^t)                            ▷ Section 3.4.1
7:         (M^t, L^t) = GENTEAMS(M^t, L^t)
8:         for all m_i ∈ M^t do
9:             for all p_k ∈ P^t do
10:                EVALUATE(m_i, p_k)                      ▷ Section 3.2.
11:            end for
12:        end for
13:        P^{t+1} = SELPOINTS(P^t)                        ▷ Section 3.4.2
14:        (M^{t+1}, L^{t+1}) = SELTEAMS(M^t, L^t)
15:        t = t + 1
16:    end while
17:    return BEST(M^t)                                    ▷ Section 3.5
18: end procedure
```

3.3 Initialization

Training of the three populations begins with their initialization, lines 3 and 4 of Algorithm 1. Point initialization, line 3, generates each point by first *selecting a class label* with uniform probability and then, if possible, selecting a training exemplar matching that class label with uniform probability.[5] The uniform sampling of class labels provides the bias towards building a point population with equal representation of training exemplars from each class. The process continues until $P_{size} - P_{gap}$ unique points are created, with tests performed to ensure that a point only appears once. The other P_{gap} points are added using the same process once the main loop of the breeder model of evolution begins in order to mirror the process of removing P_{gap} points per generation.

[5] Should all points for a class already be present in the point population, the class label is reselected.

Team and learner initialization, line 4, produces each of the $M_{size} - M_{gap}$ initial teams by first selecting two different classes with uniform probability and then associating a new arbitrary bidding program with each of those labels, thus each initial team represents a minimally complex solution. Moreover, at this stage there is no learner appearing in more than one team. This will gradually change under the action of the variation operators. In addition, the variable length representation assumed by the team population will let team size and composition for each member of the team population evolve independently over time. As per the point population the other $M_{size} - M_{gap}$ teams are added once the main loop of the breeder model of evolution begins.

3.4 Breeding

The 'main loop' of Algorithm 1 orchestrates the breeding cycle by repeating three steps until the maximum generation limit is encountered:

1. Generate points and teams: Routines **GenPoints** and **GenTeams** introduce a fixed number of points and teams at each generation (P_{gap} and M_{gap} respectively). This can also imply that a variable number of new learners are introduced into the learner population.
2. Evaluate teams over all points: The **Evaluate** routine of Section 3.2 is called to return a vector of binary 'true / false' outcomes from applying each team to the subset of the training partition defined by the point population.
3. Remove points and teams: Routines **SelPoints** and **SelTeams** deterministically delete a fixed number of points and teams per generation (P_{gap} and M_{gap} respectively). Tests are also applied to delete any learner that are no longer indexed by teams.

In the following we detail the process for generating and removing points / teams. Section 3.2 having introduced the process for evaluating point–team pairs.

3.4.1 Point and team generation

Step 6 Algorithm 1 adds P_{gap} points to the point population, where this mirrors the removal of points after the inner-loop of fitness evaluation, Step 10. Thus, the evaluation of teams over points is always conducted over P_{size} points and M_{size} teams. The process assumed for adding points at each generation – **GenPoints** – is the same as that described for point initialization (Section 3.3). As such there is no concept of a parent–offspring relationship. Specifically, points are merely indexes to exemplars in a larger training partition in which there is no ordering enforced between exemplars. To do so would require the application of some form of similarity metric to the entire training partition, a potentially expensive process. Without such distance information, 'offspring selection' therefore takes the form of a purely stochastic process, albeit under the bias of equal class representation (Section 3.3).

Conversely, the process for team generation – **GenTeams** – is able to make use of genotypic information. Moreover, the action of variation operators is hierarchical, resulting in the modification of both teams and learners through the following three stage process:

Stage 1 – Select parents: A pair of parents are selected from the team population with uniform probability (m_i, m_j) from the current population complement, M^t.

Stage 2 – Team level variation (crossover): Any genes (*cf.*, learner references) common to both parents are copied to both offspring (m'_i, m'_j); defining a pair of 'twins'. The inheritance of such common material is assumed to provide the basis for the exploitive component of the search process. Any remaining 'uncommon' genetic material from the parents is assigned to each offspring with uniform probability, or the explorative component of the search conducted at the team 'level'. Naturally, tests are enforced during the explorative part of offspring construction to ensure that the resulting teams satisfy the minimum (maximum) gene complement limit of 2 (ω).

Stage 3 – Learner level variation (mutation): The following processes are applied to each offspring (m'_i, m'_j) independently: 1) shuffle the order of genes (*cf.*, learner references) within an offspring. This has no impact on the team, but varies the order in which the following mutation operators are applied. 2) Sequentially test each learner index for removal with probability, p_d so long as the team learner complement is at least two. 3) Sequentially test each learner indexed by the offspring for *duplication* with probability p_a so long as the team learner complement does not exceed ω. Denote the duplicated learner $l' = \langle c', b' \rangle$. With probability p_c a 'class mutation' operator changes c' to a different class label. The content of the duplicated individual's bidding program, b', is then subject to variation through the deployment of representation specific mutation operators. Given the linear GP representation assumed in this work (Section 3.1) four instruction-wise operators are utilized: add, delete, swap and mutate. Instruction add or delete insert or remove instructions with uniform probability. Instruction swap selects and interchanges two instructions, thus varying instruction order. Instruction mutate flips the bit value of a current instruction.

The process is complete when up to $M_{gap}/2$ pairs of (team) parents have been created i.e., M_{gap} offspring, so bringing the total team population compliment back up to M_{size}. Note that the process of introducing learner variation in 'Stage 3' implies that only through team offspring may new learner individuals appear. At following generations the action of the 'team level variation' operator might result in other (offspring) teams inheriting (indexes to) the more recently created learners.

3.4.2 Point and team removal

Section 3.2 defines the process for building the pairwise matrix of outcomes, $G(m_i, p_k)$. This provides the basis for estimating a $M_{size} \times M_{size}$ distinction vector at the centre of the competitive coevolutionary frameworks reviewed in Section 2.1. Thus, from the distinction vector, a non-dominated Pareto front of points is identified relative to the current point population, denoted by the set $\mathcal{F}(P^t)$. The remaining dominated points being denoted by the set $\mathcal{D}(P^t)$.

The following process is assumed for selecting the $P_{size} - P_{gap}$ points that 'survive' into the next generation, or **SelPoints**. One of three distinct scenarios exists depending on the relative size of the Pareto front to the number of individuals surviving per generation:

$|\mathcal{F}(P^t)| == P_{size} - P_{gap}$: Implies that there are as many points in the Pareto front as points that can survive per generation. The next point population is formed by the content of $\mathcal{F}(P^t)$ alone.

$|\mathcal{F}(P^t)| > P_{size} - P_{gap}$: Implies that there are *more* points in the Pareto front than can survive to the next generation. A heuristic therefore needs defining to provide an additional basis for prioritizing the non-dominated solutions. Specifically, competitive fitness sharing [38] is used to discount the significance of a distinction based on the relative uniqueness (of a distinction), or:

$$\sum_i \frac{d_k[i]}{1 + N_i} \tag{4}$$

where i iterates over all the distinction vector entries, $d_k[i]$ is the ith entry in p_k's distinction vector, and N_i counts the number of points in $\mathcal{F}(P^t)$ that make the same distinction as the ith entry. Normalization by the number of points 'N_i' that make the same distinction i in Eq. 4 implies that individuals with more unique distinctions receive more weight. The worst P_{gap} points are then discarded.[6]

$|\mathcal{F}(P^t)| < P_{size} - P_{gap}$: Implies that more points survive to the next generation than exist in the Pareto front $\mathcal{F}(P^t)$. Thus, all points from the Pareto front appear at the next generation along with *some* dominated points $\mathcal{D}(P^t)$. Equation (4) is again utilized to provide the ranking (this time of $\mathcal{D}(P^t)$) from which dominated points are sampled until the partition of points surviving to the next generation is full.[7]

Naturally, the process for team selection – **SelTeams** – follows an analogous process by identifying the $|M_{size} - M_{gap}|$ teams surviving between consecutive generations. In this case a team's outcomes over all points in P^t is used to calculate its competitive fitness sharing score and its membership in $\mathcal{F}(M^t)$ or $\mathcal{D}(M^t)$. Team selection considers outcomes against all the points involved in the current generation (including the ones selected against) as this is viewed as providing a more informative measure of performance compared to considering only the selected points. Finally, in the case of the learner population, L^t, each individual is tested to confirm that it still receives at least one index from the team population. Any learners with no team index are 'killed'. Thus, the learner population size floats between $2 \times M_{size}$ and $\omega \times M_{size}$.

3.5 Concluding comments

Since evolution results in a population containing multiple teams, one 'champion' must be selected as the final best solution, line 17 of Algorithm 1. Here, the team yielding the highest class-wise detection on the entire training data set is returned, where this metric is defined in Section 4.4.

[6] Where this represents all the dominated points (should any exist) and enough of the non-dominated points to complete P_{gap}. Prioritization of the latter as established by the ranking of non-dominated points as established by the sharing function.

[7] In this case N_i counts the number of points in P^t – as opposed to $\mathcal{F}(P^t)$ – that make the same distinction as the ith entry.

In summary, a coevolutionary algorithm has been designed in which competitive coevolution between points and teams provides the basis for scaling the algorithm to large training partitions; with (team) fitness evaluation being conducted over an evolving subset of exemplars identified by the point population. Solutions are specified through a combinatorial search conducted to establish the relevant composition of a team of bid-based GP individuals. A symbiotic relation between a population of teams and programs (learners) separates the process of combinatorial search and program optimization; whereas the bid-based GP methodology establishes which *single* individual is responsible for providing each class label *within* a team.

4 Evaluation Methodology

The selection of a cross-section of benchmarking data sets is described in Section 4.1, and a summary of the baseline filter / wrapper-based SVM model is given in Section 4.2. Parameterization of the SBB and SVM models is briefly discussed (Section 4.3), and the metrics deployed for evaluating post training classification performance presented (Section 4.4). Classifier deployment in all cases assumes that the cascading of multiple independent classifiers is not appropriate. Instead either a voting policy must be assumed (as in the SVM) or, as in the case of SBB, a single solution produces the labels for all classes.[8] For convenience, all the data sets and source code for SBB and SVM models are available from a single URL[9] as the original SVM distribution was modified to provide wider reporting of performance metrics; whereas data sets although publicly available were preprocessed as described below.

4.1 Data Sets

This work uses data sets from the domains of document analysis and image processing, Table 1. The image processing 'Multifeature' and 'Gisette' data sets [1] both pertain to the recognition of handwritten digits. No pre-processing was applied in either case. The Multifeature data set is a 10 class problem – representing digits 0 to 9 – with each class equally represented. Gisette is a binary classification problem in which the task is to distinguish between '4' and '9' ((55% (45%) of the exemplars are in-class (out-class)). Moreover, Gisette has the additional property that half of the attributes are 'probes,' thus redundant from the perspective of building an appropriate classification model (the probe attributes are not identified).

In the case of the document analysis domain, three binary classification problems were constructed from the UCI Bag-of-words data set [1] which contains documents from several repositories. The documents themselves are unlabeled but it is known from which repository each is sourced. Thus, we first combine documents from the NIPS, Enron and New York Times (NYT) repositories and use the resulting union of exemplars to construct attributes with respect to the words

[8] As discussed in related work, Section 6.1, this does not preclude the coevolution of hierarchical models, but this is outside the scope of this paper.

[9] http://web.cs.dal.ca/~mheywood/Code/SBB

Table 1 Data sets used in the comparison.

Dataset	Exemplar counts		Attribute Count
	Training	Test	
Handwritten character recognition			
Multifeature	1,510	490	649
Gisette	6,000	1,000	5,000
Document Classification: Bag-of-words			
NIPS	7,000	3,500	12,419
Enron	7,000	3,500	28,102
NY Times	7,000	3,500	102,660

found across *all three* repositories. To construct the NIPS problem, we use the same union of exemplars but restrict the words to those found only in the original NIPS repository; the documents sourced from the NIPS repository were then labeled as in-class and all others as out-of-class. The construction of the Enron and NYT problems was performed in an analogous manner. Hence, all three problems contain the same number of exemplars, Table 1, while their attribute counts vary. This resulted in the largest attribute spaces deployed during the ensuing performance evaluation. Class representation was also generally unbalanced with in-class representation at 14%, 29% and 57% respectively for NIPS, Enron and NYT.

4.2 Non-linear SVM with F-score Filter / Wrapper

Large margin methods such as SVM, Maximum Entropy Classifiers (MaxEnt) and AdaBoost represent a popular class of supervised learning algorithms deployed under the classification domain. Such models employ a convex optimization process during training guaranteeing predictable convergence properties. Moreover, the inclusion of regularization terms may improve robustness, avoiding overtraining, and provide an opportunity to identify subspaces over the original set of attributes under an embedded model of attribute selection (as opposed to wrapper or filter models [22]). A recent study by Haffner [16] – with respect to linear large margin methods – established a common framework for describing the properties of SVM, MaxEnt and AdaBoost, particularly with respect to the large attribute space problem. The study considered the computational resources necessary to support training and the resulting classifier performance under problem domains with large attribute counts. Moreover, the utility of (filter based) dimension reduction methods such as PCA or attribute selection has repeatedly proved effective at reducing the dimensionality sufficiently to make the deployment of non-linear SVM models practical [6,17]. Indeed the preprocessing step needs only be relatively simple – thus computationally efficient – in order to be effective in practice [14]. In this study, we expand on a filter based non-linear SVM model – placed third in the original ANIPS[10] feature selection challenge [6].

Specifically, the F-score based ranking of attributes from the original study is retained to rank attributes, but deployed within the context of a wrapper method-

[10] We distinguish between the 'NIPS' partition of the document repository from the 'Advances in Neural Information Processing (ANIPS)' conference venue.

ology, thus enabling us to iterate on classifier optimization and attribute selection. Such an approach provides the opportunity to investigate the interaction between attribute count, SVM accuracy, and SVM complexity. The F-score approach provides a measure for the linear discrimination of an attribute and is distinct from the classifier performance metric of the same name [6]. In the context of an attribute filter, F-score estimates the discriminatory capability of an attribute from the negative and positive class exemplars. The larger the F-score the greater the discrimination. Letting $\hat{x}_i$ denote the average of the ith attribute as a whole and $\hat{x}_i^{(+)}, \hat{x}_i^{(-)}$ denote the average of the ith attribute but partitioned into positive and negative exemplar instances alone, then the F-score for attribute i is defined by:

$$F(i) = \frac{\left(\hat{x}_i^{(+)} - \hat{x}_i\right)^2 + \left(\hat{x}_i^{(-)} - \hat{x}_i\right)^2}{\frac{1}{T^{(+)}-1}\sum_{k=1}^{T^{(+)}}\left(x_{k,i}^{(+)} - \hat{x}_i^{(+)}\right)^2 + \frac{1}{T^{(-)}-1}\sum_{k=1}^{T^{(-)}}\left(x_{k,i}^{(-)} - \hat{x}_i^{(-)}\right)^2} \tag{5}$$

where $T^{(+)}$ and $T^{(-)}$ are the number of positive and negative exemplars in the training partition; $x_{k,i}^{(+)}$ and $x_{k,i}^{(-)}$ represent the kth positive and negative exemplar of attribute i respectively.[11]

After applying the F-score metric a relative ranking appears of the degree of (linear) discrimination each attribute provides. At this point a threshold needs applying with all attributes below the threshold being ignored. In the original deployment Chen and Lin retained a validation sample and iteratively increased the threshold until the SVM classifier performance plateaued [6]. In this work, we retain each SVM model resulting from each increment in the attribute threshold. This enables us to plot a performance curve for attribute support versus classification performance. Given the comparative nature of the benchmarking study, thresholds are therefore selected to span the range of attributes identified by SBB. Thereafter the above combination of F-score attribute ranking and SVM classifier will be referred to as 'F+SVM'.

4.3 Configuration

With the exception of the larger team sizes, the SBB parameterization follows that of the original work [26], Table 2. Modifications to the original SBB code distribution concentrated on support for indirect indexing (of data files), implying that an 'associative array' be used to ensure that only data corresponding to the indexed attributes be explicitly retrieved during training. Naturally, since evolution is a stochastic process, 50 runs of the SBB algorithm were performed for Multifeature, Gisette and NIPS. The computational overhead of the larger attribute spaces implied limits of 30 runs in the case of Enron, 20 runs in the case of NYT. Moreover, performance under the training partition was used to determine the top 50% of SBB solutions deployed under test. Given the stochastic nature of credit assignment in GP this is considered to be of more significance – and computationally more feasible – than parameter refinement over multiple runs (the latter being appropriate for the deterministic SVM (Section 4.2)).

[11] The F-score filter and SVM implimentation is available from: http://www.csie.ntu.edu.tw/~cjlin/

Table 2 SBB algorithm parameterization. Max. Registers and Max. Instructions represent limits applied per learner.

Parameter	Value	Parameter	Value
Point pop. size (P_{size})	90	Team pop. size (M_{size})	90
Max. Generations (t_{max})			30 000
Prob. learner deletion (p_d)	0.1	Prob. learner addition (p_a)	0.1
Prob. class mutation (p_c)	0.1	Max. team size (ω)	100
Point generation gap (P_{gap})	30	Team generation gap (M_{gap})	60
Max. Registers	8	Max. Instructions	96

The original LIBSVM scripts for building the F-score metric were modified for scaling to larger data sets. The LIBSVM distribution was itself modified to perform the cross-validation test of model performance using the same class-wise detection metric as SBB, Section 4.4.

4.4 Post Training Performance Metrics

Performance post training will be initially assessed from the perspective of classification and feature count. In the case of classification performance we make use of detection (sensitivity) as measured class wise, resulting in a multi-class measure of detection. Thus, defining the class specific detection rate as $DR(i) = \frac{tp(i)}{tp(i)+fn(i)}$ where *tp(i)* and *fn(i)* are the true positive and false negative counts under class $i \in \{1, ..., C\}$; leading to the following definition for class-wise detection rate:

$$CW\text{-}detection = \frac{DR(1) + \ldots + DR(C)}{C} \tag{6}$$

Such a measure is independent of the distribution of exemplars per class. Thus, under an imbalanced binary data set in which 95% of the exemplars were out-class, a degenerate classifier might label all exemplars as the out-class and achieve an accuracy of 95%. In contrast, the CW-detection metric would return a sensitivity of 50% (or more generally $1/C$), and identify the degenerate condition. As mentioned above, the CW-detection metric is also used post training under the SVM cross-validation process for model selection and the SBB best team selection process. In both cases, this is with respect to the training partition.

Feature counts will be measured in terms of the number of unique domain attributes utilized in the model. The case of establishing a complexity metric for comparing SVM and SBB solutions will be derived once model specific concepts of complexity have been established (Sections 5.2 and 5.3).

5 Benchmarking Results

Benchmarking results will first be summarized in terms of classification performance using 2-d scatter plots of CW-detection versus attribute count under the test partition, Section 5.1. SBB attribute counts include all the unique attribute indexed by the team as a whole. The underlying goal of the plots is to establish

baselines for discussing attribute subspace and accuracy against which the context of SBB solutions can be assessed. Sections 5.2 and 5.3 will then consider the relative complexity of the resulting solutions.

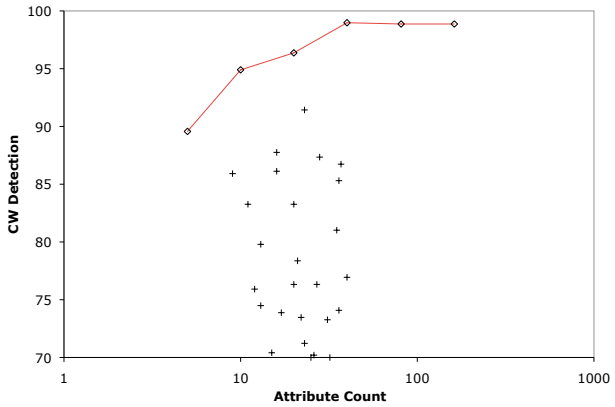
5.1 Classification Performance

Figure 4 establishes the relative performance of the models under the test partition of the character recognition data sets. In the case of Multifeature, subplot (a), SBB consistently results in solutions with 7 to 40 attributes (from a total of 649), but with a wide spread in class-wise detection rate (70% to 90%). Conversely, all but one F+SVM solution provides a CW-detection of 90% or more while requiring the support of 5 to 162 attributes; indeed the F+SVM solutions dominate with respect to CW-detection. Relative the earlier study [9], both SBB and F+SVM are significantly better than C4.5 and MaxEnt results on Multifeature.

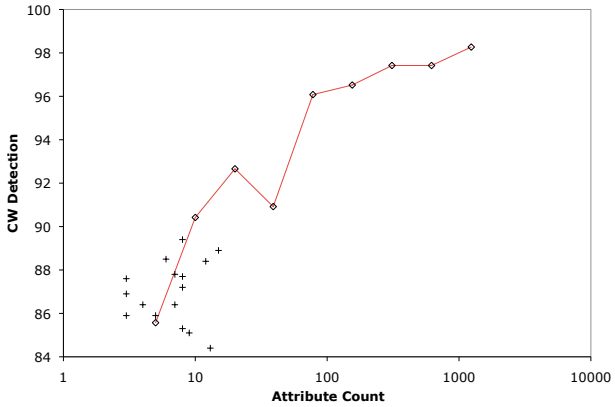
Under the Gisette data set, subplot (b), SBB solutions again use a smaller number of attributes. However, unlike the Multifeature data set, solutions identified under the F+SVM paradigm are no longer able to entirely dominate all SBB solutions; the simplest F+SVM solution is now dominated by multiple SBB solutions. Thus, reducing the number of attributes supporting F+SVM solutions under Gisette has more impact than under Multifeature; whereas the performance of SBB is less sensitive to Gisette than Multifeature. From the earlier study, C4.5 and MaxEnt were able to better SBB in terms of accuracy, but only by incurring significant increases in attribute count [9].

Performance under the Bag-of-words data sets, Figure 5, suggests that (the single) best case instances of classification are still generally attained under the F+SVM paradigm. However, it is also apparent that this requires the support of larger attribute counts; that is *more than* 320, 90 and 80 attributes for NIPS, Enron and NYT respectively. Conversely, SBB generally provides solutions with much lower attribute counts while demonstrating CW-detection rates within 0.2%, 2% and 3% of the best F+SVM results (NIPS, Enron, and NYT respectively). Indeed under both Enron and NYT, F+SVM classification performance breaks down at the levels of attribute counts utilized by all SBB solutions; whereas classification performance under NIPS is limited to a common small range of variation for both algorithms (0.6%), making attribute count the only distinguishing factor. With regards to the earlier study, SBB dominated the results from C4.5 and MaxEnt, with SBB providing either the same accuracy under much lower attribute counts or SBB providing both better classification and smaller attribute counts [9].

The picture of performance as presented so far is limited to an indirect measure of solution complexity. Specifically, Figures 4 and 5 are limited to expressing model complexity in terms of the number of attributes indexed and as such are unable to identify whether reductions in attribute count are balanced by incorporating more complexity in the model itself. Clearly, attribute count alone is not sufficient to meaningfully describe model complexity. Section 5.2 will therefore investigate the issue of SVM and SBB model complexity further using model specific counts of complexity. Section 5.3 makes the case for a common measure of model complexity based on the number of (GP) instructions necessary to estimate (SVM) support vectors.



(a) Multifeature

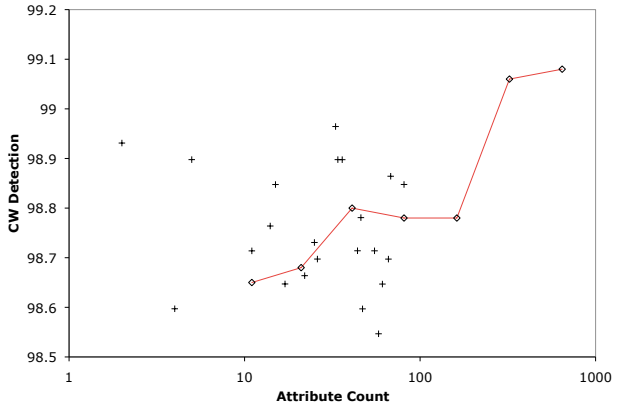


(b) Gisette

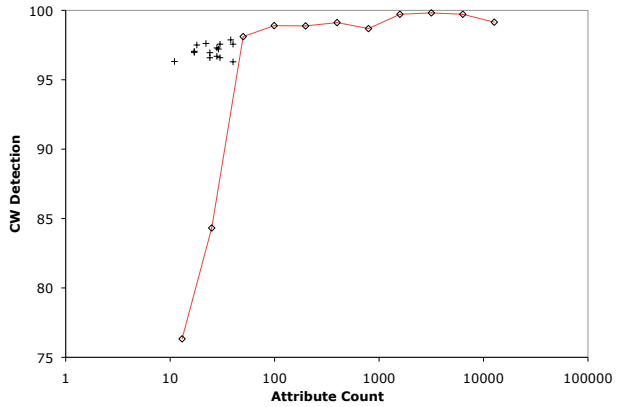
Fig. 4 Test CW-detection (in percent) versus attribute count of SBB (+) and F+SVM (◇ with line). Points towards the upper left hand corner of the plot dominate the performance of points falling to the lower left (much like an ROC curve). SBB solutions are plotted per run and therefore summarize the degree of variation resulting from initialization of the population and the stochastic process of credit assignment. F+SVM solutions are plotted for increasing levels of pruning, as discussed in Section 4.2, yielding a curve.

5.2 Complexity – Model specific

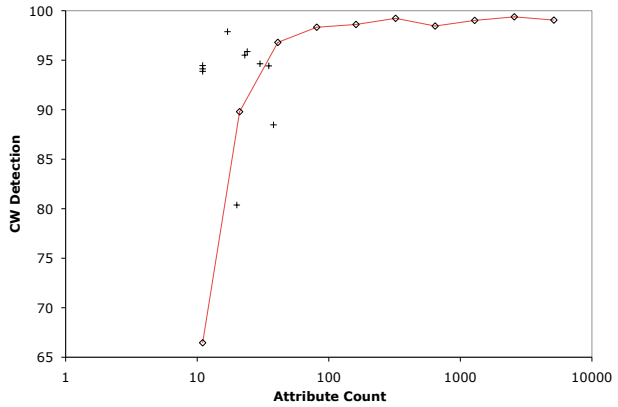
As indicated above, complexity of non-linear models is not necessarily a monotonic function of attribute counts. For example, in the case of the SVM, the number of support vectors might summarize model complexity. Figure 6 plots support vector counts versus corresponding attribute counts for each solution identified under the F+SVM approach. It is now apparent that one of two trends generally result. In the first case, the relationship between model complexity and attribute count appears to be log-linear i.e., linear on a log scale. For example, less complex solutions (cases with a lower support vector count) are associated with the larger attribute counts under Gisette and Multifeature; whereas under NIPS the relationship is reversed but remains linear.



(a) NIPS

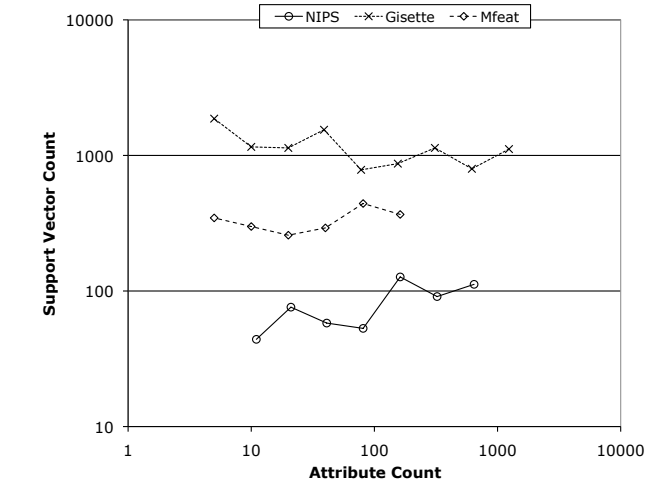


(b) Enron

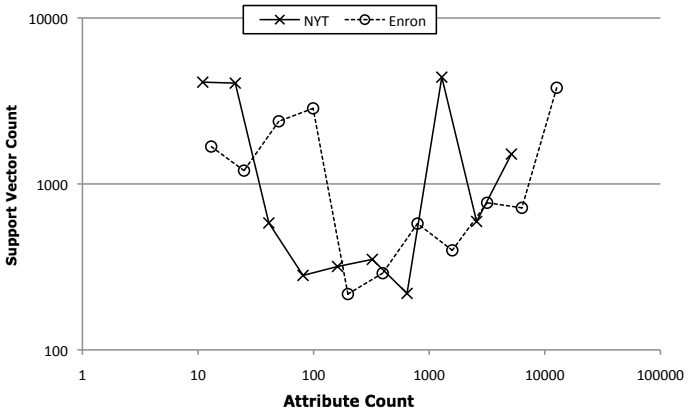


(c) NYT

Fig. 5 Test CW-detection (in percent) versus attribute count of SBB (+) and F+SVM (◇ with line). Points towards the upper left hand corner of the plot dominate the performance of points falling to the lower left (much like an ROC curve). SBB solutions are plotted per run and therefore summarize the degree of variation resulting from initialization of the population and the stochastic process of credit assignment. F+SVM solutions are plotted for increasing levels of pruning, as discussed in Section 4.2, yielding a curve. Note that the x-axis is a logarithmic scale.



(a) linear



(b) non-linear

Fig. 6 F+SVM support vector counts versus attribute counts. Interaction between SV count and Attribute support is either (a) Linear – Multifeature, Gisette, NIPS – or (b) Non-linear – NYT, Enron. Highest attribute counts corresponds to initial F-score selection. Attributes are sampled at incrementally higher F-score thresholds to provide solutions using fewer attributes. Both axes are log scaled.

The second trend of behaviour under the SVM paradigm takes the form of a second order interaction between the number of attributes included and support vector count. Thus, under the remaining data sets SVM model complexity increases towards the lower and higher attribute counts, and is minimized at some 'sweet spot' in between. This is particularly apparent under NYT and Enron.

This trend is mirrored in the CW-detection results, Figures 4 and 5. The data sets with the linear interaction between complexity and attribute count – Multifeature, Gisette and NIPS – retained comparatively strong CW-detection rates as the number of supporting attributes decreased. The two remaining data sets – both of which resulted in a non-linear interaction between attribute count

and support vector count – displayed a knee in the classification–attribute count profile in which classification performance dropped significantly as the number of attributes decreased further.

In the case of SBB solutions, complexity can be expressed at the team level and at the level of individual members comprising a team, Figure 7. At the team level, model complexity is viewed in terms of the number of team members, Figure 7 (a), and the total number of unique attributes indexed across the team, Figure 7 (b); the latter corresponds to the per solution attribute counts on the x-axis in Figures 4 and 5. At the team member level, we consider the number of attributes indexed by individual team members and their instruction count, Figure 7 (c) and (d) respectively. Given the very high incidence of code bloat (introns) in linear GP, common practice is to prune instructions not in the execution path of the output register (structural introns) [2]. Such a practice is followed here.

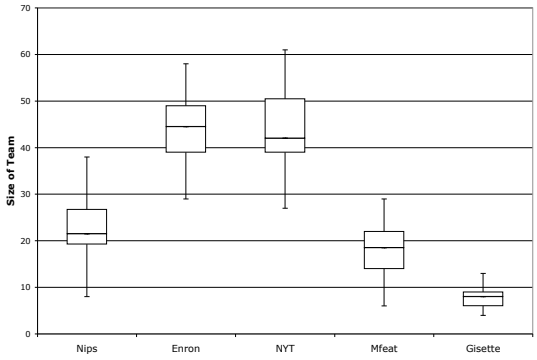
It is now apparent that the pattern recognition data sets, representing smaller attribute spaces, are associated with smaller teams at the team level, Figure 7 (a). Moreover, at the team member level the number of attributes indexed is consistently very small independent of the data set, or typically less than 3 attributes per individual[12], Figure 7 (c). Indeed, only on the character recognition data set did the number of attributes indexed reach 4 to 6 terms (max). The total instruction count per individual is also very low, irrespective of data set, typically ranging from 5 to 9 instructions, Figure 7 (d). Thus, individuals remain simple irrespective of data set. SBB introduces most complexity in terms of the number of team members, Figure 7 (a), or a worst case maximum of 61 in the case of NYT. In general, SBB was able to make most use of subspace decomposition under the bag-of-words data set, with Multi-feature resulting in greatest solution complexity (as measured in terms of instruction and attribute count).

5.3 Complexity – Common metric

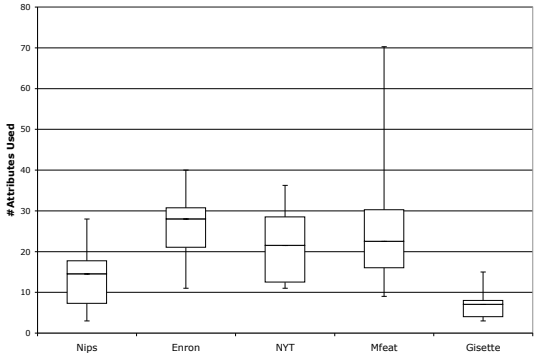
At this point we have been able to qualify typical SBB and SVM solution complexity under model specific metrics, however, a more direct comparison would be useful. On the face of it, this might be difficult to achieve because SVM complexity is measured in terms of support vectors and SBB complexity is a function of the number of team members and instruction counts.

To meaningfully compare SBB and SVM solution complexities a common metric should be identified. Specifically, SVM solution complexity will again take the form of support vector count, where each support vector applies a kernel to a vector product of attributes. Each support vector requires a vector product, or from an SBB perspective, as many register–attribute multiplications as attributes. A common measure of complexity would therefore express the number of vector products that the total instruction count of a SBB team may support. Thus, given an SVM solution and the number of attributes supporting this solution (as identified under the F-score metric), we establish the number of support vectors that the

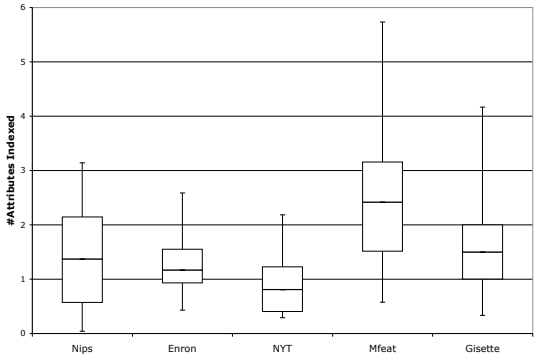
[12] Attribute counts of zero appear and are indicative of members within a team establishing a constant bid value against which other team members have learnt to establish their bidding policy. Such a characteristic might enable further post training simplification of the team composition but was not considered here.



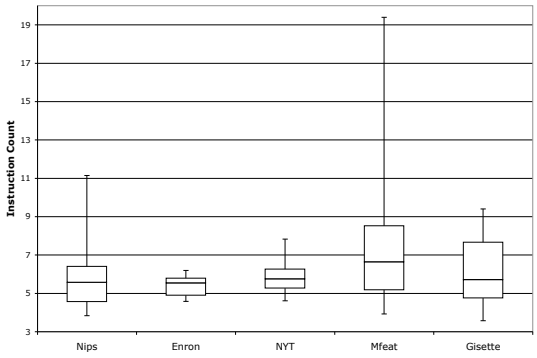
(a) Number of individuals in a team



(b) Number of unique attributes indexed across all team members



(c) Number of unique attributes indexed by each individual



(d) Individual instruction counts

Fig. 7 Quartile summary of SBB composition (max-min whiskers). Subplots (a) and (b) denote team-wise complexity, subplots (c) and (d) individual-wise complexity.

SBB team-wise instruction count would be able to define. Specifically, given an SVM solution, the complexity of a comparable SBB solution is defined as:

$$\frac{Total\_SBB\_Instr}{SVM(\#attributes)} \tag{7}$$

where *Total_SBB_Instr* is the total team-wise instruction count, estimated by assuming a worst case bound of the 3rd quartile instruction count (Figure 7 (d)) and 3rd quartile team size (Figure 7 (a)) or $Total\_SBB\_Instr(\text{Enron}) = 5.8 \times 49 \rightarrow 284.2$ instructions; and $SVM(\#attributes)$ is the attribute count for *each* SVM solution under the *same* data set. For example, in the case of the Enron data set the SVM solution at the knee of the CW-classification curve consists of 2,385 support vectors and 50 attributes (Figure 5.(b)). SBB solutions utilize a worst case instruction count of 284.2 instructions or a total of 5.68 support vectors under such an attribute space. Thus, for each SVM attribute–support vector pair we estimate a corresponding SBB capability to calculate support vectors under the total (worst case) data set specific SBB instruction count.

Figure 8 plots the resulting SBB versus SVM solution complexities given the above common measure of 'support vector count'. Given the wide variation in SVM complexity (i.e., support vector count, Figure 6), a logarithmic scale is used on the x-axis. Equal complexity is indicated by the $y{=}x$ curve; thus points towards the right of the curve indicate increasing complexity of SVM relative to SBB solutions. It is clear that SVM solutions are generally at least an order of magnitude more complex i.e., to the right of the $y{=}0.1x$ curve; whereas no SVM solution is simpler than the SBB cases. Given the higher than average SBB model complexities assumed in Eq. 7 (3rd quartile counts were used) this is a strong indication of a complexity–accuracy trade off between the two approaches. Moreover, given that the F+SVM methodology provided the best case solutions under the original ANIPS competition for the character recognition domain, and was within 1% of the winning solution under the ANIPS Bag-of-words data set we are not attempting to cripple the straw man. SBB solutions clearly represent the choice approach under the larger attribute spaces presented by the Bag-of-words domain, and return solutions with one to two orders of magnitude simpler under the character recognition domain.

6 Related Work

6.1 Evolutionary attribute selection

As indicated in the introduction, the GP paradigm implicitly takes an embedded approach to model building with individuals required to explicitly specify both program and attributes. Various previous works have therefore considered GP and GA for attribute selection / feature construction [20, 39, 44]; albeit within the context of a filter or wrapper methodology.[13] A more recent development considers the feature creation / selection problem from the perspective of a two stage process [45]: (1) identify a mapping from the original feature space to some new feature

[13] Feature selection (construction) was performed using a GA (GP) and an independent classification algorithm employed for constructing the classifier / validating the selected attributes.

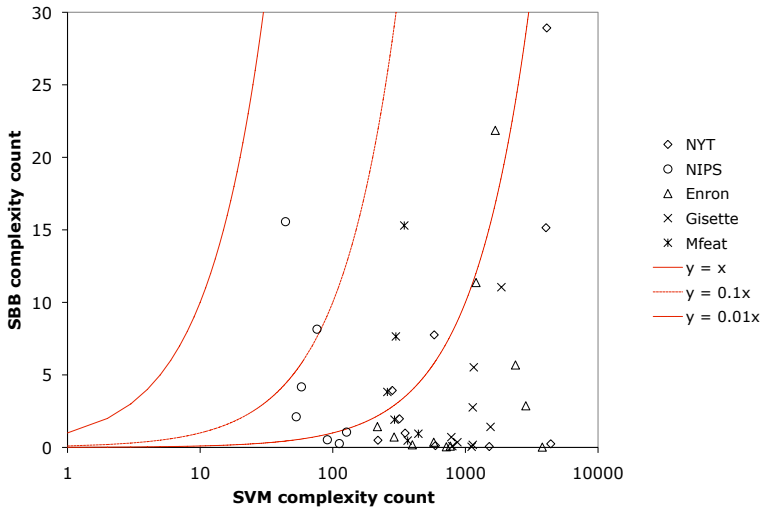


Fig. 8 F+SVM versus SBB solution complexity. The complexity of each F+SVM solution (support vector count, x-axis) is compared against the complexity of a composite SBB solution assuming the same attribute count (Eq. 7, y-axis). Equal model complexity is indicated by the '$y = x$' line, points to the left of this line indicate simpler SVM solutions, whereas points to the right indicate simpler SBB solutions. SVM models with at least 1 order (2 orders) of magnitude greater complexity fall to the right of the $y = 0.1x$ ($y = 0.01x$) curve.

space, and (2) apply an independent model to provide class labels relative to the new feature space. Step 1 makes use of a unique form of GP with performance evaluated relative to a multi-objective evaluation of Step 2, the classification stage i.e., a wrapper style of algorithm. Such a process naturally requires that the classifier be a linear model; a constraint that implies that if the mapping is effective, then nothing more complex should be necessary at the classification stage. Finally, in adopting a Graphic Processing Unit (GPU) framework for 'general purpose' computing, Langdon and Harrison deploy a multi stage process for attribute selection and classifier construction in a gene mining task [23]. An initial population of GP individuals is first evolved under a 15 node limit[14] relative to a very large attribute space of a million. Analysis of GP individuals from this first run would identify a smaller attribute space of up to ten thousand attributes. Re-applying the process a third time limited the number of attributes to around one hundred. In order to provide sufficient sampling of such a large attribute space the initial population (in the first pass) consisted of five million programs. The GPU makes the evaluation of such a large population feasible, whereas the low node count ensured that solutions were transparent and readily interpretable. The principle constraint appeared to be the relatively small exemplar count (91), where the PCI host to GPU interface might reduce the efficiency on larger data sets unless an appropriate subsampling scheme is adopted.[15]

Compared to the framework adopted in this work, we note that identifying solutions with non-overlapping behavioural traits is also not generally considered

[14] Tree structured GP.

[15] In terms of the total memory requirement, a million attribute data set with 91 exemplars is most similar to the NIPS data set as reported here (Table 1).

as an objective, hence reducing the resolution at which any correlation between attribute support and outcome could be made. Conversely, this work demonstrates that the ability to construct teams with potentially more than one classifier per class results in a partitioning of the data set and the discovery of unique attribute subspaces per partition. Indeed, a recent survey paper of GP as applied to the supervised learning task of classification indicates that research as a whole continues to focus on addressing single aspects of the classification task [11], without recognizing that data sets generally require multiple factors to be addressed simultaneously.

6.2 Evolutionary problem decomposition

As remarked in Section 2.2 cooperative coevolution assumes an explicitly 'divide and conquer' approach to constructing candidate solutions. In the following we provide a summary of current approaches relative to an architectural categorization: Single population, Multi-population, and Hierarchical.

Single population: Teams are composed from a number of individuals as sampled from the *same* population. Naturally, maintaining sufficient diversity within the population becomes very important. One approach to this is to place a lot of emphasis on designing fitness functions that explicitly capture diversity as well as overall accuracy (of the team) *e.g.*, negative correlation [28,5, 13], strongly typed GP [21], local membership functions [33,31,32]; frequently under the context of multi-objective fitness formulations. One potential disadvantage of the single population model is that a recently introduced offspring can disrupt good quality team behaviour, as recognized in the 'profit sharing problem' of learning classifier systems [42]. Conversely, a representation might be assumed in which a team (individual) explicitly consists of n programs [3]. Prior knowledge for the relevant number of cooperating programs is necessary as teams also specify an context for crossover operators. Thus, crossover at the team level would emphasize trading material between programs in the same team loci. One outcome of adopting such a scheme, however, was strong team behaviour, but 'weak' behaviour from individual team members [40]. In effect team members were learning to act everywhere, potentially negating insight into solutions. Recent approaches for addressing such a drawback represent the team as an archive of individuals independent from the population [31,32] or apply some form of ensemble selection post training [5,13].

Multi-population: Teams are composed by sampling individuals from n *different* populations. Each element of a solution is associated with an independent population or 'species'. In the framework of Potter and de Jong the number of species is not pre-specified, but begins at a suitable lower bound [37]. Fitness is evaluated across a team of individuals, one sampled from each population currently in existence. However, a mechanism is then required for pushing the fitness assigned to the team, back to the elements in their independent populations. Without this it was not possible to apply credit assignment on the populations i.e., selection was performed at each population independently. Additional populations can be added when the (team) fitness is considered to have plateaued. Defining the conditions for such a plateau, however, are

notoriously difficult. Conversely, the Orthogonal Evolution of Teams (OET) framework circumvents many of these drawbacks by performing parent selection at the 'level' of individuals whereas survivor selection is determined at the team 'level' [40]. The principle drawback under OET being a requirement to define the specific number of team members a priori.

Hierarchical: Both the aforementioned 'Single population' and 'Multi-population' approaches emphasize a 'flat' lateral representation of the teaming task. Thus, one of three scenarios appears: 1) a stochastic selection operator defines team membership, 2) a priori knowledge explicitly fixes the number of members comprising a team, or 3) the fitness function attempts to reward team-like interactions. Conversely, representations can be adopted that explicitly reflect the multi-level nature of declaring teams versus individuals (*cf.*, collaborators). The SANE scheme for evolving neural networks made use of a population of neurons and a second population to represent the network as a whole *cf.*, the team [35]. Neither population independently represents a solution, only when selecting an individual from the network population (team level) and then building a solution using the components referenced from the neuron population (individual level) is a solution explicitly defined. Assuming a variable length representation for the declaration of teams implies that team 'size' need not be predefined. Moreover, declaring variation operators hierarchically – from team to team members – protects currently functioning teams from the loss of important team members. Specific examples of frameworks utilizing explicitly hierarchical representations include Evolutionary Strategies (ES) [43], Evolutionary Neural Networks [35] and Symbiosis in general [18]. Indeed, the construction of solutions with multiple team members operating at multiple levels of cooperation have also been demonstrated in both GP [27] and ES [43].

7 Conclusions

Data mining under large attribute spaces is revisited from the context of coevolutionary model building using the specific instance of the SBB algorithm. This paradigm extends the original embedded basis of GP to the parallel identification of teams of classifiers with non-overlapping behaviour from a single run. The initial hypothesis was that, in doing so, simpler solutions would be identified with team members using fewer attributes compared to solutions relying on a single classifier. This was indeed the case. Attribute subspaces associated with SBB team members were found to consist of no more than a worst case scenario of 6 attributes, whereas the total team attribute subspace might consist of up to 70 attributes (both under the multifeature data set). Moreover, worst case instruction counts for the corresponding team members was generally 5 to 7 instructions, indicating that small attribute subspaces have not been traded for higher model complexity.

In comparison with the widely used SVM model of large margin classification, a strong bias towards simpler models was apparent in the SBB solutions. Even when model complexity was compared across the entire team, the resulting attribute subsets were generally an order of magnitude lower than those resulting from the SVM. The penalty paid for this is reflected in the ensuing classification performance. However, under the Bag-of-words domain, SBB solutions were

generally non-dominated with equivalent classification performance achieved and models utilizing a fraction of the attributes or model complexity.

Pruning the attribute spaces supporting the SVM models resulted in two distinct profiles: there was either a modest impact on the resulting classification performance (Multifeature, Gisette and NIPS), or there was a distinct knee in the attribute count – detection curve after which performance dropped significantly (Enron and NYT). Moreover, cases with the knee-style profile also observed a non-linear interaction between attribute count and model complexity as SVM complexity increased before the knee (as attribute subset size decreased) and then increased after some sweet spot in attribute count as the number of attributes included in the model increased.

Several avenues of future work are of interest. In particular, on review of the attributes selected under SBB and F+SVM methodologies it became readily apparent that the subsets of attributes selected were rather different. A more detailed characterization of these differences would be worth undertaking. Secondly, we note that implementations of Bayesian methods for embedded or filter style model building / attribute selection are increasingly becoming available. Thus, future work could also benefit from including such models within the comparison. Finally, the bioinformatic domain is becoming an increasingly significant source of data sets with large attributes and might also represent a good fit for coevolutionary style algorithms.

Naturally, further development of algorithms for providing accurate yet simple solutions is generally encouraged. The current focus on measuring performance in terms of accuracy alone has resulted in models that are essentially opaque [10]. More generally, the design of evolutionary methods that address multiple task properties simultaneously – class imbalance, large data sets, large attribute spaces – represents a relatively 'untouched' opportunity for further research. With this in mind, the authors believe that coevolutionary mechanisms potentially provide a rich repertoire of capabilities for addressing multiple requirements simultaneously.

8 Acknowledgments

The authors gratefully acknowledge scholarships provided by MITACS and NSERC and equipment provided under the CFI New Opportunities program (Canada). This research was conducted while J. Doucette was an NSERC USRA at Dalhousie University.

References

1. A. Asuncion and D. J. Newman. UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/mlrepository.html]. Irvine, CA: University of California, Dept. of Information and Comp. Science, 2008.
2. M. Brameier and W. Banzhaf. A comparison of linear Genetic Programming and Neural Networks in Medical data mining. *IEEE Transactions on Evolutionary Computation*, 5(1):17–26, 2001.
3. M. Brameier and W. Banzhaf. Evolving teams of predictors with linear Genetic Programming. *Genetic Programming and Evolvable Machines*, 2(4):381–407, 2001.
4. J. Cartlidge and D. Ait-Boudaoud. Autonomous virulence adaptation improves coevolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):215–229, 2011.

5. A. Chandra, H. Chen, and X. Yao. *Trade-off between diversity and accuracy in ensemble generation*, chapter 19, pages 429–464. 2006. In ([19]).
6. Y.-W. Chen and C.-J. Lin. *Combining SVMs with Various Feature Selection Strategies*, chapter 12, pages 315–324. 2006. In ([15]).
7. E. D. de Jong. A monotonic archive for Pareto-coevolution. *Evolutionary Computation*, 15(1):61–93, 2007.
8. J. Doucette and M.I. Heywood. GP classification under imbalanced data sets: Active subsampling and AUC approximation. In M. O'Neill *et al.*, editor, *European Conference on Genetic Programming*, volume 4971 of *LNCS*, pages 266–277, 2008.
9. J. Doucette, P. Lichodzijewski, and M.I. Heywood. Evolving coevolutionary classifiers under large attribute spaces. In R. Riolo, T. McConaghy, and U.-M. O'Reilly, editors, *Genetic Programming Theory and Practice*, volume VII, pages 37–54. Springer-Verlag, 2009.
10. C. Drummond. Machine learning as an experimental science (revisited). In C. Drummond, W. Elazmeh, and N. Japkowicz, editors, *AAAI Workshop on Evaluation Methods for Machine Learning*, pages 1–5, 2006.
11. P. G. Espejo, S. Ventura, and F. Herrera. A survey on the application of Genetic Programming to Classification. *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, 40(2):121–144, 2010.
12. S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In J. Kelemen and P. Sosik, editors, *Proceedings of the 6th European Conference on Advances in Artificial Life*, volume 2159 of *LNAI*, pages 316–325, 2001.
13. C. Gagné, M. Sebag, M. Schoenauer, and M. Tomassini. Ensemble Learning for Free with Evolutionary Algorithms? In D. Thierens *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1782–1788, 2007.
14. I. Guyon, S. Gunn, A. B. Hur, and G. Dror. *Design and Analysis of the NIPS2003 Challenge*, chapter 9, pages 237–263. 2006. In ([15]).
15. I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Selection: Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*. Spinger-Verlag, 2006.
16. P. Haffner. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 48:239–261, 2006.
17. P. Haffner and S. Kanthak. Fast kernel learning with sparse inverted index. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-scale Kernel Machines*, pages 51–71. MIT Press, 2007.
18. M. I. Heywood and P. Lichodzijewski. Symbiogenesis as a mechanism for building Complex Adaptive Systems: A review. In C. Di Chio *et al.*, editor, *EvoComplex*, volume 6024 of *LNCS*, pages 51–60, 2010.
19. Y. Jin, editor. *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Spinger-Verlag, 2006.
20. K. Krawiec. Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.
21. R. Kumar, A.H. Joshi, K.K. Banka, and P.I. Rockett. Evolution of hyperheuristics for the biobjective 0/1 knapsack problem by multiobjective Genetic Programming. In M. Keijzer *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1227–1234, 2008.
22. T.N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. *Embedded Methods*, chapter 5, pages 137–165. 2006. In ([15]).
23. W. B. Langdon and A. P. Harrison. Gp on spmd parallel graphics hardware for mega bioinformatics data mining. *Soft Computing*, 12(12):1169–1183, 2008.
24. R. Law. The symbiotic phenotype: Origins and evolution. In L. Margulis and R. Fester, editors, *Symbiosis as a Source of Evolutionary Innovation: Speciation and Morphogenesis*, chapter 5, pages 57–71. MIT Press, 1991.
25. P. Lichodzijewski and M. I. Heywood. Coevolutionary bid-based Genetic Programming for problem decomposition in classification. *Genetic Programming and Evolvable Machines*, 9(4):331–365, 2008.
26. P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In M. Keijzer *et al.*, editor, *ACM Proceedings of the Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

27. P. Lichodzijewski and M. I. Heywood. The Rubik Cube and GP temporal sequence learning: An initial study. In R. Riolo, T. Soule, and Bill Worzel, editors, *Genetic Programming Theory and Practice*, volume VIII, chapter 3, pages 35–54. Springer-Verlag, 2010.
28. Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
29. L. Margulis and R. Fester, editors. *Symbiosis as a Source of Evolutionary Innovation.* MIT Press, 1991.
30. J. Maynard Smith. *A Darwinian View of Symbiosis*, chapter 3, pages 26–39. 1991. In ([29]).
31. A. McIntyre and M.I. Heywood. Pareto cooperative-competitive genetic programming: A classification benchmarking study. In R. Riolo, T. Soule, and Bill Worzel, editors, *Genetic Programming Theory and Practice*, volume VI, chapter 4, pages 41–60. Springer-Verlag, 2008.
32. A. McIntyre and M.I. Heywood. Classification as clustering: A Pareto cooperative-competitive GP approach. *Evolutionary Computation*, 19(1):137–, 2011.
33. A. R. McIntyre and M. I. Heywood. MOGE: GP classification problem decomposition using multi-objective optimization. In M. Keijzer *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 863–870, 2006.
34. A.R. McIntyre and M.I. Heywood. Cooperative problem decomposition in Pareto Competitive classifier models of coevolution. In M. O'Neill, editor, *European Conference on Genetic Programming*, volume 4971 of *LNCS*, pages 289–300, 2008.
35. D. E. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(4):373–399, 1998.
36. J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection. In L. Spector *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 493–500, 2001.
37. M. Potter and K. de Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
38. C.D. Rosin and R.K. Belew. New methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.
39. M.G. Smith and L. Bull. Genetic Programming with a Genetic Algorithm for Feature Construction and Selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.
40. R. Thomason and T. Soule. Novel ways of improving cooperation and performance in Ensemble Classifiers. In D. Therens *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1708–1715, 2007.
41. G. M. Weiss and R. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Articial Intelligence Research*, 19:315–354, 2003.
42. T. H. Westerdale. Local reinforcement and recombination in Classifier Systems. *Evolutionary Computation*, 9(3):259–281, 2001.
43. S. X. Wu and W. Banzhaf. A hierarchical cooperative Evolutionary Algorithm. In J. Branke *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 233–240, 2010.
44. Y. Zhang and P. I. Rockett. *Feature extraction using multi-objective Genetic Programming*, chapter 4, pages 75–99. 2006. In ([19]).
45. Y. Zhang and P. I. Rockett. A generic multi-dimensional feature extraction method using multi-objective genetic programming. *Evolutionary Computation*, 17(1):89–115, 2009.