Budgeted Classification with Rejection: An Evolutionary Method with Multiple Objectives

Nolan H. Hamilton and Errin W. Fulp
Department of Computer Science
Wake Forest University, Winston-Salem, NC, USA
Email: haminh16@wfu.edu, fulp@wfu.edu

Abstract—Classification systems are often deployed in resource-constrained settings where labels must be assigned to inputs on a budget of time, memory, etc. Budgeted, sequential classifiers (BSCs) address these scenarios by processing inputs through a sequence of partial feature acquisition and evaluation steps with early-exit options. This allows for an efficient evaluation of inputs that prevents unneeded feature acquisition. To approximate an intractable combinatorial problem, current approaches to budgeted classification rely on well-behaved loss functions that account for two primary objectives (processing cost and error). These approaches offer improved efficiency over traditional classifiers but are limited by analytic constraints in formulation and do not manage additional performance objectives. Notably, such methods do not explicitly account for an important aspect of real-time detection systems—the fraction of "accepted" predictions satisfying a confidence criterion imposed by a risk-averse monitor.

We propose a problem-specific genetic algorithm to build budgeted, sequential classifiers with confidence-based reject options. Three objectives—accuracy, processing time/cost, and coverage—are considered. The algorithm emphasizes Pareto efficiency while accounting for a notion of aggregate performance via a unique scalarization. Experiments show our method can quickly find globally Pareto optimal solutions in very large search spaces and is competitive with existing approaches while offering advantages for selective, budgeted deployment scenarios.

Index Terms—machine learning, budgeted classification, reject option, early-exit, selective classification, evolutionary computation

I. Introduction

Many real-world classification scenarios present latency constraints between observation of inputs and label assignment, and a common difficulty regards balancing processing cost with accuracy [1]–[3]. In these *budgeted* learning settings, the magnitude of resources (e.g., time, memory) expended while evaluating inputs represents the *cost of classification*, where a greater cost of classification is generally associated with improved classifier performance.

Cost of classification can be specified further as the sum of *feature acquisition cost* and *classifier evaluation cost*. Feature acquisition cost addresses the resources spent generating features for classification that are not present during the initial observation of inputs at test-time. With features acquired, classifier evaluation cost measures the resources exhausted while assigning labels to inputs using a trained classifier and is often negligible compared to feature acquisition cost [4].

As an example scenario warranting a budgeted approach, consider the task of classifying e-mail messages as spam. For timeliness and users' security, a large influx of messages must be evaluated quickly, and informative features such as subject-line character diversity, sender reputation score, sender location, URL count, etc. can require considerable cpu-time to compute. Generating all of these features for every input may therefore impose unacceptable processing cost. Fortunately, it is often possible to use small, cheap subsets of features to correctly classify a considerable fraction of test-time inputs [1]–[3], [5], [6], and *budgeted, sequential classifiers* leverage this property to reserve intensive processing for only the inputs that require it. Using this design, a set of classifiers with varying feature sets are arranged in a sequence of stages to evaluate inputs [1], [2], [7]. Once an input receives a confident class prediction in a particular stage, processing ceases and no cost is incurred for the remaining stages' features that were not used. Specific use-cases of sequential, budgeted learning can be found across a diverse range of application domains [8], [9]. However, while budgeted learning schemes present an opportunity for efficient processing of inputs in time-critical instances, there are a multitude of important design considerations that can affect performance of these protocols, making configuration a difficult optimization problem that has received significant attention in the past decade [1]–[3], [5], [6], [10].

II. Related Work

A variety of budgeted approaches have been proposed in the past decade. In this section, we conduct a brief survey of some prominent algorithms. In general, budgeted classifiers balance the cost-accuracy trade-off by minimizing a well-behaved objective function that is increasing with respect to error and cost of classification. Note, there are several loosely-related learning paradigms that will not be addressed in this manuscript. *Classifier cascades*, for example, leverage a fundamental assumption of class imbalance and do not assign positive labels at intermediate stages [11].

The authors in [2] formulate the problem with *early-exit* options at each stage that prevent unnecessary feature acquisition by ceasing processing when early-stage predictions are deemed conclusive. To avoid difficult combinatorial aspects, the order of features is fixed beforehand. The authors construct and minimize a global, smooth cost function by coordinate descent and can thus guarantee local optimality of solutions—but only for the chosen ordering of features and stages. Wang et al. [7] offers improved theoretical guarantees relative to the

approach proposed in [2] by formulating the problem in a convex framework (linear program). The authors are then able to guarantee globally optimal solutions under their approximate formulation and a fixed feature/stage order.

Several existing methods incorporate feature selection/order implicitly. [3] proposes *GreedyMiser*—a feature-budgeted variant of stage-wise regression [12]. Limited-depth regression trees are used as weak learners and are constructed with a modified impurity function accounting for cost of feature extraction. These weak learners are combined to form a final classifier. For its simplicity and efficacy, GreedyMiser has become one of the most popular and best-cited feature-budgeted approaches to classification and is frequently used as a benchmark [5], [6], [13], [14]. [5] proposes *Cost-Sensitive Tree of Classifiers*; This method builds a budgeted tree of classifiers with leaf nodes optimized for a specific subset of the input space.

While the described methods offer a marked improvement in efficiency over non-budgeted classification of inputs in resource-constrained settings, several limitations are consistent throughout this body of work. First, many of the methods do not optimize the order of the features/stages [2], [7] and instead resort to "increasing cost" heuristics to choose an ordering of features (expensive features placed at later stages). But this can be inefficient if, for instance, a large fraction of cheap features are uninformative. In addition, intuitively modeling complex feature interactions beforehand can prove difficult, and such interactions can prove consequential if variables are more/less discriminative when grouped together [1].

Another common theme of the methods discussed above is that they exploit a cost/accuracy trade-off in which cheaper solutions are generally less accurate than costly solutions. Unfortunately, the decreased accuracy of the cheaper classifiers can restrict applicability in critical, real-world environments. One possible remedy to increase accuracy of classifiers is to apply a *reject option* that rejects low-confidence predictions (i.e., inputs with unconfident predictions are discarded). These *selective classifiers* have been studied independently from budgeted classification and seek to balance the *coverage*-accuracy tradeoff, where coverage refers to the expected fraction of inputs that are *not* rejected [15], [16]. Existing budgeted approaches do not apply reject options, but perfect coverage is often unnecessary in practical scenarios [16], and conservative decision makers with cost constraints may benefit in sacrificing coverage (rather than processing efficiency) for improved accuracy.

III. EMSCO—*Evolutionary Multi-Stage Classifier Optimizer*

The limitations discussed in the previous section motivate a budgeted *and* selective protocol accounting for accuracy, cost, and coverage during optimization. We propose such a method with the aim of:

1) Offering greater accuracy than existing budgeted methods by rejecting uncertain predictions

2) Matching or reducing processing cost compared to existing budgeted methods
3) Preserving high coverage

A problem-specific genetic algorithm serves as the fundamental mechanism for optimization. Such genetic algorithms (GAs) employ a population-based approach to optimization in which the generation/population G_1 is a product of mutation, selection, and crossover on the solutions in G_0. GAs are noted for their ability to find optimal or near-optimal solutions for very large combinatorial problems in polynomial time and properly manage multiple objectives in a Pareto efficient manner [17]. These algorithms also grant significant mathematical flexibility in formulation, as there is no need for objectives to be smooth or otherwise well-behaved. In our setup, three objectives are considered: coverage (g_1), accuracy (g_2), and (inverse) feature acquisition cost (g_3) (See Section III-C for a detailed description of these objectives.) These objectives are optimized over a set of feasible stage designs, defined in Section III-B as ordered partitions of the feature set.

Note, we move several supplementary resources (pseudo-code, demos, additional experiments, etc.) that readers may find informative to a public repository[1].

A. Problem Setting

As in [2], we assume a set of training examples from past instances for which measurements of all features $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n$ and correct labels are available. The aim is to find system designs that reduce feature acquisition cost and maintain high accuracy and coverage at test/prediction time.

Upon initial observation at test-time, input $\mathbf{x}$ begins with no acquired features, and features are thereafter attained as needed through a sequence of stages. That is, stage j acquires a subset of features $Q_j \subseteq \mathcal{F}$ to evaluate[2] input $\mathbf{x}$. With these features acquired, the stage-j classifier, $\mathscr{C}(Q_j, \mathbf{x})$, learned beforehand on features in Q_j during the training phase, returns a set of prediction confidences $\mathscr{P}_{\mathbf{x_j}} = \{p_1, p_2, \ldots, p_l\}$ corresponding to possible labels for input $\mathbf{x}$. Note, every feature has a corresponding cost of acquisition given by the set $\mathcal{C}$—acquiring feature $\mathcal{F}_i$ for evaluation of $\mathbf{x}$ increases the cost of classification incurred by the input by $\mathcal{C}_i$ units.

We assume a *risk-averse monitor* has decided to accept only confident predictions to improve accuracy. At preterminal stages $j < k$, an early-exit decision is made to determine whether the input prediction is sufficiently confident to be accepted or if additional processing is necessary. Let $\bar{p}_j = \max \mathscr{P}_{\mathbf{x_j}}$ and $\hat{p}$ be a confidence threshold specified by the monitor. At stage $j < k$, if $\bar{p}_j < \hat{p}$, the input is sent to stage $j + 1$ where it is evaluated with feature set Q_{j+1} (Note, $Q_j \subset Q_{j+1}$). Conversely, if $\bar{p}_j \geq \hat{p}$, processing ceases, and $\mathbf{x}$ is assigned the label corresponding to the maximum class

---

[1]Available at https://github.com/nolan-h-hamilton/EMSCO-supplement

[2]For its generalization ability, computational efficiency, and accurate class probability estimates, EMSCO employs L_2-regularized logistic regression ($\lambda = 1$) at each stage in this paper. EMSCO's performance can be improved by using more complex classification methods (e.g., Random Forest) at each stage, but this may result in markedly increased training time for large datasets.

probability. At the final stage k, if $\bar{p}_k < \hat{p}$, evaluation of $\mathbf{x}$ is deemed *inconclusive* with no label assigned. In this case, we say that $\mathbf{x}$ has been *rejected*. This is a safe but generally undesirable result as it decreases utility of the system—while wanting high accuracy, the above-mentioned risk-averse monitor also desires that the system yield insight. With a confidence threshold specified *a priori* according to a system monitor's preferences and misclassification penalties, we aim to find a stage configuration that provides high accuracy, low processing cost, and high coverage. Figure 1 offers a visual depiction of sequential classification with early-exits and a reject option.

B. Solution Space

Let $k \in \mathbb{N}$ with $k < n$. Solutions are k-*partitions* of feature set $\mathcal{F}$. That is, $\bigcup_{j=1}^{k} Q_j = \mathcal{F}$ where $Q_j \neq \emptyset$ denotes the features acquired in stage j. Due to the sequential nature of BSCs, the ordering of the stages must also be considered, and the number of ordered k-partitions $|\mathcal{P}_{(n,k)}|$ on a feature set with dimension n is computed by multiplying the number of unordered partitions by $k!$:

$$|\mathcal{P}_{(n,k)}| = k! \cdot S_2(n,k) = \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n, \qquad (1)$$

where $S_2(n,k)$ represents the number of unordered k-partitions of a set with n elements, or a *Stirling number of the second kind*.

To consider a range of possible stage counts, we define the solution space $\mathcal{S}_{(n,k)}$ as the set of all solutions with *up to and including* k stages. That is,

$$\mathcal{S}_{(n,k)} = \bigcup_{j=1}^{k} \mathcal{P}_{(n,j)}. \qquad (2)$$

For instance, $\mathcal{S}_{(n,k=3)}$ includes solutions with one, two, or three stages.

Theorem 1. *(Size of Search-Space) For fixed $k \in \mathbb{N}$, the number of feasible solutions $|\mathcal{S}_{(n,k)}|$ is asymptotically equivalent to k^n.*

Proof. Since there is no intersection between terms on the RHS of (2), we have $\sum_{j=1}^{k} |\mathcal{P}_{(n,j)}| = |\mathcal{S}_{(n,k)}|$. Taking the limit of ratio $\frac{|\mathcal{P}_{(n,j)}|}{j^n}$ as $n \to \infty$ yields $|\mathcal{P}_{(n,j)}| \sim j^n$. We can then write:

$$\lim_{n \to \infty} \frac{1}{k^n} |\mathcal{S}_{(n,k)}| = \lim_{n \to \infty} \frac{1}{k^n} \sum_{j=1}^{k} |\mathcal{P}_{(n,j)}| = \lim_{n \to \infty} \frac{1}{k^n} \sum_{j=1}^{k} j^n$$

$$= \lim_{n \to \infty} \left(\frac{k}{k}\right)^n = 1. \text{ That is, } |\mathcal{S}_{(n,k)}| \sim k^n.$$

$\square$

In practice, all features in stage j are automatically appended to stage $j+1$ as they are already acquired and can be referenced for no cost. However, this technicality does not affect the above analysis, since it depends only on the unique elements acquired at each stage.

As our defining structure for BSCs, ordered feature set partitions possess several beneficial properties. In many cases, certain features are uninformative alone but become highly discriminative when evaluated together [1]. By allowing stages to contain multiple features, these scenarios are acknowledged implicitly during optimization. Additionally, using the representation proposed in Section IV-A, ordered feature set partitions can be conveniently modeled as chromosomes.

C. Objectives

Objectives are designed to incorporate important performance aspects of a real-time classification system. The relationship between objectives is explored visually in Figure 2. Note, in this subsection, m denotes the index of the final stage at which input $\mathbf{x}$ is processed. $\hat{\mathcal{X}}$ is a held-out subset of data used to estimate performance of solutions.

1) Coverage: In our setup, coverage is measured as the fraction of predictions that meet a targeted confidence threshold, $\hat{p}$. Several methods have been proposed to determine reject decisions [16], but we utilize confidence-based reject decisions because they are frequently used in practice [18], [19] and efficient to compute.

$$g_1(Q) = \frac{1}{N} \sum_{\mathbf{x} \in \hat{\mathcal{X}}} \mathbb{1}[\bar{p_m} \geq \hat{p}] \qquad (3)$$

If coverage were not considered during optimization, a risk-averse monitor accepting only confident predictions may obtain little information from the resulting configuration since a system rejecting the vast majority of inputs (but correctly assigning labels to a few) could be favored. We assume system monitors desire accuracy but also wish that the classification system yield sufficient insight and does not reject an excessive fraction of predictions.

2) Accuracy: As in selective classification [15], we measure accuracy as the proportion of *accepted and correctly classified* inputs with respect to the total number of accepted predictions. Let y denote the true label for $\mathbf{x} \in \hat{\mathcal{X}}$, and $\bar{\mathscr{C}}_m(Q_m, \mathbf{x})$ return the label corresponding to the maximum class probability after evaluation in stage m. For $N^* = N \cdot g_1(Q)$, accuracy is measured as:

$$g_2(Q) = \begin{cases} \frac{1}{N^*} \sum_{\mathbf{x} \in \hat{\mathcal{X}}} \mathbb{1}[\bar{\mathscr{C}}_m(Q_m, \mathbf{x}) = y \ \& \ \bar{p_m} \geq \hat{p}] & N^* > 0, \\ 0 & N^* = 0 \end{cases}$$
$$\qquad (4)$$

The case $N^* = 0$ was not realized in this paper's experiments.

3) Cost: Because feature acquisition typically comprises the bulk of test-time processing expense [4], we disregard classifier evaluation cost and measure cost $g_3^*(\cdot)$ as the average sum of acquisition costs per test input:

$$g_3^*(Q) = \frac{1}{N} \sum_{\mathbf{x} \in \hat{\mathcal{X}}} \left(\sum_{j=1}^{m} \mathcal{C}_{Q_j} \right),$$

where $\mathcal{C}_{Q_j}$ denotes the sum of feature costs acquired in stage j. Note that the cost of rejected records is included since the system requires resources to arrive at an inconclusive result.
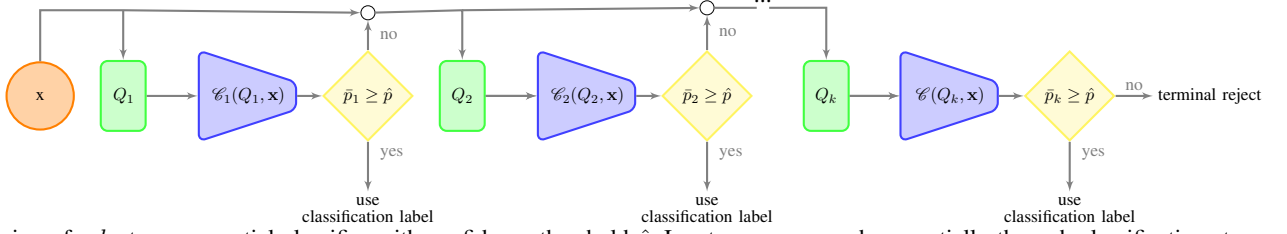
Fig. 1: Design of a k-stage sequential classifier with confidence threshold $\hat{p}$. Inputs are processed sequentially through classification stages until they achieve a class probability greater than $\hat{p}$ or all features have been exhausted.
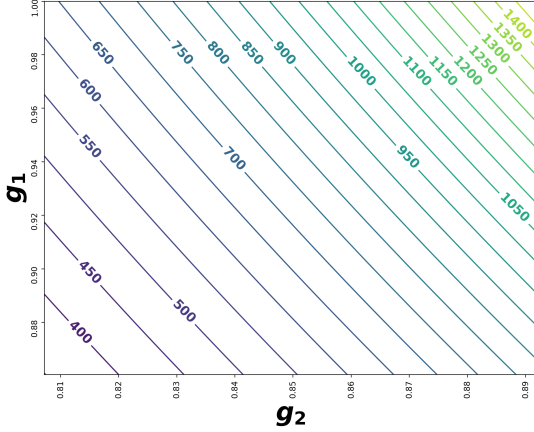

Fig. 2: *Performance Trade-offs*. A smooth approximation of the fitness landscape for the Credit data set (See Section V-A) with $k = 4$ exhibiting the trade-offs between cost, coverage, and accuracy. Color is determined by cost (g_3^*), with the yellow regions corresponding to solutions with greater cost. Cost is greatest where accuracy and coverage are highest.

To maintain consistency as a maximization problem and a $(0, 1]$ or $[0, 1]$ scale for objectives, the minimum raw cost of all solutions in the current population is determined and used for normalization. Then, for $Q \in G_h$, *inverse cost* $g_3(Q)$, is measured as:

$$g_3(Q) = \frac{\min_{U \in G_h} g_3^*(U)}{g_3^*(Q)} \quad (5)$$

D. Problem Definition

EMSCO addresses the following global multi-objective optimization by seeking solutions that are *non-dominated* [17] in the global solution space $\mathcal{S}_{(n,k)}$:

$$\underset{Q \in \mathcal{S}_{(n,k)}}{\arg\max} \left(g_1(Q), g_2(Q), g_3(Q) \right). \quad (6)$$

In this manuscript, solutions to this problem are referred to as "globally optimal" or "globally non-dominated".

IV. EVOLVING BUDGETED, SEQUENTIAL CLASSIFIERS

This section details the construction and behavior of EMSCO. We first describe each of its components independently and then connect them for summary in Subsection IV-D.

A. Chromosome Representation

To employ an evolutionary approach a chromosome representation for BSC designs (ordered feature set partitions) is needed. Ordered feature set partitions can be conveniently represented as lists of integers. Let Q be a solution in $\mathcal{S}_{(n,k)}$. Then the chromosomal representation of Q is denoted as $[Q]$, where

$$[Q] \in \{0, 1, \ldots, k-1\}^n. \quad (7)$$

The jth list element $[Q]_j$ corresponds to the jth feature's stage assignment. The elements in $[Q]$ are *zero-indexed*; so $[Q]_j = s$ assigns feature j to stage $s + 1$. For instance, $\{0, 0, 2, 1, 2\}$ represents a system in which the first two features are assigned to the first stage, the fourth feature is assigned to the second stage, and the third and fifth features are assigned to the third stage. Note, stage count of solution Q is denoted as $|Q|$, with $|Q| = (\max [Q]) + 1$.

This relation between chromosome representations and ordered feature set partitions is not bijective. For example, $\{0, 0, 0, 2\} \in \{0, 1, 2\}^4$, but since stage two is empty, this chromosome does not correspond to any ordered feature set partition in $\mathcal{S}_{(4,3)}$. This issue is attenuating for n large relative to k, but as a proactive remedy, EMSCO checks for empty stages with a function, $gaps([Q])$, that returns true if any stage is empty. If any such gaps exist, the solution is modified using a "stage compression" procedure (Algorithm 1). For instance,

---

Algorithm 1: Stage Compression

1 **function** $compress([Q])$
2 **if** $gaps([Q])$ **then**
3      $current\_stage \leftarrow 0$;
    // $unique(S)$ returns unique elements in S.
4   **for** $stage \in unique([Q])$ **do**
5   **for** $index, value \in enumerate([Q])$ **do**
6   **if** $value = stage$ **then**
7                  $[Q]_{index} \leftarrow current\_stage$;
8   **end**
9              $current\_stage \leftarrow current\_stage + 1$;
10   **end**
11   **end**
12 **end**
13 **return** $[Q]$;

$\{0, 0, 2, 3\}$ is compressed to $\{0, 0, 1, 2\}$ in the following steps:

$$\{0, 0, 2, 3\} \rightarrow \{0, 0, 1, 3\} \rightarrow \{0, 0, 1, 2\}.$$

B. Evolutionary Processes

EMSCO's evolutionary operators are designed to leverage problem-specific knowledge while maintaining sufficient

breadth of search. These operators are applied in sequence to form a new population per generation. Each operator is detailed in the following subsections.

1) Selection: Selection is the process in which individual solutions (chromosomes) are chosen from a population for the recombination stage. EMSCO utilizes roulette wheel selection (RWS) [20]. In this scheme, a chromosome's probability of selection is proportional to its *fitness* (as described in Section IV-C). That is, solutions with greater fitness have a greater probability of being selected while not completely excluding solutions with low fitness. This latter property is considered desirable in our problem context.

2) Elitism: The elitism protocol applied by EMSCO is designed to preserve all *unique* non-dominated solutions to the subsequent generation. Let E_0^* denote the set of unique non-dominated chromosomes within generation G_h, and let G_h^* denote the set of all unique solutions in generation G_h. For elitism parameter $b \in [0,1]$, the top

$$M = \max \left(round\left(b \cdot |G_h^*| \right), |E_0^*| \right),$$

unique solutions according to fitness (12) in each generation are sent to the subsequent generation without modification. Note, $round(x)$ returns the nearest integer for $x \in \mathbb{R}^+$. As can be seen in Algorithm 2, this elitism protocol requires only $|G| - M \leq |G|$ chromosomes to be created and evaluated in each generation.

3) Mutation: In general, mutation operators are used to maintain genetic diversity from one generation's population of chromosomes to the next [20]. In addition to maintaining diversity, we design EMSCO's mutation operator to leverage problem-specific aspects of sequential, budgeted classification.

If performance is comparable between solutions $Q_A, Q_B \in G_h$, the solution with lower stage count is preferred since it is more appealing from the perspective of model simplicity and its worst-case number of classifier evaluations is lower than the alternative. Thus, to encourage lower stage counts during optimization, a discrete, monotonically decreasing probability distribution is desired to mutate chromosomes' stage assignments. The beta-binomial distribution with $\alpha = 1$ & $\beta > \alpha$, satisfies these criteria and possesses several convenient properties for adjusting bias of the mutation operator toward low stage assignments.

Let $0 \leq j \leq |Q|$ be a potential stage assignment for the ith feature in chromosome Q. At index i, if $rand(0,1) < \hat{m}$, the probability mass function for stage assignment $[Q]_i$ is:

$$\mathbb{P}_{mut.} \left([Q]_i \leftarrow j \right) = \binom{|Q|}{j} \frac{B\left(j+1, |Q|-j+\beta \right)}{B\left(1, \beta \right)} \quad (8)$$

where $B(\cdot)$ denotes the beta function. For a chromosome with stage count $|Q|$, this distribution has expected value $\frac{|Q|}{\beta+1}$, and β can be increased (decreased) to increase (decrease) bias towards low stage counts.

When warranted, it is important that the mutation operator allows creation of new stages—in scenarios with many expensive features, a low stage count may yield high feature
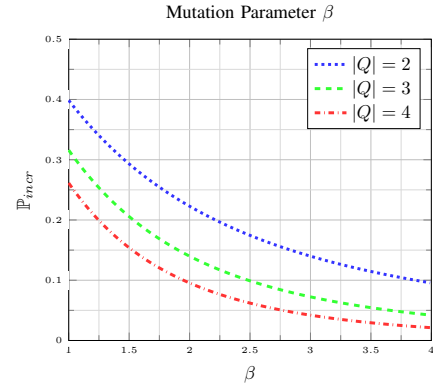

Fig. 3: Greater β corresponds to lower probability of incrementing stage count. Fifteen features and a mutation rate of 10% are assumed for this demonstration.

acquisition cost at each stage. The mutation operator creates new stages in $[Q]$ by assigning $|Q|$ to an element[3]. Let

$$p_{Q^+} = \mathbb{P}_{mut} \left([Q]_j \leftarrow |Q| \right) = \beta \left(\frac{\Gamma(|Q|+1) \cdot \Gamma(\beta)}{\Gamma\left(|Q| + \beta + 1 \right)} \right)$$

For mutation rate $\hat{m}$ and number of features n, the probability that the mutation operator increments solution Q's stage count is given by:

$$\mathbb{P}_{incr}(Q) = 1 - (1 - \hat{m} \cdot p_{Q^+})^n, \quad (9)$$

since this is the complement of the event in which no features are assigned stage $|Q|$. The effect of β on $\mathbb{P}_{incr}$ is depicted in Figure 3.

4) Recombination: EMSCO's recombination/crossover operator takes into account the differing contexts of stage assignments within the parent solutions. To implement crossover, two parent chromosomes $P_A, P_B \in G_h$ are selected according to the protocol described in Section IV-B1. With probability $1 - \hat{r}$, recombination/crossover sends one of P_A, P_B to G_{h+1} unchanged via a fair coin flip. With probability $\hat{r}$, P_A and P_B are combined with a problem-specific variant of uniform crossover [20] to produce a child $C \in G_{h+1}$. In this case, the first step in recombination of P_A and P_B randomly selects a stage count for the child solution based on the parents' stage counts—with equal probabilities $(.3\bar{3})$, three options are considered for $|C|$: $|P_A|, |P_B|, round\left(\frac{|P_A|+|P_B|}{2} \right)$, where the third option is incorporated to facilitate consideration of solutions with intermediate stage counts between the parents'. After a stage count is decided, each of the child's features must be assigned a stage. Each ith assignment is initially chosen to be either $[P_A]_i$ or $[P_B]_i$ with equal probability (0.5). Let $R \in \{P_A, P_B\}$ denote the parent corresponding to this choice. In an attempt to maintain context of the parent's stage assignment for the ith feature, the initial selection $[R]_i$ is first normalized according to $|R|$ to acquire the relative order of the stage assignment within parent solution R. This ratio is then multiplied by $|C|$, rounded, and decremented to obtain a zero-indexed, approximate analog of $[R]_i$ in $[C]$.

---

[3]Recall, the list representation is zero-indexed at each element, so a stage assignment of $|Q|$ sends the feature to new stage $|Q| + 1$

C. Evaluating Chromosomes

For a solution $Q \in G_h$, fitness is computed using the L_2 norm of $\{g_1(Q), g_2(Q), g_3(Q)\}$, denoted by $\mathscr{E}(Q)$, *and* an exponential term accounting for the *Pareto rank* of Q with respect to the current population. In this paper, Pareto rank, or *non-domination level* [21], is an indication of the solution's performance from the perspective of Pareto efficiency. This mixed formulation of fitness respects Pareto efficiency and raw aggregated performance simultaneously.

1) Pareto Rank of Chromosomes: To compute $rank(Q)$ in generation G_h, we first determine the non-dominated solutions in G_h. This set of solutions E_0 is then removed from G_h. The non-dominated solutions in set $G_h \setminus E_0$ are then stored in the set E_1. This process of removing non-dominated solutions continues so that E_t contains the non-dominated solutions in the population $G_h \setminus E_0 \setminus E_1 \setminus \ldots \setminus E_{t-1}$. Once all solutions have been assigned to some E_t, the ranking procedure is finished. Let E_{t^*} denote the final non-dominated set removed from G_h, and let $Q \in E_t$. Then

$$rank(Q) = t^* - t. \tag{10}$$

Our assignment of non-domination level is flipped— the first non-dominated set is commonly assigned zero rank [21]. However, we define $rank(\cdot)$ so that the initial set of non-dominated solutions is assigned the greatest value. The motivation for this becomes clearer in the following subsection, where we define fitness.

2) Fitness Function: Non-domination is an important aspect of performance, and as seen in (6), defines EMSCO's optimization problem formally. However, in the context of BSC design, relying on non-domination level exclusively can lead to an ineffective prioritization of objectives during selection because there may be important practical differences among solutions in the same non-domination level. As a remedy, we consider rank *and* scalarized performance simultaneously via the fitness function.

To account for both Pareto efficiency and aggregate performance, fitness of chromosome Q is computed as a product involving $rank(Q)$ and the L_2 norm of objectives,

$$\mathscr{E}(Q) = \sqrt{g_1(Q)^2 + g_2(Q)^2 + g_3(Q)^2}. \tag{11}$$

Note that $\mathscr{E}(\cdot) > 0$ since $g_3(Q)$ is always positive. Let $l_\mathscr{E}$ and $u_\mathscr{E}$ denote the minimum and maximum values of $\mathscr{E}(\cdot)$ in generation G_h, respectively. For some $\epsilon > 0$, we set $\gamma = \frac{u_\mathscr{E}}{l_\mathscr{E}} + \epsilon$. Fitness is then defined for $Q \in G_h$ as:

$$f(Q) = \gamma^{rank(Q)} \cdot \mathscr{E}(Q) \tag{12}$$

In this manuscript's experiments, we use $\epsilon = 0.01$, but this value could be increased to encourage greater separation between non-domination levels (See Property 2).

The global *scalarized* optimization problem posed by this fitness function is:

$$\underset{Q \in \mathcal{S}_{(n,k)}}{\arg\max} f(Q) \tag{13}$$

Since f is increasing with respect to rank and g_1, g_2, g_3, a solution to (13) is likewise a solution to (6).

D. Algorithmic Properties

EMSCO begins by generating a random initial population of $|G|$ chromosomes by calling the function $init()$. This function creates solutions by applying the mutation operator to the default one-stage solution $([0, 0, \ldots, 0])$. As a result, the beginning population consists of only two-stage solutions.

After the first population has been generated, the main loop begins. At any point, if the number of unique non-dominated solutions equals the initially specified population size $|G|$, population size is incremented by $inc \in \mathbb{Z}_{\geq 0}$. By default, $inc = 0$, and this setting was used in all experiments. However, problems with very large non-dominated fronts may warrant setting $inc > 0$. To mitigate computational expense incurred by large populations, if an increased population size is no longer necessary in later generations, $|G|$ is decremented (lines 13-14 in EMSCO pseudocode).

Three halting conditions are checked before each iteration by calling the function, $converged()$, which returns true if any of the following conditions are satisfied—(i) the maximum number of iterations (max_iter) have been executed, (ii) the highest-scoring chromosome has remained constant for the past g generations, or (iii) $inc = 0$ and the number of unique non-dominated solutions ($elite\_size$) is equal to population size $|G|$. If none of these halting conditions are true, the loop proceeds, and the next population of solutions is produced using elitism, selection, recombination, and mutation as described in Section IV-B. When the loop ends, the list of all non-dominated solutions in the final generation is returned, sorted by fitness. We use $\mathscr{N}(G_h)$ to denote the set of all non-dominated solutions in G_h.

Algorithm 2: EMSCO
---
1   $G_0 \leftarrow init()$;
2   $|G|_{cpy} \leftarrow |G|$;
3   $h \leftarrow 0$;
4   **while not** $converged()$ **do**
5      $rank(G_h)$;
6      $sort(G_h, key = f(\cdot))$;
7      $elite\_size \leftarrow \max\left(round\left(b \cdot |G_h^*|\right), |E_0^*|\right)$;
8      $elite\_pop \leftarrow G_h^*[0 : elite\_size]$;
9      $G_{h+1} \leftarrow_{append} elite\_pop$;
10   **if** $elite\_size = |G|$ **then**
11          $|G| += inc$;
12   **end**
13   **if** $elite\_size < (|G| - inc)$ **and** $|G| > |G|_{cpy}$ **then**
14          $|G| = |G| - inc$;
15   **end**
16   **while** $|G_{h+1}| < |G|$ **do**
17          $P_A, P_B \leftarrow select()$;
18          $[C] \leftarrow recombine([P_A], [P_B])$;
19          $[C] \leftarrow mutate([C])$;
20   **if** $gaps([C])$ **then**
21             $compress([C])$;
22   **end**
23          $G_h \leftarrow_{append} C$;
24   **end**
25      $h \leftarrow h + 1$;
26   **end**
27   **return** $\mathscr{N}(G_h)$;

Several notable properties follow immediately as consequences

of the design described heretofore.

Property 1. *EMSCO preserves globally non-dominated solutions and returns them at the terminal generation.*

The above property follows immediately from the elitism protocol described in Section IV-B2), since any globally non-dominated solution is likewise non-dominated in any subset of the population. We can then conclude that if any globally non-dominated solution $D \in \mathcal{N}(\mathcal{S}_{(n,k)})$ is encountered in any generation, it is guaranteed to be returned at the terminal generation.

Another notable property of EMSCO regards its incorporation of both non-domination level and scalarized performance, $\mathscr{E}(\cdot)$, into the fitness function that determines probability of selection.

Property 2. *Let $Q \in G_h$, and let probability of selection be denoted as $\mathbb{P}_{sel.}(Q)$. For chromosomes $R_A, R_B \in G_h$:*

$$rank(R_A) < rank(R_B) \implies \mathbb{P}_{sel.}(R_A) < \mathbb{P}_{sel.}(R_B),$$

even if $\mathscr{E}(R_A) > \mathscr{E}(R_B)$. (ii) Within a particular rank, probability of selection is strictly increasing with respect to $\mathscr{E}(Q)$.

Proof. (i) This property is a consequence of fitness proportionate selection and the construction of $f(\cdot)$ detailed in Section IV-C. Let R_A, R_B be chromosomes in generation G_h such that $r_B = rank(R_B) > r_A = rank(R_A)$. Because $\frac{u_{\mathscr{E}}}{l_{\mathscr{E}}}$ maximizes the ratio between any two $\mathscr{E}(R_A)$ and $\mathscr{E}(R_B)$, we have

$$\left(\frac{u_{\mathscr{E}}}{l_{\mathscr{E}}} + \epsilon\right)^{r_B - r_A} > \frac{\mathscr{E}(R_A)}{\mathscr{E}(R_B)}.$$

Manipulating the above inequality, we obtain

$$\left(\frac{u_{\mathscr{E}}}{l_{\mathscr{E}}} + \epsilon\right)^{r_A} \mathscr{E}(R_A) < \left(\frac{u_{\mathscr{E}}}{l_{\mathscr{E}}} + \epsilon\right)^{r_B} \mathscr{E}(R_B). \tag{14}$$

By (12), we then have $f(R_A) < f(R_B)$. The result (i) then follows from the use of roulette wheel selection in which $\mathbb{P}_{sel.}(Q) = \frac{f(Q)}{\sum_{U \in G_h} f(U)}$.
(ii). If R_A, R_B are in the same rank/non-domination level, the $\left(\frac{u_{\mathscr{E}}}{l_{\mathscr{E}}} + \epsilon\right)^{\hat{r}}$ terms cancel in (14). Fitness comparisons between R_A, R_B are then dependent on $\mathscr{E}(\cdot)$ exclusively. (ii) then follows from fitness proportionate selection. $\square$

It should also be mentioned that separation in fitness (and therefore probability of selection) between each rank decreases as rank decreases. For instance, there is greater separation between fitness scores in the Pareto front and second non-domination level than between fitness scores in the second-to-last and last non-domination levels.

V. Experiments

To evaluate EMSCO's capabilities, experiments are conducted on three data sets from the UCI Machine Learning Repository [22] and two synthetic data sets. A variety of confidence thresholds and feature cost schemes are considered. Experiments in Section V-B empirically evaluate EMSCO's capacity for global

optimization, and experiments in Section V-C compare EMSCO with various budgeted and/or selective classification protocols.

We randomly split the data sets into 50-25-25 training, validation, and testing sets. Doing so allows for efficient computation of unbiased estimates for out-of-sample error. EMSCO uses the training sets to learn individual classifiers $\{\mathscr{C}(Q_1), \ldots, \mathscr{C}(Q_j), \ldots, \mathscr{C}(Q_{|Q|})\}$. Validation sets are used to measure $(g_1(Q), g_2(Q), g_3^*(Q))$ during optimization/tuning. The test sets are then used to estimate out-of-sample performance in the comparative experiments.

A simple "sweep" [23] procedure over a discretized set of selections for $|G|, \hat{m}, \hat{r}, \beta, b$ is used to tune EMSCO. A scalarization similar to $\mathscr{E}(\cdot)$ but with a population-independent measure of inverse cost ($(1 - \frac{g_3^*}{\sum \mathcal{C}_i})$) is used to compare performance of parameter combinations. To ensure this manuscript is self-contained, and to promote reproducibility, the parameters selected for each experiment are listed explicitly in the next section. Readers may note that the parameters returned for each instance are quite similar, demonstrating EMSCO's low variance with regard to parameter changes. In light of this, if tuning is computationally expensive, we suggest $\hat{m} = \frac{1}{n}$, $\hat{r} = 0.8$, $|G| = 300$, $b = 0.2$, and $\beta = 2$ as default parameters.

A. Data Sets

Data sets are chosen to provide a robust evaluation of the methods and to include pathologies occurring frequently in real data. Many deployment scenarios in which budgeted learning is applied rely on a fairly small number of sensors/features for classification [2], and we accordingly restrict our experiments to data sets with at most fifty features.

Feature costs are assigned with scaling functions dependent on an assigned *cost class*—an integer value representing the cost incurred while acquiring the respective feature. The cost assignment schemes for each experiment are designed to represent a wide variety of potential deployment scenarios. Likewise, four distinct confidence thresholds ($\hat{p} = 0.55, 0.65, 0.75, 0.85$) are considered. These values were selected to provide a range of confidence thresholds that represent administrators who are only *mildly* risk averse (low confidence thresholds) to those who are *very* risk averse (high confidence thresholds) and are willing to sacrifice substantial coverage in order to achieve high accuracy.

1) Synthetic50 Data Set: The Synthetic50 data set was constructed using `sklearn`'s `make_classification()` function [24]. The data set consists of 50 features and 4000 records. Binary labels are assigned to the records proportionally (50-50). In this experiment, we account for cases where all features are of roughly equal cost. More precisely, we set all $T_i = 1$, and use scaling function $h(T_i) = 10T_i$. We set `n_informative=25`—a parameter of `make_classification()` specifying the number of "informative", non-redundant features[4]. All features are continuous, and four clusters are present in the data (two for each

---

[4]For more information regarding sklearn's make_classification() tool, please refer to https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

Cost Class	Pima	Heart	Credit	Synthetic50
1	$(1,8)$	$(1,8,11,12)$	$(4,6,9,11,12,13)$	$(1,2,\ldots,50)$
2	$(3,4,6,7)$	$(2,3,4,5,6,7,9,10)$	$(1,3,5,7,10,14)$	$()$
3	$(2,5)$	$()$	(8)	$()$

Feature Acquisition Cost Classes. Each entry corresponds to the set of features assigned to the cost class in the leftmost column. For Pima, Credit, and Heart, features are as indexed on the UCI Machine Learning Repository [22]. Synthetic15 is excluded from this table since it employs a different protocol for assigning feature costs.

Data Set	k=2	k=3	k=4
Heart	24	28	33
Credit	47	141	171
Pima	18	42	65

TABLE I: Global Pareto Frontier Sizes

label). In this data set, we account for less cautious settings and set $\hat{p} = 0.55$. Tuning led to mutation rate 0.05, crossover rate 0.80, elite population percent 0.2, population size 300, mutation bias parameter 2.5.

2) Synthetic15 Data Set: The Synthetic15 data set consists of 8000 records and 15 features. 12/15 features are deemed informative using the `n_informative` parameter of sklearn's `make_classification()` function. Addressing scenarios with a *highly* conservative decision maker willing to sacrifice coverage for improved accuracy, we set $\hat{p} = 0.85$. To account for another possible cost dynamic, we do not assign cost *classes* for this data set, but instead use a noisy, rounded function of the Gini importance (denoted in the following equation as z_i) for $\mathcal{F}_i$. Costs for each feature are assigned as $\mathcal{C}_i = \max(1, round\,(100z_i + randint(-1,1)))$, where $randint(-1,1)$ returns either 1 or -1 with equal probability. This procedure yields cost set $\mathcal{C} = [6,8,4,8,8,1,9,10,6,1,9,9,4,1,7]$.

As done for the Synthetic50 data set, we use a balanced class distribution (50% of records have label '1', and 50% have label '0') and two clusters for *each* label. Furthermore, to make the classification task more challenging, we increase proximity between the two possible classes by setting `class_sep` $= 0.85$ and assign random labels to 2% of the data. Mutation rate 0.075, crossover rate 0.80, elite population percent 0.2, population size 250, and mutation bias parameter 2.0 were returned by the tuning procedure.

3) Pima Diabetes Data Set: The Pima Diabetes data set contains 768 records and is a popular benchmark for evaluating performance of classifiers. Features are binary, integer, or real-valued. Cost classes $T_i \in \{1,2,3\}$ were assigned to each feature depending on the time and complications inherent in conducting the individual tests. For example, 'Age' is assigned $T_i = 1$, and 'Plasma glucose concentration' is assigned $T_i = 3$. For this data set, we use the linear scaling function $h(T_i) = 100T_i$. The tuning procedure returned mutation rate 0.075, crossover rate 0.80, elite population percent 0.2, population size 300, mutation bias parameter 2. A 0.65 confidence threshold is used for this data set to account for moderately strict classification instances. We refer to this data set as "Pima" for the remainder of this paper.

4) Australian Credit Approval (Statlog) Data Set: This data set is available on the UCI Machine Learning Repository [22] and consists of 14 features given in 690 credit applications to a large bank. The target for this data set is binary. Feature descriptions are not provided given the sensitive nature of the data, but the values are designated as continuous or categorical.

This data set contains missing values.

Three cost classes are assigned to the features according to their Gini importance [25], where features of greater importance are assigned greater cost classes. To account for scenarios with greatly varying feature costs, the cost-scaling function is set as $h(T_i) = 10^{T_i}$. Mutation rate 0.075, crossover rate 0.80, elite population percent 0.2, population size 300, and mutation bias parameter 2.5 were returned by the sweep tuning procedure. A 0.75 confidence threshold is applied to predictions. This data set is referred to as "Credit" in this paper.

5) Heart Failure Clinical Records: This data set is available on the UCI Machine Learning Repository and was compiled from 300 patients [22]. The data set consists of 12 pertinent clinical features that are a mix of categorical, integer, and real values. With the guidance of the data set's author, two time classes were assigned to each feature representing the acquisition cost. For this data set, the cost-scaling function is set as $h(T_i) = 10^{T_i}$, where T_i denotes the cost class ('1' or '2') of the ith feature. Mutation rate ($\hat{m}$) 0.075, crossover rate ($\hat{r}$) 0.75, elite population percent (b) 0.2, population size ($|G|$) 300, and mutation bias parameter (β) 2.0 were returned by the tuning procedure. To account for conservative classification environments, a 0.75 confidence threshold is used to determine early-exit and reject decisions. We refer to this data set as "Heart" for the duration of this paper.

B. Global Optimization Experiment

These experiments assess EMSCO's ability to find global optima in a large solution space during the learning phase. This first requires establishing a "ground truth" fitness landscape from which we can derive the Pareto frontier to use as a point of reference for the solutions in EMSCO's populations. To this end, we measure $(g_1(Q), g_2(Q), g_3(Q))$ for all $Q \in \mathcal{S}_{(n,k)}$. Since this represents a significant computational burden, we consider only the Heart, Credit, and Pima data sets with $k \leq 4$. Running these experiments took roughly three weeks with a 32-core Intel Xeon E5-2650 and 256G RAM.

After establishing the fitness landscape and computing the respective Pareto front for each data set and stage-count ($k = 2 \ldots 4$), EMSCO was run 50 times. In every run, at each hth generation, the number of unique globally optimal solutions X_h (determined by comparison with the predetermined Pareto front) was recorded. These values were then averaged with respect to the 50 samples to obtain $\bar{X}_h$ and depicted in Figure 4.

EMSCO shows itself capable of quickly finding globally Pareto optimal solutions in very large search spaces (e.g., Credit $k = 4$ with 250 million solutions). In the Heart and Pima experiments, EMSCO attained nearly all global optima—an encouraging result when considering the Heart, $k = 4$
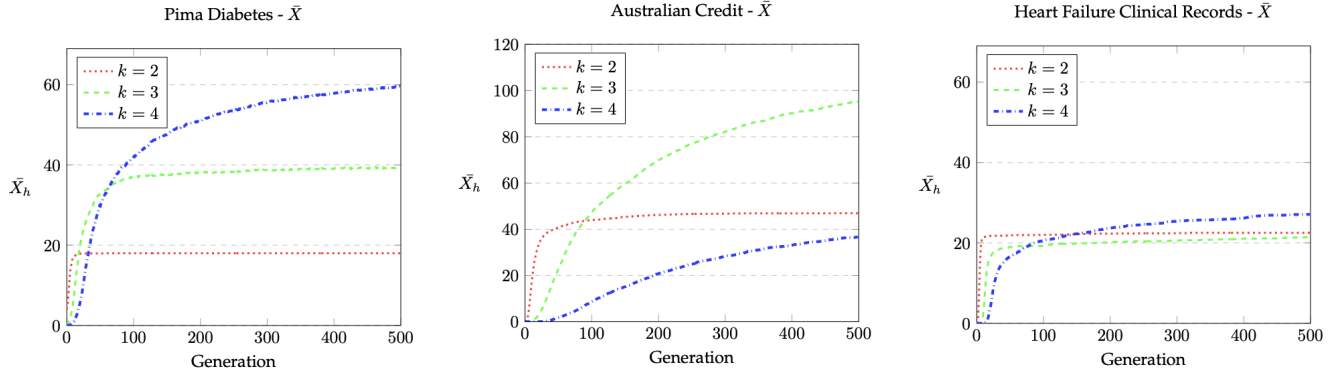
Fig. 4: *Unique, Globally Non-Dominated Solutions at each hth Generation.* $\bar{X}_h$ denotes the sample mean (50 trials) for the count of unique globally optimal solutions present in EMSCO's hth generation during optimization.

search space contains nearly 15 million total solutions and only 33 globally non-dominated solutions (Table I). Another notable pattern evident in these results is the monotonicity of the curves, illustrating Property I empirically.

C. Comparative Experiment Design

These experiments evaluate the performance of EMSCO against related approaches. To the best of our knowledge, EMSCO is the first *budgeted* classification method that leverages prediction confidence to improve accuracy by rejecting uncertain predictions. A direct comparison between EMSCO and well-established alternative methods of the same exact nature is therefore not possible. To argue the merit of our perspective and EMSCO specifically, we then compare against a variety of closely-related methods.

Greedy Miser (GM) [3] is a variant of stage-wise regression [12] with a feature-budgeted loss function. In this method, regression trees are added iteratively to form a cost-effective ensemble classifier. *Cost-Sensitive Tree of Classifiers* (CSTC) [5] builds a tree of classifiers optimized for a specific sub-partition of the input space. The aim is to ensure that inputs are classified using only the most pertinent features defined for particular regions of the input space. Doing so reduces unnecessary feature extraction while maintaining accuracy. GM and CSTC do not apply reject options as these methods are non-selective by design and their confidence scores do not represent true class probabilities[5]. As done in their respective papers, GM and CSTC's parameters were tuned during validation using grid search on a large discretized set.

A *Cost-Ordered T-Stage Classifier* (CO-T) is considered as a heuristic utilizing a popular cost-ordered perspective for ordering features in budgeted systems [2], [7]. In this setup, a stage is added for each of the T increasing cost classes. A confidence-based reject option at the final stage is applied.

We also consider Cost-and-Coverage-Tuned LASSO method (CaCT LASSO) that performs feature selection during training and then evaluates all inputs in a single stage at test-time with the chosen features before applying a confidence-based

---

[5]In contrast, EMSCO utilizes logistic regression at each stage which is considered to be a well-calibrated model with interpretable class probabilities [26]

reject decision. Cost is then computed as the sum of C_i such that the respective regression coefficient for feature $\mathcal{F}_i$ is non-zero. This method is tuned on the validation set for $\lambda \in \{0, 0.1, 0.2, 0.3, \ldots, 10\}$ while accounting for accuracy, cost, and coverage. Note, CO-T and CaCT LASSO methods apply logistic regression at each stage to classify inputs.

1) EMSCO Performance Measurement: Experiments for each data set consist of 50 EA runs. For each of these runs, at the terminal generation, the chromosome with maximum fitness is returned and appended to a list. When the 50 EA runs are finished, the chromosomes comprising this list are used to compute average g_1, g_2, g_3^* values *on the test set*, marking the performance listed in Table II. Due to the stochastic nature of evolutionary algorithms, a 95% margin of error (ϵ_{EA}) for performance is displayed in Table III. Note, EMSCO is run with the parameters listed in Section V-A and $max\_iter = 150, k = \min(round(\frac{n}{2}), 10)$. Average feature acquisition cost (g_3^* as defined in Section III-C3) over the test set is used to measure efficiency of the methods, since g_3 only carries meaning within a particular population of solutions.

2) Comparison Experiment Results: Objective values achieved by each method are listed in Table II. Among others, a few notable results arise:

- EMSCO is non-dominated in all experiments, performing better than alternatives in at least one objective. Moreover, EMSCO Pareto dominates GM and CSTC in the Synthetic50 experiment and nearly dominates these methods in the Pima experiment as well, if not for the 1% reduction in coverage.
- EMSCO offers greater accuracy than the budgeted benchmarks (GM and CSTC) in every experiment. Likewise, in 3/5 experiments (Pima, Synthetic15, Synthetic50), EMSCO also induces substantially lower processing cost than these methods.
- EMSCO surpasses every method in at least 2/3 objectives in a majority of experiments (Pima, Synthetic15, and Synthetic50).

Such results suggest that EMSCO is effective in increasing accuracy compared to non-selective budgeted methods while maintaining strong coverage and low cost. However, it is worth noting that in the Credit and Heart experiments, GM

Dataset	Accuracy%					Coverage%					Cost				
	EMSCO	GM [3]	CSTC [5]	CaCT Lasso	CO-T	EMSCO	GM [3]	CSTC [5]	CaCT Lasso	CO-T	EMSCO	GM [3]	CSTC [5]	CaCT Lasso	CO-T
Pima	71.1	68.1	67.3	82	72	99	100	100	83.4	97	242.4	500	624	1100	524
Credit	83.1	76.4	78.1	89.1	81.3	96.1	100	100	80.1	93.2	365.89	107	522	1420	695.8
Heart	83.4	79.2	81.1	88	87.1	89.3	100	100	65	72	102.4	38.6	560	120	255.3
Synthetic50	66.4	65	64.3	82.6	77.6	100	100	100	92.7	97.5	16.47	76.5	72.2	70	500
Synthetic15	90	81.2	85.3	92.4	91.2	66	100	100	52.3	58.5	56.57	75.7	61.3	60	79.7

TABLE II: *Comparative Experiment Results*

Data Set	Accuracy	Coverage	Cost
Pima	0.5%	0.3%	8.2
Credit	0.7%	0.1%	17.7
Heart	0.4%	1.2%	5.1
Synthetic50	0.6%	0.1%	4.3
Synthetic15	0.1%	0.1%	2.4

TABLE III: 95% margin of error values for EMSCO in Table II.

offers considerably lower processing cost. In less conservative settings where accuracy is not paramount, this performance could be favored over EMSCO's depending on the decision maker's preferences.

VI. CONCLUSION

We have introduced EMSCO as a novel approach to manage the objectives of both *selective and budgeted* classification. Experiments conducted on a variety of data sets, confidence thresholds, and cost assignment protocols suggest that the proposed method is capable of finding and maintaining global optima in large solution spaces. Additionally, in multiple experiments, EMSCO is able to simultaneously offer lower cost and greater accuracy than popular budgeted benchmarks.

Future work may consider adding a feature selection option to EMSCO that alters the chromosome representation to include a value (e.g., '-1') signifying the respective feature should be excluded. Doing so may promote model simplicity and improve scaling to high-dimensional data. Additionally, since real-world scenarios often impose disproportionate penalties for false-positives and false negatives, use of distinct confidence thresholds (one for each possible label) may prove useful.

REFERENCES

[1] S. Ji and L. Carin, "Cost-sensitive feature acquisition and classification," *Pattern Recogn.*, vol. 40, no. 5, pp. 1474–1485, May 2007. [Online]. Available: https://doi.org/10.1016/j.patcog.2006.11.008

[2] K. Trapeznikov, V. Saligrama, and D. Castañón, "Multi-stage classifier design," *Machine Learning*, vol. 92, no. 2-3, pp. 479–502, May 2013. [Online]. Available: https://doi.org/10.1007/s10994-013-5349-4

[3] Z. Xu, K. Weinberger, and O. Chapelle, "The greedy miser: Learning under test-time budgets," 2012.

[4] M. J. Kusner, W. Chen, Q. Zhou, Z. Xu, K. Q. Weinberger, and Y. Chen, "Feature-cost sensitive learning with submodular trees of classifiers," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. AAAI Press, 2014, p. 1939–1945.

[5] Z. E. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle, "Classifier cascades and trees for minimizing feature evaluation cost," *Journal of Machine Learning Research*, vol. 15, no. 62, pp. 2113–2144, 2014. [Online]. Available: http://jmlr.org/papers/v15/xu14a.html

[6] F. Nan, J. Wang, and V. Saligrama, "Pruning random forests for prediction on a budget," in *Advances in Neural Information Processing Systems 29: NeurIPS 2016, Barcelona, Spain*, 2016, pp. 2334–2342.

[7] J. Wang, K. Trapeznikov, and V. Saligrama, "An LP for Sequential Learning Under Budgets," vol. 33. PMLR, 22–25 Apr 2014, pp. 987–995. [Online]. Available: http://proceedings.mlr.press/v33/wang14b.html

[8] N. H. Hamilton, S. McKinney, E. Allan, and E. W. Fulp, "An efficient multi-stage approach for identifying domain shadowing," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[9] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with bert," 2019.

[10] J. Janisch, T. Pevný, and V. Lisý, "Classification with costly features as a sequential decision-making problem," *CoRR*, vol. abs/1909.02564, 2019. [Online]. Available: http://arxiv.org/abs/1909.02564

[11] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features." in *CVPR (1)*, 2001. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2001-1.htmlViolaJ01

[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451

[13] F. Nan and V. Saligrama, "Adaptive classification for prediction under a budget," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4730–4740. [Online]. Available: http://papers.nips.cc/paper/7058-adaptive-classification-for-prediction-under-a-budget.pdf

[14] D. Andrade and Y. Okajima, "Adaptive covariate acquisition for minimizing total cost of classification," 2020. [Online]. Available: https://arxiv.org/abs/2002.09162

[15] R. El-Yaniv and Y. Wiener, "On the foundations of noise-free selective classification," *Journal of Machine Learning Research*, vol. 11, no. 53, pp. 1605–1641, 2010. [Online]. Available: http://jmlr.org/papers/v11/el-yaniv10a.html

[16] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," 2017.

[17] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[18] D. Lichtblau and C. Stoean, "Cancer diagnosis through a tandem of classifiers for digitized histopathological slides," *PLoS One*, 2019. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/30650087/

[19] B. N. Patel, L. B. Rosenberg, G. Willcox, D. Baltaxe, M. Lyons, J. A. Irvin, P. Rajpurkar, T. J. Amrhein, R. Gupta, S. S. Halabi, C. Langlotz, E. Lo, J. G. Mammarappallil, A. J. Mariano, G. Riley, J. Seekins, L. Shen, E. Zucker, and M. P. Lungren, "Human–machine partnership with artificial intelligence for chest radiograph diagnosis," *NPJ Digital Medicine*, vol. 2, 2019.

[20] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer, 2015, gebeurtenis: 2nd edition.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[23] K. De Jong, "Parameter setting in eas: a 30 year perspective," in *Parameter setting in evolutionary algorithms*. Springer, 2007, pp. 1–18.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[25] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in Neural Information Processing Systems*, vol. 26, 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/e3796ae838835da0b6f6ea37bcf8bcb7-Paper.pdf

[26] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.