EUSC: A Clustering-based Surrogate Model to Accelerate Evolutionary Undersampling in Imbalanced Classification

Hoang Lam Le[a,*], Dario Landa-Silva[a], Mikel Galar[c], Salvador Garcia[b], Isaac Triguero[a]

[a] *Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, Nottingham NG8 1BB, United Kingdom*
[b] *Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain*
[c] *Department of Automatics and Computation, Universidad Pública de Navarra, Campus Arrosadía s/n, 31006 Pamplona, Spain*

Abstract

Learning from imbalanced datasets is highly demanded in real-world applications and a challenge for standard classifiers that tend to be biased towards the classes with the majority of the examples. Undersampling approaches reduce the size of the majority class to balance the class distributions. Evolutionary-based approaches are prominent, treating undersampling as a binary optimisation problem that determines which examples are removed. However, their utilisation is limited to small datasets due to fitness evaluation costs. This work proposes a two-stage clustering-based surrogate model that enables evolutionary undersampling to compute fitness values faster. The main novelty lies in the development of a surrogate model for binary optimisation which is based on the meaning (phenotype) rather than their binary representation (genotype). We conduct an evaluation on 44 imbalanced datasets, showing that in comparison with the original evolutionary undersampling, we can save up to 83% of the runtime without significantly deteriorating the classification performance.

Keywords: Data Preprocessing, Evolutionary undersampling, Surrogate models, Imbalanced classification, Fitness approximation

1. Introduction

Learning from skewed data is a challenge arising in multiple domains such as bioinformatics, business management, or network analysis [1, 2, 3]. Focusing on two-class datasets, the problem happens when samples from the minority class (usually the class of interest) are highly outnumbered

[*] Corresponding author
Email address: Hoang.Le@nottingham.ac.uk (Hoang Lam Le)

by the counterparts from the majority class [4, 5, 6]. The majority and minority classes are typically known as the 'negative' and 'positive' classes, respectively.

In skewed datasets, canonical classification algorithms may be biased towards the majority class, being unable to appropriately predict examples from the minority class [7, 8]. Solutions tackling this difficulty can be grouped into data preprocessing [9, 10, 11] and algorithmic modification [12, 13]. Those operating at the algorithmic level modify the learning algorithms to make them aware of the imbalanced situation at the learning stage, while those at data-level preprocessing intervene in the cardinalities of positive and negative classes to make them less critically unequal. Cost-sensitive learning [14, 15] and ensemble-based methods [16, 17, 18] have also become very popular. Cost-sensitive techniques can be considered algorithm level modifications that try to learn more characteristics of minority class examples by incorporating a higher cost to their misclassification. Ensemble-based methods usually combine an ensemble learning algorithm (e.g. Bagging, Boosting) [16] with one of the mentioned approaches (e.g. data preprocessing [19], cost-sensitive techniques [20]) to establish a combination of multiple base classifiers.

Data-level preprocessing solutions consist of undersampling (concerned with eliminating redundant examples in the majority class [21, 22]), oversampling (generates new artificial data for the minority class [9, 10, 23]), and hybrid methods (a combination of the previous two) [24]. While all approaches are proved effective in many studies, oversampling and hybrid methods tend to generate more data, which may result in a higher computational cost. Undersampling, on the other hand, aims at reducing the data size, which is more advantageous when employed in large datasets or big data scenarios [25]. Among other strategies for undersampling, Evolutionary Undersampling (EUS) [22], an evolutionary instance selection strategy [26] for imbalanced classification, has been demonstrated to be very effective in multiple studies, especially in combination with Ensemble-based approaches [27, 28, 29].

EUS is an example of optimisation techniques to improve machine learning processes [30]. This does not only help to balance the distribution of classes but, as an instance selection approach, this also allows us to remove noisy instances in the majority class, which is a common issue in real-world applications [31, 32]. In particular, the EUS algorithm performs a binary search guided by an Evolutionary Algorithm (EA) to optimise the selection of (training) examples from the majority class that improves the classification performance. The chromosome quality is measured by classifying the entire training dataset based on the preprocessed set represented by the chromosome.

Similarly to most previous works (e.g. in the original EUS [22]), in this paper we adopt the Nearest Neighbour (NN) [33] rule as base classifier. The resulting preprocessed dataset, however, should be ready to be used by any classifier.

Despite its effectiveness, the EUS method is typically very time-consuming, especially in large datasets, due to the cost associated to fitness evaluation. Further advancement of the EUS requires two conditions: (1) reduce the processing time and (2) still guarantee a high classification performance. In the recent literature, the processing time of EUS (and other instance selection/generation based approaches) is being reduced by using distributed approaches in big data platforms [25, 34], increasing the need for a larger number of computing nodes.

In this work, we are interested in reducing the computational cost of EUS by using fitness approximation approaches [35, 36], such as surrogate models [37, 38, 39], which could accelerate the expensive computation of the classification performance of each chromosome. Surrogate-based methods allow us to reduce the computational cost of search algorithms, as opposed to parallelisation techniques that merely focus on reducing processing time. This kind of approach has been widely investigated in various problems employing evolutionary optimisation techniques [40, 41, 42, 43], following different approaches, such as fitness inheritance and machine learning methods [36]. Existing methods are usually designed for problems in the continuous search landscape. However, methods for combinatorial domains have been under-explored [44, 45] due to the complexity of the field which requires domain knowledge to apply fitness approximation.

In the field of evolutionary instance selection (for imbalanced and standard classification), primitive approaches for fitness approximation have been employed to reduce the computational cost (windowing [46]) and processing time (stratification [47]). The underlying idea of these methods is to consider subsets of training data for fitness evaluation, reducing the cost on larger datasets. Whilst these approaches are important in addressing large datasets with EUS, the use of surrogate models for evolutionary instance selection is an under-developed area in the literature that can highly reduce the computational cost of this kind of search technique.

In this paper, we propose a two-stage clustering-based surrogate model for EUS (EUSC) that allows us to compute fitness values faster. As opposed to windowing or stratification approaches, EUSC considers the entire training data when computing fitness values. However, it only performs real evaluations for a limited number of chromosomes. First, a preliminary clustering stage of majority examples allows us to transform binary chromosomes into real coding chromosomes that represent the overall location of the instances selected in a solution. Then, in every generation, the entire population is clustered using the new intermediate chromosome representation to approximate the fitness values based on their similarity and imbalanced ratio. To the best of our knowledge, this is the first surrogate-based model for EUS, and one of the very few surrogate models that work on a combinatorial problem [45]. The main contributions of this work are:

- We investigate the challenge of devising surrogate models for a combinatorial/binary optimisation problem (instance selection for undersampling). We discuss the weaknesses of using the Hamming distance to compute the similarity between binary chromosomes, which would not reflect well how similar two solutions are. The main novelty of this work lies in developing a means to perform that similarity computation based on the phenotype of the chromosome.
- We propose a clustering-based surrogate model for EUS which highly reduces the computational cost of this method, speeding up the algorithm without reducing significantly its classification performance. Thus, the main contribution is in the acceleration of the fitness evaluation of an instance selection method in the context of imbalanced classification. To validate its performance, we explore different variants of the proposed method considering multiple factors affecting the model, then analysing empirically their effectiveness. We compare the proposed surrogate model against the original EUS algorithm and EUS with windowing evaluation on 44 standard imbalanced datasets with various imbalance ratios. In comparison with the windowing strategy, the results obtained show that EUSC does not only reduces runtime enormously, but it also provides a high-quality solution which does not significantly decrease classification performance in comparison to the original EUS.

The rest of this paper is organised as follows. In Section 2, we introduce related works, consisting of EUS, recent work on accelerating fitness evaluation in evolutionary search, and existing approaches to speed up EUS. Next, Section 3 describes our proposal in detail. In Sections 4 and 5, we introduce the experimental framework used in this study, and the results with associated analysis, respectively. Finally, we make several concluding remarks in Section 6.

2. Background

This section presents background information about EUS for imbalanced classification (Section 2.1) and different techniques to accelerate EAs (Section 2.2). Finally, we discuss existing approaches to accelerate evolutionary instance selection techniques (Section 2.3).

2.1. Evolutionary undersampling for imbalanced classification

In learning with skewed data, balancing the class distributions can alleviate the bias of standard classification algorithms towards the majority class. Among other approaches, undersampling is an interesting alternative for large datasets as they reduce the number of samples in the majority class (contrary to oversampling which generates artificial minority class samples), consequently, enabling standard algorithms to be capable of identifying examples from both classes more accurately. Undersampling techniques can inherit from instance reduction methods which were initially designed for other preprocessing purposes in learning methods (instance selection and generation [26, 48]).

The simplest way to obtain a balanced subset of the original data is to randomly undersample the majority class [10]. However, this non-heuristic approach may discard important data in the negative class due to the randomness in this mechanism. EUS [22] on the other hand, is an evolutionary instance selection algorithm that carries out a heuristic search to optimise the subset of samples that are selected, and thus can increase the accuracy of a classifier on both classes. The search is guided by an EA, namely CHC [49], which is efficient at maintaining the balance of exploration and exploitation by applying different mechanisms such as incest prevention, reinitialisation of the population when the search does not progress and the competition among parents and offspring for selecting the elitist. The reduced set is evolved from undersampled instances until the highest performance computed by a fitness function is achieved or stopping conditions are met. EUS can achieve two goals which are the balance of samples between classes and high classification accuracy when the selected negative samples are the most representative.

Assuming binary classification, a formal specification of the problem is the following: A two-class dataset has N^- negative class instances and N^+ positive class instances. Let x_i be an instance in the dataset where $x_i = (x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_m}, x_{i_\omega})$, with x_i belonging to a class given by x_{i_ω} and an m-dimensional feature space in which the feature value at the k^{th} position of the i^{th} sample is denoted as x_{i_k}. In the EUS approach, a candidate solution is a binary chromosome in which each gene takes a value of $\{1, 0\}$ to represent the presence or absence of an instance x_i, respectively. As only majority class instances are examined for elimination, the size of a chromosome is thus equal to N^-. Positive samples are automatically concatenated with the selected negative examples to form a final reduced set RS for classification. The representation of a chromosome in the EUS algorithm is expressed as: $chr_j = (v_{x_1}, v_{x_2}, v_{x_3}, \ldots, v_{x_{N^-}})$ where $v_{x_i} \in \{1, 0\}$ indicates whether sample x_i is included or not.

EUS maintains a population of NP chromosomes that are assessed and ranked based on their quality which considers two main factors: classification performance and class imbalance. The fitness function uses RS to classify the entire training dataset. Similarly to most previous works, in this paper we adopt the NN rule [33] as base classifier. However, in imbalanced classification, traditional accuracy measures are no longer valid as they neglect the fact that there is an imbalanced class distribution. Two commonly used alternatives are geometric mean (GM) [50] and the area under the curve (AUC) of Receiver Operating Characteristic [51]. Both measures have been extensively and interchangeably used in many experimental studies of imbalanced classification. In this paper, we will focus on the GM, defined in Equation 1 to report the classification performance. This is not only because it has been used in many experimental studies on imbalanced classification [5, 22, 25], but also because it can reflect the balance between the true positive rate (TP_{rate}) and true negative rate (TN_{rate}) at the same time and therefore the contribution on either class does not have a higher impact than that of the other.

$$GM = \sqrt{\mathrm{TP}_{rate} \times \mathrm{TN}_{rate}} \tag{1}$$

The complete fitness function for a chromosome chr_j looks like this:

$$f_{chr_j} = \begin{cases} GM_{chr_j} - \left|1 - \frac{N^+}{s^-}\right| \cdot P & \text{if } s^- > 0 \\ GM_{chr_j} - P & \text{if } s^- = 0, \end{cases} \tag{2}$$

where s^- is the number of selected negative instances and P is a penalisation factor that focuses on the balance between both classes. P is typically set to 0.2 as recommended by the authors, since it provides a good trade-off between both objectives.

The time required to evaluate the quality of each chromosome highly depends on the size of the training set. In this work, we are interested in developing a fast EUS that can quickly estimate the fitness values of chromosomes without misleading the search.

2.2. Accelerating fitness evaluation in evolutionary search

In many optimisation problems, the evaluation of a solution may have a high computational cost due to the function's complexity or massive calculation. In the literature, numerous studies have been performed to speed up fitness evaluations [36, 52, 37]. Broadly speaking, we can find delta evaluation approaches and fitness approximation. Delta evaluation is a way of computing only the different parts between two solutions [53]. It can make use of previously evaluated similar regions and reuse those parts in the evaluation of a new individual. The strategies of delta evaluation are based on analytical computation to identify which part in the expression needs re-calculating. For example, in a timetabling problem [54], instead of evaluating every timetable as only small changes are made between one timetable and the next, it is possible to merely compute the changes and update the previous cost with the value of that calculation.

Extensive studies have been proposed in the family of fitness approximation from a simple approach like fitness inheritance to advanced techniques like machine learning methods [36, 55, 37]. Fitness inheritance is initially inspired by the idea that an offspring can also inherit a fitness value from its parents, not only its own genes. Thus, its quality can be obtained from where it derives from instead of through a function. Two classical approaches for fitness inheritance [56] are the averaged inheritance (adopt the average fitness of its parents) and proportional inheritance (fitness is weighted based on the amount of genetic material taken from each parent). These ideas were later further investigated on several studies using for example fitness sharing in multi-objective optimisation problems [57, 58], or using fitness inheritance with Bayesian optimisation [59].

Machine learning techniques such as clustering and supervised learning can be used in numerous ways to alleviate fitness evaluation. Clustering algorithms including hierarchical clustering, partition clustering, and overlapping clustering, typically aim to decrease the number of original function evaluations [60]. These approaches split the entire population (based on the chromosome representation) into a number of groups by a clustering algorithm, and then the chromosomes closest to the clusters' centres are evaluated by the exact function, while other cluster members are approximated according to their distance to the evaluated solutions [61, 62, 63]. However, this approach is mostly applicable for continuous optimisation problems only, and still challenging in combinatorial search space due to the problem of computing the correlation among solutions to interpolate the fitness value [45].

Supervised learning techniques aim to create a surrogate model that can approximate the fitness function by prediction. The model is adjusted based on the known data points accumulated from the evaluation history. Naturally, most surrogate models are assumed spatial models which means the prediction task is about exploiting accepted spatial relations such as a smooth change in a response surface between the fitness values of a query point and known data-points [64]. In other words, a data-driven model is constructed with the assumption that there is continuity among data points, at which a small variation in decision variables will cause a smooth change in the response space. This makes surrogate models naturally suited to continuous optimisation problems, while not easily applicable to combinatorial optimisation problems [44] because the response in combinatorial space does not necessarily vary smoothly when the discrete variables produce a minor variation. Hence, choosing an appropriate metric to express the correlation between chromosome representation and its quality has been a difficult task, which makes combinatorial landscape analysis significantly challenging. More discussion can be found in [44, 45].

Both clustering and supervised learning approaches for surrogate models are under-investigated when dealing with combinatorial or binary optimisation problems. In this paper, we will focus on a clustering-based approach to approximate fitness values that does not require collecting enough data to train a machine learning algorithm. However, the ideas proposed in this paper to cluster binary chromosomes may also be useful for supervised approaches.

2.3. Reducing processing time of evolutionary instance selection

Several primitive approaches of fitness approximation have been used for evolutionary instance selection strategies (such as EUS), namely stratification [47] and windowing [46].

The first approach distributes the initial data into several disjoint strata which each stratum still preserves original class distributions. Each stratum is then individually processed by an evolutionary-based strategy to produce different reduced sets (RSs). Finally, all RSs are joined into a final global set. In this approach, the fitness function is not directly applied for the original data, but it is used with each stratified small subset. The ultimate goal of stratification is to deal with the memory consumption limitation rather than speed. The computational cost at evaluation is not reduced in total, but the real fitness function is approximated by the way it is used with a smaller scale of data.

Windowing also splits the training data into several strata with equal class distribution, but it computes the performance on one stratum to represent for the entire initial data. A windowing scheme employs all strata using round-robin policy to estimate chromosomes' fitness during multiple iterations of an evolutionary process. As the data quantity is reduced at each evaluation, the demand for computation is therefore reduced. Despite many positive elements, this approach also has several drawbacks which possibly limit it from extending in applications. Although this method can handle larger datasets, each stratum is empirically limited to no more than tens of thousands of instances [65] in evolutionary-based strategies. Thus, the method can alleviate the burden of fitness computation in relatively large scale datasets [47] but not in extreme scenarios like big data. As such, this approach does reduce the computational cost of an evolutionary instance selection and was first used for EUS in [25]. The windowing approach for EUS is dependant on the imbalanced ratio (IR) (defined as the ratio of the number of instances from the negative class and the positive class). For example, in a two-class dataset with 100k instances and an $IR < 9$, each evaluation of any chromosome is processed with more than 20k samples of a stratum (10k samples from each class). Thus, the lower IR, the more computation is required.

In the big data context, current parallelisation approaches based on MapReduce aim at reducing the processing time by splitting the datasets into several disjoint blocks (similarly to the stratification approach) that are handled in parallel [66]. In [25], a two-level parallel scheme combining MapReduce and windowing was proposed for EUS. This approach reduces both computational costs and processing time, but relies on windowing (for imbalanced sets) to reduce the computational cost. Note that windowing could easily be replaced by the proposed EUSC.

3. EUSC: Evolutionary undersampling with a clustering-based surrogate model

In this section, we describe the proposed EUSC framework in detail. Section 3.1 motivates the proposed approach, detailing the challenges to perform clustering-based fitness approximation. Finally, Section 3.2 provides a detailed description of the proposed model.

3.1. Challenges to perform a cluster-based fitness approximation

As mentioned in Section 2.2, surrogate models usually rely on distance measures between solutions to perform fitness approximations. As EUS encodes solutions as binary chromosomes, the Hamming distance appears as a natural option to measure the similarity among these binary chromosomes. However, if we do so, we would be expecting this distance metric to reflect the change of chromosomes' representation in the fitness landscape, so that, the fitness of chromosomes varies according to the change of the Hamming distance. In preliminary experiments, we observed that this option was not feasible and performed poorly.

In (imbalanced) classification, some instances may be very important when performing classification, due to their location in the classification space (e.g. those instances in the decision boundaries between classes are typically very important). Using the genotype of the chromosome, the Hamming distance considers the presence of all instances with equal merit, ignoring and neglecting the degree of difference that the actual feature values of a particular example may reflect in the fitness computation, misleading the fitness inference. For example, a chromosome with only one gene swapped would be very similar according to this metric, but it may lead to a great difference in classification performance.

This motivates us to propose the EUSC algorithm which can address the challenge of computing differences among different solutions. The main contribution of this work lies in establishing a bridge connecting the chromosomes' representation and their quality in the fitness landscape. The use of estimated fitness values might seem to be linked to a reduction of the performance. However, currently we need to remember two main considerations for any existing instance selection algorithm [26]: (1) the use of training data to compute fitness values is in itself an approximate way to measure the quality of a solution and determine how well the resultant RS allows us to learn a concept; (2) the search algorithm may overfit the training data. With these ideas in mind, we aim to develop a surrogate method capable of reducing the computation cost of EUS without misleading the search.

3.2. Two-stage clustering-based surrogate model for EUS

The main advantage in the proposed EUS with clustering is the ability to quickly obtain the fitness value of a chromosome without always computing its classification performance. To do so, we propose a two-step process based on clustering, which is represented in Figure 1. Note that the proposed surrogate model has been integrated with EUS, which was based on the CHC algorithm. However, the ideas proposed here could be implemented in any evolutionary-based algorithm for undersampling (or instance selection). The added computational cost of the proposed method will be introduced twice as we progress into the two-step process based on clustering.

The complete pseudo-code for EUSC is presented in Algorithm 1 which emphasises (in boldface) the main modifications to the EUS described in Section 2.1. Our method follows the same structure of the CHC algorithm [49] that maintains an excellent balance between exploration and exploitation. The CHC algorithm has several important characteristics:

- It uses a heterogeneous uniform cross-over (HUX) for the combination of two chromosomes aiming at maximising the differences of the offspring from their parents (Algorithm 1, line 11).

- It applies an incest prevention mechanism in the combination with the HUX operator to impede the cross of two parents if they are too similar in terms of their Hamming distance.
- It has a restart mechanism to reinitialise the entire population when the evolution does not progress (Algorithm 1, lines 19-21).

3.2.1. Phase 1: Intermediate form for chromosomes

In EUS, the original binary representation for chromosomes is helpful to determine which samples are selected/discarded. However, as stated before, one of its major drawbacks is that it does not represent the real phenotype of a chromosome, which is the actual position of the selected instances that interferes significantly in the classification performance.

Our first step to develop a surrogate model for EUS is related to transforming this binary representation into a real-coding one. Note that it is key that this process is very quick to really take advantage of a surrogate model (rather than using a real fitness evaluation). The real-coding intermediate form is created in two steps:

- Step 1: Before commencing the EUS algorithm, the training set is first split into positive and negative sample sets. Then, we group all majority class samples into different regions based on their feature values. This step will later allow us to reflect in which area the selected instances are predominately based. In our experiments, we focus on the well-known k-means algorithm to group data into k_1 clusters $\{T_1, T_2, ...T_{k_1}\}$. This is the considered computational cost added to the EUSC model compared to the EUS. Note that this clustering task may be considerably slow for big datasets, however, it is only conducted once. The information of these clusters is used during the search whenever a chromosome is needed to be transformed into an intermediate form.
- Step 2: Whenever we need to approximate fitness values (initialisation or during the evolutionary cycle), binary chromosomes will be transformed into intermediate forms with the support of $\{T_1, T_2, ...T_{k_1}\}$. Note that each training instance was allocated to a cluster in the previous step. Each intermediate form is a vector of k_1 features. Firstly, from the binary

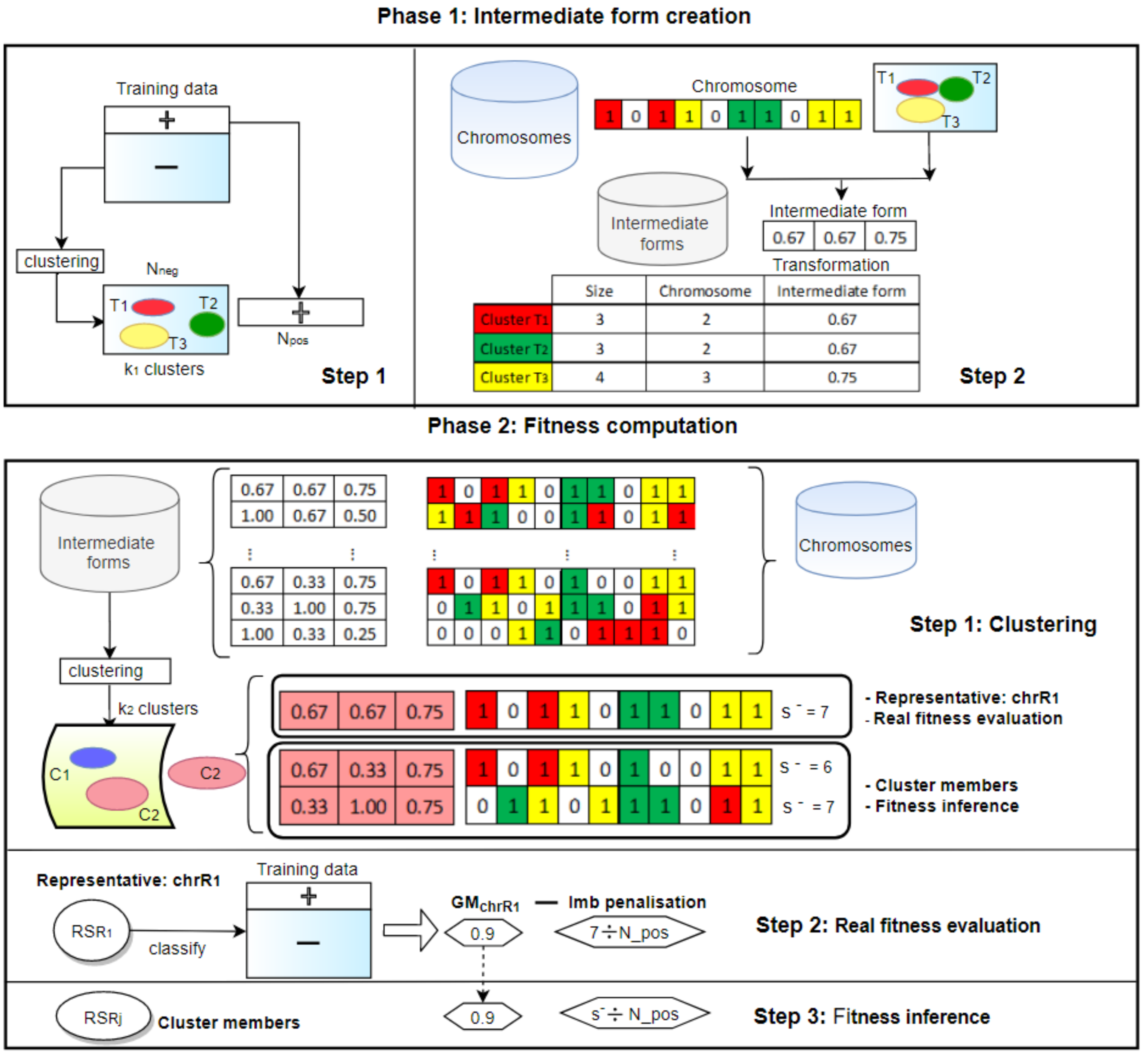

Figure 1: Two phases of an EUSC process: Chromosome transformation at phase 1 and fitness inference at phase 2.

chromosome, we count the number of selected instances (i.e. genes with a 1) that fall into each one of the clusters. Each value of the intermediate chromosome is a real number, obtained from dividing the number of selected instances by the cluster's size. These k_1 values tell us the proportion of selected samples at each region, which is approximate information about the location of the selected samples in the instance space. Thus, this is a simple and fast strategy to obtain an approximate phenotype for each binary chromosome.

Algorithm 1 Evolutionary undersampling clustering-based algorithm - EUSC

```
Require: Training Data (TR)
PS, NS = Split(TR)
$\{T_1, T_2, ...T_{k_1}\}$ = Apply $k_1$-means(NS)
NP = Randomly initialise the population of chromosomes
if Infer_At_Init = True then
    Intermediate_Population ← Transform binary representation (NP)
    Fitness Inference (Intermediate_Population, NP)
else
    Real_Evaluation (NP)
end if
while eval < MAX_EVALUATIONS do
    newPop ← Select M chromosomes from NP based on the diversity in their structure.
    if M > $k_2$ then
        Intermediate_Population ← Transform binary representation (newPop)
        Fitness Inference(Intermediate_Population, newPop)
    else
        Real_Evaluation (newPop)
    end if
    NP = Select best candidates from NP and newPop
    if stagnated = True then                                    ▷ Restart mechanism
        NP = Mutation (Best solution in NP)
        Real_Evaluation (NP)
    end if
end while
```

3.2.2. Phase 2: Fitness computation

After transformation, a chromosome has also a real-coding representation, which allows us to apply fitness inference following similar ideas implemented in the literature for continuous optimisation problems [61]. Algorithm 2 describes in detail the fitness inference mechanism. This stage consists of the following three steps:

Algorithm 2 Fitness Inference

```
Require: Intermediate_Population and Population of Chromosomes that need fitness inference
$\{C_1, C_2, ...C_{k_2}\}$ = Apply $k_2$-means(Intermediate_Population)
$\{chrR_1, chrR_2, ..., chrR_{k_2}\} \leftarrow$ Representatives for each cluster, either Centroid or Random
for each cluster $C_j$ in $\{C_1, C_2, ...C_{k_2}\}$ do
    $GM_{chrR_j} \leftarrow$ Classification performance($RS_{chrR_j}$)
    $Imb_{chrR_j} \leftarrow$ Imbalance penalisation ($chrR_j$)
    $Fitness_{Rep_{C_j}} = GM_{chrR_j} - Imb_{chrR_j}$
     //Infer the fitness of members in $C_j$ using computed $GM_{chrR_j}$
    for each chromosome $chr_m$ in cluster $C_j$ $\{chr1_{C_j}, chr2_{C_j}, ..., chrk_{C_j}\}$ do
        $Imb_{chr_m} \leftarrow$ Imbalance penalisation ($chr_m$)
        $Fitness_{chr_m} = GM_{chrR_j} - Imb_{chr_m}$
    end for
end for
```

- Step 1: To compute the similarity among the current chromosomes considered for evaluation, their intermediate forms are fed into a clustering algorithm (again we focus on k-means in our experiments) to split the population into k_2 clusters $C_1, C_2, ..., C_{k_2}$. This is an additional computational cost added to EUSC in comparison to EUS. However, the size of the data (i.e. a subset of $NP \times k_1$) we are clustering is so small that this time could be considered almost negligible. In this way, the clustering task also indirectly groups the chromosomes into different regions in the binary space. Note that the binary representation is still needed at evaluation time (either real or inferred) to compute the balance between classes (See Equation 2).

- Step 2: Compute the fitness value of only k_2 representatives using the real fitness function defined in Equation 2. Here we examine different approaches for the selection of representatives. In particular, we analyse the effect of selecting them randomly or using the centroid chromosome from each cluster $C_1, C_2, ..., C_{k_2}$. As a result, we have a set of representative chromosomes $\{chrR_1, chrR_2, ..., chrR_{k_2}\}$ for which we can compute their GM values $\{GM_{chrR_1}, GM_{chrR_2}, ..., GM_{chrR_{k_2}}\}$.

- Step 3: In this final step, we can now infer the fitness values for the remaining chromosomes. As defined in Equation 2, the fitness function consists of GM and imbalance penalisation. The GM values of the k_2 representatives will be reused for all the chromosomes belonging to the same cluster. This means that all the members in the same cluster with the representative simply get the representative's GM and thus the cost of classification performance is saved. However, the component of imbalance penalisation can be quickly computed from the binary chromosome for each particular solution. We are aware that transferring the same GM to all members of a cluster may seem to be an oversimplification, and more elaborated solutions will be investigated in the future. However, as we will see in the experimental section, this simple fitness inheritance mechanism allows us to achieve very competitive results.

We have described above the underlying ideas of EUSC but there are a number of factors that should also be taken into consideration. In the evolutionary search, chromosomes are evaluated at initialisation and during the evolutionary loop. When designing EUSC, we realise that the initialisation may be a key step for the entire search, and using approximate values to begin with may not be ideal. Thus, in our experiments we investigate the influence of applying inference at both *Initialisation* and *Evolution* or merely at the *Evolution* phase. Note that whenever the population NP is restarted, we have decided to only perform real fitness evaluations to ensure the search is not misled.

4. Experimental framework

In this section, we present the experimental framework in which our proposal, the original EUS and the EUS with Windowing evaluation will be compared. This section begins with the description of the used imbalanced datasets (Section 4.1) and is followed by the parameter configuration (Section 4.2). Finally, we briefly introduce the non-parametric statistical tests (Section 4.3) that will be used to analyse the results.

4.1. Datasets

In our experiments we consider numerous two-class imbalanced datasets with different imbalanced ratios, from low to high. The datasets are obtained from the KEEL dataset repository [67] which has been used as a resource for many experiments in previous studies [27, 68]. Table

1 summarises the properties of the datasets and is sorted in the order of increasing imbalanced ratio. The range of imbalanced ratio goes from 1.5 to 9 for low imbalanced datasets and over 9 for highly imbalanced datasets. Each row represents a dataset showing its name (**Dataset**), the number of attributes (**Att**), the number of samples (**Samp**), the percentage of examples for each class (**%Class(min,maj)**) and the imbalanced ratio (**IR**). In total, there are 44 datasets in the experimental setup; each one is partitioned using 5-fold stratified cross-validation which delivers an adequate number of positive class samples in the test partitions. In a 5-fold cross-validation scheme, the original input data is randomly partitioned into 5 subsets. To compute the performance of a method, a single subset is retained to test the model, while the remaining 4 subsets are used as training data. This process is repeated until each one of the 5 subsets serves once as the test data. Thus, the overall performance of each dataset is given by an average of the 5 results from the 5 folds.

Table 1: Summary of datasets from low to highly imbalanced ratios.

Dataset	**Att**	**Samp**	**%Class (min,maj)**	**IR**
glass1	9	214	(0.36, 0.64)	1.82
ecoli-0_vs_1	7	220	(0.35, 0.65)	1.86
wisconsinImb	9	683	(0.35, 0.65)	1.86
pimaImb	8	768	(0.35, 0.65)	1.87
iris0	4	150	(0.33, 0.67)	2.00
glass0	9	214	(0.33, 0.67)	2.06
yeast1	8	1484	(0.29, 0.71)	2.46
habermanImb	3	306	(0.26, 0.74)	2.78
vehicle2	18	846	(0.26, 0.74)	2.88
vehicle1	18	846	(0.26, 0.74)	2.90
vehicle3	18	846	(0.25, 0.75)	2.99
glass-0-1-2-3_vs_4-5-6	9	214	(0.24, 0.76)	3.20
vehicle0	18	846	(0.24, 0.76)	3.25
ecoli1	7	336	(0.23, 0.77)	3.36
new-thyroid1	5	215	(0.16, 0.84)	5.14
new-thyroid2	5	215	(0.16, 0.84)	5.14
ecoli2	7	336	(0.15, 0.85)	5.46
segment0	19	2308	(0.14, 0.86)	6.02
glass6	9	214	(0.14, 0.86)	6.38
yeast3	8	1484	(0.11, 0.89)	8.10
ecoli3	7	336	(0.10, 0.90)	8.60
page-blocks0	10	5472	(0.10, 0.90)	8.79
yeast-2_vs_4	8	514	(0.10, 0.90)	9.08
yeast-0-5-6-7-9_vs_4	8	528	(0.10, 0.90)	9.35
vowel0	13	988	(0.09, 0.91)	9.98
glass-0-1-6_vs_2	9	192	(0.09, 0.91)	10.29
glass2	9	214	(0.08, 0.92)	11.59
shuttle-c0-vs-c4	9	1829	(0.07, 0.93)	13.87
yeast-1_vs_7	7	459	(0.07, 0.93)	14.30
glass4	9	214	(0.06, 0.94)	15.46
ecoli4	7	336	(0.06, 0.94)	15.80
page-blocks-1-3_vs_4	10	472	(0.06, 0.94)	15.86
abalone9-18	8	731	(0.06, 0.94)	16.40
glass-0-1-6_vs_5	9	184	(0.05, 0.95)	19.44
shuttle-c2-vs-c4	9	129	(0.05, 0.95)	20.50
yeast-1-4-5-8_vs_7	8	693	(0.04, 0.96)	22.10
glass5	9	214	(0.04, 0.96)	22.78
yeast-2_vs_8	8	482	(0.04, 0.96)	23.10
yeast4	8	1484	(0.03, 0.97)	28.10
yeast-1-2-8-9_vs_7	8	947	(0.03, 0.97)	30.57
yeast5	8	1484	(0.03, 0.97)	32.73
ecoli-0-1-3-7_vs_2-6	7	281	(0.02, 0.98)	39.14
yeast6	8	1484	(0.02, 0.98)	41.40
abalone19	8	4174	(0.01, 0.99)	129.44

4.2. Parameter configuration

EUS has been widely used in the literature and we employ the parameter values used in [22], including the number of evaluations, population size, penalisation factor and others, as it has been done in most studies for a fair comparison. EUSC includes two additional parameters, namely k_1 and k_2. k_1 is the number of clusters when the majority class examples are divided in stage 1 of the algorithm, while k_2 is the number of groups into which intermediate chromosomes are categorised in stage 2. We explored a great number of combination pairs $\{k_1, k_2\}$ where each k_1 in {2, 4, 6, 8, 10, 12, 14, 20} is combined with each k_2 in {2, 4, 6, 8, 10, 12}, resulting in a total of 48 pairs of $\{k_1, k_2\}$. We analysed the performance of EUC on all datasets with these pairs and only found slight differences among them. Thus, we decided not to report all results in this study for the sake of simplicity as they do not affect the conclusion of comparing the EUSC schemes to the EUS algorithm. In particular, we have chosen the pair $\{6, 6\}$ to present the outputs. Furthermore, as discussed in Section 2.1, EUS uses a heuristic search to find the subset of examples which can be different depending on the starting point of search. Hence, to make a fair comparison, we set up 10 fixed seeds to let each algorithm begin from the same point in 10 different executions. The results presented are the average of these ten execusions.

To analyse the effectiveness of our EUSC proposal, we will use four different strategies (summarised in Table 2), in which each is varied by two factors Representative (*Rep*) and Inference (*Infer*). As discussed in Section 3.2.2, these two elements contribute their significant influence on the behaviour of the search. *Rep* is a boolean value to determine whether the representative is a random or the centroid chromosome. *Infer* is a boolean value to determine whether fitness inference is used at both *Initialisation* and *Evolution* or merely at the *Evolution* phase. The results obtained from these four schemes will be contrasted against two benchmarks: (1) the original EUS, which is assumed to achieve the highest g-mean, but the most time-consuming approach; (2) EUS using windowing to evaluate its fitness function. Table 2 summarises all the parameters used in the experiments.

4.3. Non-parametric tests for statistical analysis

As there is not an established procedure to assert whether a classifier is better than another, statistical approaches have been adopted to determine whether the difference in performance obtained from the experiments is real or random. While a parametric test requires several assumptions to

Table 2: Parameters used for different configurations of EUSC.

Rep	Infer	Scheme	Shared parameters
Random	Initialisation & Evolution	R-IE	Population Size = 50, Max Evaluations = 10000
	Evolution	R-E	Probability of inclusion HUX = 0.25
Centroid	Initialisation & Evolution	C-IE	Measure = g-mean, $k_1 = 6$
	Evolution	C-E	$k_2 = 6$, Number of Runs = 10

be satisfied, such as a normal distribution of inputs and homogeneity of variance, a non-parametric test is free from those requirements [69, 70]. In this paper, we will employ non-parametric tests, specifically the Friedman Aligned-Ranks test [71] plus a Holm post-hoc test [72] to perform statistical analysis on the performance of the algorithms. Initially, the Friedman Aligned-Ranks test conducts multiple comparisons to detect statistical differences in the performance of multiple algorithms. This test will establish a ranking order of all compared algorithms. Then, the Holm post-hoc test is used to discover whether the highest ranking algorithm (the control algorithm) presents statistical differences with respect to the remaining methods. In our experiment, a level of significance of $\alpha = 0.05$ is adopted. Further information discussing these tests can be referred at `http://sci2s.ugr.es/sicidm/`.

5. Analysis of results

In this section, we present an examination of the results. It begins with a detailed analysis of the behaviour of the EUSC scheme through an evolutionary search cycle (Section 5.1), and is followed by the report of the runtime as well as the reduction in the number of fitness function calls (Section 5.2). Finally, we statistically compare the performance of the different algorithms (Section 5.3).

5.1. Detailed analysis of the behaviour of EUSC

EUSC approximates the fitness values for some of the chromosomes during the evolutionary, which may change the behaviour of the original EUS. The aim of this section is to investigate whether our fitness approximation approach produces values close to the real fitness ones and determine which of the four EUSC configurations perform better on the 44 used datasets. To do so, we carry out the following two experiments.

Experiment 1. The first experiment involves an analysis of the fitness difference between approximated and true fitness values through an evolutionary search cycle. Note that the fitness difference is reported as an average figure in relation to the number of chromosomes whose fitness is approximated in each generation (which may differ in every generation). This is due to the nature of CHC which does not always produce an offspring of NP elements in every generation. The number of chromosomes to be generated depends on the population diversity. Hence, the surrogate model for fitness approximation is only employed when the quantity of chromosomes is greater than k_2. An example of an evolutionary search process run by scheme R-IE on fold 1 of dataset ecoli2 is plotted in Figure 2. Two y-axes are sharing the same x-axis. The left y-axis presents the fitness difference, while the right one expresses the number of chromosomes that are evaluated. Note that Figure 2 is only an example to characterise the behaviour of a specific EUSC on a dataset fold. However, a similar behaviour has been observed in other datasets and the different variants.

Analysing this figure in detail, we can see how the evolutionary search cycle begins with random initialisation of NP chromosomes. Due to random allocation, these chromosomes have diverse fitness values, resulting in a large value of fitness difference. Through generations, the value progressively decreases and remains low at less than 0.05 until the end of the search. Particularly, in generations 90-150, 190-260, 290-370, and 420-440 where the fitness inference is used, while the chromosome quantity follows an upward trend, the variation of fitness difference value remains mostly unchanged, fluctuates at somewhere between 0 and 0.05. This means that while the CHC algorithm introduces more diversity to the population, the EUSC approach is yet effective to divide the chromosomes into groups at which members of each group are approximate in fitness. There appear discontinuities in the fitness difference curve, indicating that fitness approximation is not applied in these generations as the number of generated offspring is lower than k_2. Furthermore, when the upward trend of the chromosome quantity reaches the top value (NP), it suddenly drops to the bottom due to the restart mechanism of the CHC algorithm. The algorithm uses a mutation process to generate a new generation from the current best chromosome. A new similar pattern of the search restarts from this point as the population has been refreshed.

Experiment 2. With the previous experiment, we can say that at each generation of a search, the EUSC does not produce a large difference between approximated and true fitness values. If each EUSC scheme does not behave remarkably differently, it may end up with the same number

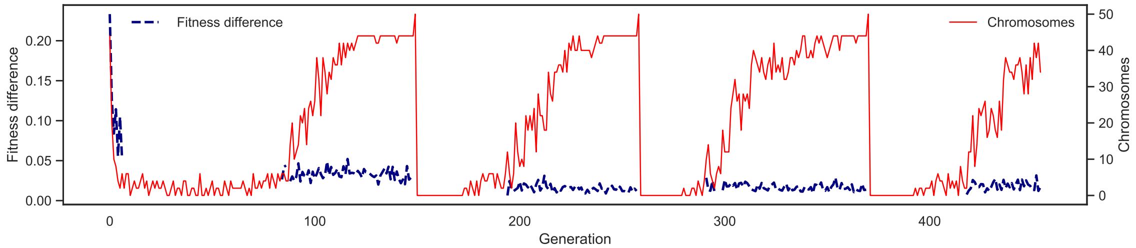

Figure 2: Behaviour of the EUSC through generations run by method R-IE on fold 1 of dataset ecoli2.

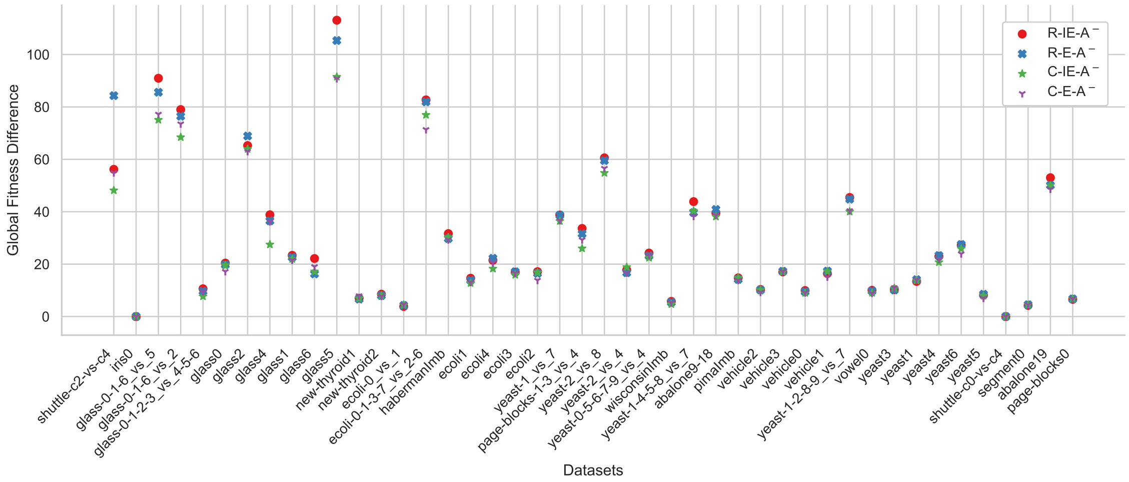

Figure 3: Total fitness difference aggregated from evolutionary search of each EUSC scheme over 5 folds of each dataset.

of generations per search, resulting in a close fitness difference in total (note that the stopping criteria is based on number of evaluations, and not number of generations). In this experiment, we aggregate all the generated fitness differences throughout the entire search and from the 5 folds of each dataset, displayed in Figure 3. The aim of this experiment is to understand how different the search of four schemes performs regarding the global fitness difference. Higher fitness difference in an EUSC configuration infers that either more generations have been conducted in a search or fitness difference is large in some particular EUSC schemes and some datasets. The results have been sorted by the number of samples of datasets for the sake of studying the influence of their size. 435

440 Centroid-based schemes (representative is the centroid chromosome) are likely to produce lower global fitness difference with respect to the random-based counterparts (representative is a random

chromosome), denoted by a lower/nearly equal value of fitness difference in all datasets. The higher aggregated value in these random-based approaches can be attributed to the representative selection. The random-based schemes might obtain a very good solution and assign its fitness to other chromosomes, at which the least difference is generated. However, there is a probability that the selection of a representative falls into the worst situation or nearby at which the inference task produces substantial fitness difference and thus adds a heavy burden to the global value. Note that if chromosomes in a cluster share approximate values, the difference may not be strongly affected by the selection mechanism.

In Figure 3 the four schemes tend to have a very similar behaviour in most of the datasets, producing a similar accumulated fitness difference value. On datasets that the accumulated fitness difference is in between 40 and 50, the four EUSC schemes gather in a finite range, meaning the search tends to be slightly more different. On datasets with the fitness difference values above 50, the four EUSC schemes show clear differences. These datasets are usually the ones having very high IR. It is much more diffused on datasets with both high IR and low samples, such as glass5 and glass-0-1-6_vs_5. As a result, with datasets having a low number of samples, it is reasonable not to use EUSC as the EUS does not suffer from high computational expense.

5.2. Runtime and real evaluations reduction

In this section, we will compare the runtime needed by all the compared algorithms in every dataset. Figure 4 plots the comparison. For the sake of clarity and observing the influence of data size on the runtime, we group the first 22 datasets in the top sub-plot (smaller datasets), 15 next datasets in the middle sub-plot (medium-size datasets) and remaining 7 datasets in the bottom one (the larger datasets used in the experiments). Looking at that figure, we can observe that:

- Overall, four EUSC schemes demanded an insignificant amount of time to perform undersampling in comparison to the time required by the original EUS. This is also the case for EUS with windowing in about half of the datasets. However, as mentioned before, the EUS with windowing is strongly influenced by the IR. Thus, its runtime on datasets with high IR was low and much lower when the size of those datasets is small. However, in other small datasets (low IR) this approach consumed a mostly comparable amount of time to EUS, and the runtime dramatically reduced in larger ones.

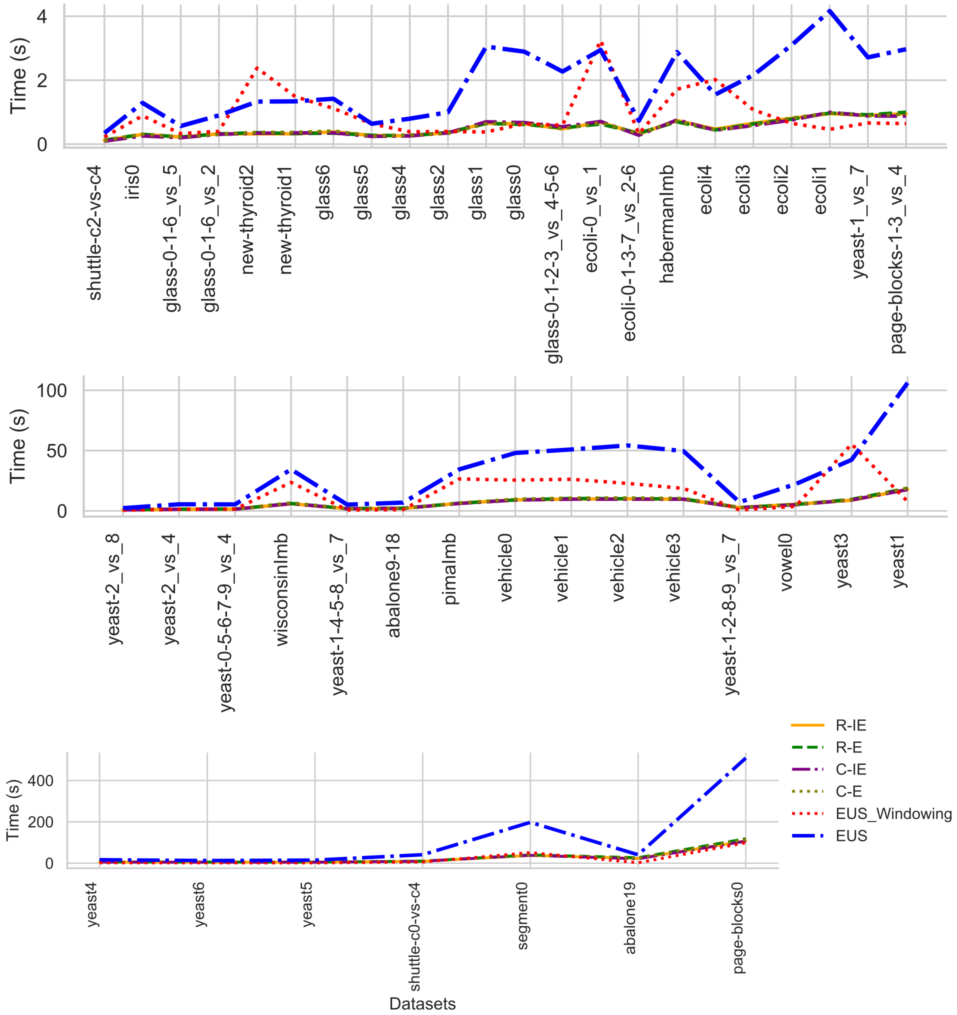

Figure 4: Comparison of the runtime (in seconds) of the involved algorithms in every single dataset. Runtime of the first 22 datasets (Top), next 15 datasets (Middle), last 7 datasets (Bottom).

- Although the four EUSC schemes have a difference in the amount of time consumed, the variation is not significant in comparison with EUS. Over 44 datasets, the minimum and maximum percentage of the runtime saved are 16.76% and 83.24%, respectively. Additionally, on average in all datasets, EUS takes 30.63s, EUS_windowing takes 9.03s, and the four EUSC schemes consume from 6.66s to 7.25s.

In addition to runtime, we also report the number of evaluations that our method saved. As

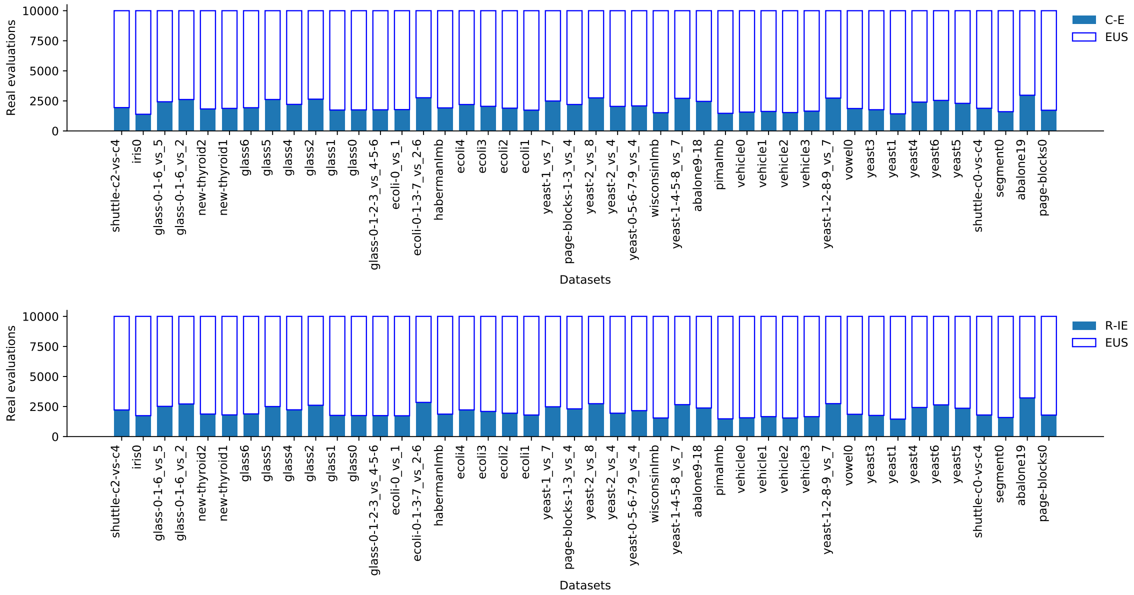

Figure 5: Histogram of real evaluations from the two EUSC schemes constrasted to the EUS algorithm over 44 imbalanced datasets.

there is not a significant difference observed in the number of real fitness function calls among four EUSC schemes, we plot the histograms with the total number of evaluations of two representative schemes (C-E and R-IE) compared to the 10000 evaluations performed by the EUS for each dataset (Figure 5). As seen in the graph, the two schemes save a significant number of evaluations, up to 80% the evaluations demanded by the original EUS in virtually most datasets. This infers a huge reduction of extensive computation in calculating the fitness for each chromosome.

5.3. *Classification performance comparison*

In the previous section, we have shown that our proposal is more advantageous than the original EUS or EUS_Windowing in terms of runtime and the number of real evaluations used. However, these gains would not be of any value if the final goal of achieving high classification performance considerably decreased. This section thus aims to report the average g-mean of all the algorithms in test data. Table 3 shows the results in detail. The best result for each dataset is highlighted in boldface. The last row shows the number of times in which an algorithm has obtained the highest performance.

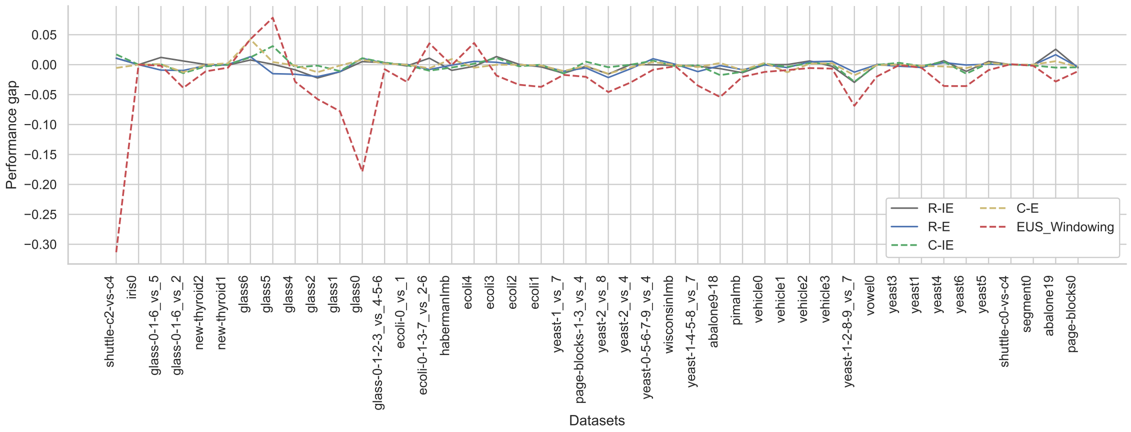

Figure 6: The difference in terms of g-mean of the EUSC schemes and EUS_Windowing constrated against the EUS.

Additionally, we also highlight how much difference in terms of g-mean of the four EUSC configurations and the EUS_Windowing against the original EUS. Every g-mean value of each EUSC scheme and EUS_Windowing subtracts the g-mean of the EUS to produce the curves, displayed in Figure 6. The plots lying above 0 signify better performance, while those under 0 mean worse. The distance measured from the baseline ($y = 0$) to a plot indicates how much an algorithm performs better/worse than the EUS. Looking at this table and figure, we can make the following observations:

- Despite using up to 80% more evaluations, EUS does not always achieve the highest g-mean. In contrast, using fitness approximation approaches does not only save a lot of computation but also might sometimes get even better performance. From table 3, we can highlight the configuration R-IE, which appears very competitive in comparison to the EUS algorithm, obtaining the same number of wins out of the 44 datasets. Other algorithms find the best solution in 5 or 6 out of 44 datasets.
- In Figure 6, four EUSC schemes oscillate around the baseline in a confined range of [-0.03, +0.04], while EUS_Windowing fluctuates with a larger distance. This infers stable performance of all configurations which is more consistent with the performance attained by the EUS technique.

Although the differences of g-mean among all involved algorithms are likely inconsiderable from

Table 3: Average g-mean of all compared algorithms over 44 datasets.

Dataset	R-IE	R-E	C-IE	C-E	EUS_Windowing	EUS
shuttle-c2-vs-c4	0.9690	0.9690	**0.9753**	0.9527	0.6449	0.9582
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
glass-0-1-6_vs_5	**0.9289**	0.9077	0.9175	0.9176	0.9151	0.9168
glass-0-1-6_vs_2	**0.6612**	0.6454	0.6409	0.6426	0.6164	0.6551
new-thyroid2	**0.9888**	0.9871	0.9856	0.9885	0.9773	0.9885
new-thyroid1	0.9851	0.9862	0.9862	**0.9882**	0.9809	0.9859
glass6	0.8724	0.8779	0.8768	0.9068	**0.9071**	0.8646
glass5	0.8298	0.8141	0.8602	0.8336	**0.9076**	0.8292
glass4	0.8712	0.8638	0.8752	0.8785	0.8513	**0.8798**
glass2	0.6879	0.6901	0.7085	0.6975	0.6525	**0.7101**
glass1	0.7668	0.7669	0.7680	0.7772	0.7010	**0.7787**
glass0	0.8015	0.8070	**0.8072**	0.8046	0.6176	0.7964
glass-0-1-2-3_vs_4-5-6	0.9493	0.9483	**0.9496**	0.9478	0.9385	0.9461
ecoli-0_vs_1	0.9580	**0.9605**	0.9591	0.9601	0.9312	0.9601
ecoli-0-1-3-7_vs_2-6	0.6800	0.6611	0.6591	0.6631	**0.7048**	0.6692
habermanImb	0.5547	0.5634	0.5594	**0.5736**	0.5635	0.5642
ecoli4	0.8972	0.9055	0.9016	0.8949	**0.9362**	0.9000
ecoli3	**0.8470**	0.8375	0.8447	0.8333	0.8153	0.8335
ecoli2	**0.9005**	0.8988	0.8971	0.8996	0.8663	0.8998
ecoli1	0.8639	0.8660	0.8676	0.8662	0.8306	**0.8677**
yeast-1_vs_7	0.7106	0.7140	0.7102	0.7144	0.7079	**0.7250**
page-blocks-1-3_vs_4	0.9575	0.9547	**0.9652**	0.9581	0.9399	0.9602
yeast-2_vs_8	0.7797	0.7739	0.7911	0.7802	0.7496	**0.7954**
yeast-2_vs_4	**0.9074**	0.9003	0.9070	0.9027	0.8774	0.9071
yeast-0-5-6-7-9_vs_4	0.7747	**0.7848**	0.7815	0.7792	0.7663	0.7749
wisconsinImb	0.9666	**0.9684**	0.9652	0.9674	0.9652	0.9678
yeast-1-4-5-8_vs_7	0.6412	0.6321	0.6427	0.6396	0.6088	**0.6437**
abalone9-18	0.7246	0.7297	0.7138	**0.7341**	0.6772	0.7313
pimaImb	0.6817	0.6867	0.6831	0.6859	0.6749	**0.6951**
vehicle0	0.9148	0.9143	0.9166	**0.9179**	0.9027	0.9148
vehicle1	**0.6716**	0.6667	0.6678	0.6588	0.6624	0.6715
vehicle2	**0.9294**	0.9280	0.9257	0.9247	0.9175	0.9232
vehicle3	0.7180	**0.7265**	0.7222	0.7232	0.7142	0.7208
yeast-1-2-8-9_vs_7	0.6469	0.6640	0.6479	0.6588	0.6078	**0.6765**
vowel0	**0.9921**	0.9918	0.9914	0.9910	0.9719	0.9918
yeast3	0.8751	0.8722	**0.8780**	0.8747	0.8740	0.8749
yeast1	0.6501	0.6509	0.6527	0.6532	0.6501	**0.6552**
yeast4	**0.8222**	0.8191	0.8206	0.8126	0.7799	0.8156
yeast6	0.8326	0.8432	0.8287	0.8381	0.8080	**0.8438**
yeast5	**0.9639**	0.9593	0.9611	0.9614	0.9494	0.9585
shuttle-c0-vs-c4	0.9960	0.9960	0.9960	0.9960	**0.9968**	0.9960
segment0	0.9884	0.9883	0.9876	**0.9890**	0.9870	0.9889
abalone19	**0.6600**	0.6509	0.6294	0.6401	0.6061	0.6343
page-blocks0	0.9096	0.9108	0.9103	0.9111	0.9038	**0.9148**
Wins	**13**	**5**	**6**	**6**	**6**	**13**

what has been analysed so far, there is no evidence to confirm whether these differences are significant. Hence, we will employ the Friedman Aligned-Ranks test to discover if there exist statistical differences in the involved algorithms. After having the ranking table provided by the Friedman test, we then use Holm post-hoc test to find out whether the highest-ranking algorithm statistically outperforms the rest. Table 4 presents the results of this test. In this table, algorithms are sorted by their ranks, from the best to the worst and each algorithm is also associated with a p_{Holm} value at the same row.

Table 4: Average rankings of the algorithms over 44 datasets (Friedman Aligned-Ranks test and Holm post-hoc test).

Algorithm	Ranking	p_{Holm}
EUS	2.7386	-
C-E	3.0568	0.4881
C-IE	3.2841	0.4881
R-IE	3.2955	0.4881
R-E	3.4091	0.3711
EUS_Windowing	5.2159	0

- As expected, the original EUS gets the lowest ranking value and is established as the control algorithm. Our EUSC schemes are placed in the middle of the table, while EUS_Windowing with the largest ranking value lies in the last position.
- With the level of significance $\alpha = 0.05$, Holm's test has no reported significant differences between EUS and four EUSC schemes. However, the test reveals the EUS algorithm statistically outperforms the EUS_Windowing as $p\text{-}value = 0$.
- Although R-IE has the same number of wins with the EUS over 44 datasets, it does not show this advantage in the Ranking and p_{Holm} columns. This is because in the datasets that R-IE does not outperform the best, it is usually ranked after C-E and C-IE. Hence, the benefits gained from winning is deducted from the loss at the lower rank.

Finally, to make sure that our proposed approach has not benefit from having the EUS_Windowing algorithm in the $1 * N$ comparison, we apply the Wilcoxon test between the original EUS and the different EUSC schemes. Table 5 shows rankings R^+ and R^- together with the p-values obtained from the Wilcoxon test. As we can see, R^+ is slightly greater than R^- in the first two rows and moderately larger in the last two rows (meaning that EUS obtains slightly better results). However, the Wilcoxon test does not report significant differences between the EUS with a level of significance $\alpha = 0.05$.

Table 5: Results of the Wilcoxon test when the EUS algorithm is contrasted against the seven most effective EUSC schemes.

EUS vs	R^+	R^-	*p-value*
C-IE	582.5	407.5	≥ 0.2
R-IE	569.5	420.5	≥ 0.2
C-E	629.5	360.5	0.1185
R-E	656.5	333.5	0.0598

6. Conclusions

This paper has investigated the challenges of using fitness approximation in evolutionary undersampling for imbalance classification at which optimising the selection of instances is a challenging binary optimisation problem. The proposed surrogate-model is based on a two-stage clustering approach that transforms binary chromosomes into real-coding ones to allow us to compute distances between chromosomes, so that, fitness values can be approximated fairly.

An extensive experimental framework was carried out to demonstrate the effectiveness of our proposal compared to the original evolutionary undersampling method considering multiple perspectives. The experiments show that we are capable of drastically reducing the runtime required to perform undersampling without significantly losing classification performance. In comparison with alternative approaches to approximate fitness values in evolutionary undersampling (namely windowing), our method has demonstrated to consistently reduce the runtime and maintain high quality solutions. As such, the proposed EUSC algorithm might not be able to work directly with very big datasets due to the memory limitation of an individual computer. However, as discussed in Sections 2.2 and 2.3, our proposal could be integrated with existing parallelisation approaches [66] to handle extremely large datasets [25], which we plan to tackle as future work.

Acknowledgement

The work of H. Lam Le was funded by a Ph.D. scholarship from the School of Computer Science of the University of Nottingham.

References

[1] H.-L. Dai, Imbalanced protein data classification using ensemble FTM-SVM, IEEE Transactions on Nanobioscience 14 (4) (2015) 350–359.

[2] B. Zhu, B. Baesens, S. K. vanden Broucke, An empirical comparison of techniques for the class imbalance problem in churn prediction, Information Sciences 408 (2017) 84–99.

[3] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, B. Yang, Machine learning based mobile malware detection using highly imbalanced network traffic, Information Sciences 433 (2018) 346–364.

[4] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge & Data Engineering (9) (2008) 1263–1284.

[5] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Information Sciences 250 (2013) 113–141.

[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Systems with Applications 73 (2017) 220–239.

[7] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, F. Herrera, Learning from imbalanced data sets, Springer, 2018.

[8] F. Thabtah, S. Hammoud, F. Kamalov, A. Gonsalves, Data imbalance in classification: Experimental evaluation, Information Sciences 513 (2020) 429 – 441.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[10] G. E. Batista, R. C. Prati, M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 20–29.

[11] I. Triguero, D. García-Gil, J. Maillo, J. Luengo, S. García, F. Herrera, Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data, WIREs Data Mining and Knowledge Discovery 9 (2) (2019) 1289.

[12] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 204–213.

[13] S.-H. Oh, Error back-propagation algorithm for classification of imbalanced data, Neurocomputing 74 (6) (2011) 1058–1061.

[14] P. Domingos, Metacost: A general method for making classifiers cost-sensitive, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 155–164.

[15] B. Krawczyk, M. Woźniak, G. Schaefer, Cost-sensitive decision tree ensembles for effective imbalanced classification, Applied Soft Computing 14 (2014) 554–562.

[16] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (4) (2011) 463–484.

[17] E. Fernandes, A. C. P. de Leon Ferreira, D. Carvalho, X. Yao, Ensemble of classifiers based on multiobjective genetic sampling for imbalanced data, IEEE Transactions on Knowledge and Data Engineering 14 (2019) 1041–4347.

[18] R. Sundar, M. Punniyamoorthy, Performance enhanced boosted SVM for imbalanced datasets, Applied Soft Computing (2019) 105601.

[19] N. V. Chawla, A. Lazarevic, L. O. Hall, K. W. Bowyer, SMOTEBoost: Improving prediction of the minority class in boosting, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2003, pp. 107–119.

[20] K. Li, X. Kong, Z. Lu, L. Wenyin, J. Yin, Boosting weighted ELM for imbalanced learning, Neurocomputing 128 (2014) 15–21.

[21] S.-J. Yen, Y.-S. Lee, Cluster-based under-sampling approaches for imbalanced data distributions, Expert Systems with Applications 36 (3) (2009) 5718–5727.

[22] S. García, F. Herrera, Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy, Evolutionary Computation 17 (3) (2009) 275–306.

[23] A. Fernández, S. Garcia, F. Herrera, N. V. Chawla, SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, Journal of Artificial Intelligence Research 61 (2018) 863–905.

[24] E. Ramentol, Y. Caballero, R. Bello, F. Herrera, SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory, Knowledge and Information Systems 33 (2) (2012) 245–265.

[25] I. Triguero, M. Galar, S. Vluymans, C. Cornelis, H. Bustince, F. Herrera, Y. Saeys, Evolutionary undersampling for imbalanced big data classification, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 715–722.

[26] S. Garcia, J. Derrac, J. R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Transactions on Pattern Analysis & Machine Intelligence (3) (2011) 417–435.

[27] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling, Pattern Recognition 46 (12) (2013) 3460–3471.

[28] B. Krawczyk, M. Galar, Ł. Jeleń, F. Herrera, Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy, Applied Soft Computing 38 (2016) 714–726.

[29] B. Sun, H. Chen, J. Wang, H. Xie, Evolutionary under-sampling based bagging ensemble method for imbalanced data classification, Frontiers of Computer Science 12 (2) (2018) 331–350.

[30] H. Song, I. Triguero, E. Ozcan, A review on the self and dual interactions between machine learning and optimisation., Progress in Artificial Intelligence 8 (2019) 143–165.

[31] H. Tao, P. Wang, Y. Chen, V. Stojanovic, H. Yang, An unsupervised fault diagnosis method for rolling bearing using stft and generative neural networks, Journal of the Franklin Institute 357 (2020) 7286–7307.

[32] V. Stojanovic, S. He, B. Zhang, State and parameter joint estimation of linear stochastic systems in presence of faults and non-gaussian noises, International Journal of Robust and Nonlinear Control 30 (2020) 6683–6700.

[33] T. M. Cover, P. Hart, et al., Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.

[34] I. Triguero, M. Galar, H. Bustince, F. Herrera, A first attempt on global evolutionary undersampling for imbalanced big data, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2054–2061.

[35] Y. Jin, M. Olhofer, B. Sendhoff, On evolutionary optimization with approximate fitness functions, in: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2000, pp. 786–793.

[36] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing 9 (1) (2005) 3–12.

[37] Y. Jin, Surrogate-assisted evolutionary computation: Recent advances and future challenges, Swarm and Evolutionary Computation 1 (2) (2011) 61–70.

[38] A. Rosales-Pérez, J. A. Gonzalez, C. A. C. Coello, H. J. Escalante, C. A. Reyes-Garcia, Surrogate-assisted multi-objective model selection for support vector machines, Neurocomputing 150 (2015) 163 – 172.

[39] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, M. Zhang, Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor, IEEE Transactions on Evolutionary Computation (2019) 1–1.

[40] A. E. Brownlee, J. A. Wright, Constrained, mixed-integer and multi-objective optimisation of building designs by nsga-ii with fitness approximation, Applied Soft Computing 33 (2015) 114–126.

[41] I. Bertini, M. De Felice, A. Pannicelli, S. Pizzuti, Soft computing based optimization of combined cycled power plant start-up operation with fitness approximation methods, Applied Soft Computing 11 (6) (2011) 4110–4116.

[42] M. Salami, T. Hendtlass, A fast evaluation strategy for evolutionary algorithms, Applied Soft Computing 2 (3) (2003) 156–173.

[43] T. Chugh, C. Sun, H. Wang, Y. Jin, Surrogate-assisted evolutionary optimization of large problems, in: High-Performance Simulation-Based Optimization, Springer, 2020, pp. 165–187.

[44] A. Moraglio, A. Kattan, Geometric generalisation of surrogate model based optimisation to combinatorial spaces, in: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2011, pp. 142–154.

[45] T. Bartz-Beielstein, M. Zaefferer, Model-based methods for continuous and discrete global optimization, Applied Soft Computing 55 (2017) 154–167.

[46] J. Bacardit, D. E. Goldberg, M. V. Butz, X. Llorà, J. M. Garrell, Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 1021–1031.

[47] J. R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters 26 (7) (2005) 953–963.

[48] I. Triguero, J. Derrac, S. Garcia, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (1) (2011) 86–100.

[49] L. J. Eshelman, The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, Foundations of Genetic Algorithms 1 (1991) 265–283.

[50] R. Barandela, J. S. Sánchez, V. Garca, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognition 36 (3) (2003) 849–851.

[51] A. P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.

[52] D. Buche, N. N. Schraudolph, P. Koumoutsakos, Accelerating evolutionary algorithms with gaussian process fitness function models, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 35 (2) (2005) 183–194.

[53] L. Bianchi, M. Dorigo, Ant colony optimization and local search for the probabilistic traveling salesman problem: a case study in stochastic combinatorial optimization, PhD dissertation, Université libre de Bruxelles (2006).

[54] P. Ross, D. Corne, H.-L. Fang, Improving evolutionary timetabling with delta evaluation and directed mutation, in: International Conference on Parallel Problem Solving from Nature, Springer, 1994, pp. 556–565.

[55] L. Shi, K. Rasheed, A survey of fitness approximation methods applied in evolutionary algorithms, in: Computational intelligence in expensive optimization problems, Springer, 2010, pp. 3–28.

[56] R. E. Smith, B. A. Dike, S. Stegmann, Fitness inheritance in genetic algorithms, in: Proceedings of the 1995 ACM Symposium on Applied computing, ACM, 1995, pp. 345–350.

[57] K. Sastry, D. E. Goldberg, M. Pelikan, Don't evaluate, inherit, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2001, pp. 551–558.

[58] L. T. Bui, H. A. Abbass, D. Essam, Fitness inheritance for noisy evolutionary multi-objective optimization, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, ACM, 2005, pp. 779–785.

[59] M. Pelikan, K. Sastry, Fitness inheritance in the bayesian optimization algorithm, in: Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 48–59.

[60] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2) (2015) 165–193.

[61] H.-S. Kim, S.-B. Cho, An efficient genetic algorithm with less fitness evaluation by clustering, in: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, Vol. 2, IEEE, 2001, pp. 887–894.

[62] Y. Jin, B. Sendhoff, Reducing fitness evaluations using clustering techniques and neural network ensembles, in: Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 688–699.

[63] A. C. Martínez-Estudillo, C. Hervás-Martínez, F. J. Martínez-Estudillo, N. García-Pedrajas, Hybridization of evolutionary algorithms and local search by means of a clustering method, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36 (3) (2005) 534–545.

[64] R. Li, M. T. Emmerich, J. Eggermont, E. G. Bovenkamp, T. Back, J. Dijkstra, J. H. Reiber, Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 2764–2771.

[65] J. Derrac, S. García, F. Herrera, Stratified prototype selection based on a steady-state memetic algorithm: a study of scalability, Memetic Computing 2 (3) (2010) 183–199.

[66] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, Mrpr: a mapreduce solution for prototype reduction in big data classification, Neurocomputing 150 (2015) 331–345.

[67] I. Triguero, S. González, J. M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. Fernández Hilario, J. Díaz, L. Sánchez, F. Herrera, et al., Keel 3.0: an open source software for multi-stage analysis in data mining, International Journal of Computational Intelligence Systems 10 (2017) 1238–1249.

[68] S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani, F. Herrera, Dynamic ensemble selection for multi-class imbalanced datasets, Information Sciences 445 (2018) 22–37.

[69] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[70] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[71] J. Hodges, E. L. Lehmann, et al., Rank methods for combination of independent experiments in analysis of variance, The Annals of Mathematical Statistics 33 (2) (1962) 482–497.

[72] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics (1979) 65–70.