SPMS-ALS: A Single-Point Memetic Structure with Accelerated Local Search for Instance Reduction

Hoang Lam Le[a,*], Ferrante Neri[a], Isaac Triguero[a]

[a] *The Optimisation and Learning (COL) Lab at the School of Computer Science, University of Nottingham, Nottingham NG8 1BB, United Kingdom*

Abstract

Real-world optimisation problems pose domain specific challenges that often require an ad-hoc algorithmic design to be efficiently addressed. The present paper investigates the optimisation of a key stage in data mining, known as instance reduction, which aims to shrink the input data prior to applying a learning algorithm. Performing a smart selection or creation of a reduced number of samples that represent the original data may become a complex large-scale optimisation problem, characterised by a computationally expensive objective function, which has been often tackled by sophisticated population-based metaheuristics that suffer from a high runtime.

Instead, by following the Ockham's Razor in Memetic Computing, we propose a Memetic Computing approach that we refer to as fast Single-Point Memetic Structure with Accelerated Local Search (SPMS-ALS). Using the k-nearest neighbours algorithm as base classifier, we first employ a simple local search for large-scale problems that exploits the search logic of Pattern Search, perturbing an n-dimensional vector along the directions identified by its design variables one by one. This point-by-point perturbation mechanism allows us to design a strategy to re-use most of the calculations previously made to compute the objective function of a candidate solution. The proposed Accelerated Local Search is integrated within a single-point memetic framework and coupled with a resampling mechanism and a crossover. A thorough experimental analysis shows that SPMS-ALS, despite its simplicity, displays an excellent performance which is as good as that of the state-of-the-art while reducing up to approximately 85% of the runtime with respect

*Corresponding author

Email address: hoang.le@nottingham.ac.uk (Hoang Lam Le)

to any other algorithm that performs the same number of function calls.

Keywords: Memetic Algorithm, Pattern Search, Instance Reduction, Classification, Data Science, Ockham's Razor in Memetic Computing

1. Introduction

Since their earliest definition [35, 34] Memetic Algorithms (MAs) were introduced to enhance upon the performance of algorithms, such as Genetic Algorithms and Simulated Annealing. Unlike the majority of other algorithms, MAs are not fixed to a specific structure but are flexible and thus versatile optimisation frameworks, see [38]. This flexibility is one of the main features of MAs which likely inspired numerous subsequent studies that shaped, over the past three decades, the field of Memetic Computing (MC).

By following the visionary ideas reported in [21] and the classification in [12], three groups/generations of MC approaches have been identified:

- **Simple Hybrids**: this group includes hybrid algorithms generated by two or more algorithms joined together in a synergistic manner. Usually, the algorithms of this type combine a global search and at least one local search. Some examples of successful hybridisations are reported in e.g. [30, 50, 31]

- **Adaptive Hybrids**: this includes hybrid algorithms where multiple local search algorithms are coordinated by an adaptive mechanism that selects the algorithmic elements at runtime. Popular selection criteria are performance-based like in hyperheuristics [44] and meta-Lamarckian learning [26, 43], diversity-based [6] or self-adaptive [42].

- **(Future) Memetic Automation**: this kind reinterprets MAs as a combination of "agents" without a predefined structure [1, 63] and investigates mechanisms to attain fully self-generated MAs. Although this design approach is still under investigation, some interesting domain-specific frameworks [15, 29] and prototypes [7] have been proposed.

The flexibility of the subject facilitating domain-specific algorithmic design is one of the reasons of the success of MAs in real-world applications, see [3, 14, 61]. In other words, while robust algorithmic design and testing on multiple abstract mathematical functions is fundamental for the development of novel memetic structures (as well as for any optimisation algorithm)

[17, 33] real-world problems often pose specific challenges which may be addressed by ad-hoc representations and specific operators [21]. Among the plethora of MC structures the need to design simple algorithm on a limited hardware inspired **Single-Point Memetic Structures** which are the focus of the present study. For example, in [39] memetic structures using virtual populations (statistical models of populations) have been implemented directly in the control cards of robots. In [9], a simplistic single-solution MC approach composed of a global evolutionary operator and a local search has been proven to be competitive with complex metaheuristics and has been successfully implemented in the control card of an helicopter robot.

The latter approach is part a family of MAs designed according to the so-called *Ockham's Razor in Memetic Computing* principle formulated in [22]: simple algorithmic structures designed by combining memes in a bottom-up approach while addressing the knowledge of the problem (prior or available at run-time) often have a high performance despite their simplicity. This idea links to other areas of optimisation research such as the pioneering studies in [60, 11] and the work on Fitness Landscape Analysis [32, 23].

The present article addresses a real-world problem in the field of data science, known as **instance reduction** [58]. Datasets can be extremely large and usually require the use of pre-processing techniques to enable data mining and machine learning techniques to learn from a cleaner and smaller dataset that is free of noise, redundant or irrelevant samples (the so-called, *Smart Data* [53]). Instance reduction is an important pre-processing procedure that pursues to shrink the original dataset and keep it as informative as by either selecting (**instance selection**) [19] or generating (**instance generation**) [51] representative instances from a very large raw dataset. This is not a trivial task, and it is essential to properly select or artificially generate those representative instances.

Instance reduction can be conceived as an optimisation problem and be tackled by search algorithms as either a binary search problem in the case of instance selection [5], or as a continuous search problem to artificially generate representative instances. In both cases, MAs have been preeminent in comparison with other approaches in terms of performance [18, 52]. The vast majority of the existing instance reduction approaches were proposed to improve the performance of the well-known Nearest Neighbour (NN) classifier [13]. However, the resulting reduced dataset may be used by any classifier [5]. In this work, we will also focus on the NN classifier.

The main issue for current instance reduction solutions is related to the

high cost of evaluating candidate solutions. When tackling bigger datasets, their runtime may become excessive and we can find in the specialised literature parallelisation approaches for instance reduction [55], which allow them to be executed, whilst increasing the need for additional computational resources. Reducing the computational cost of the fitness evaluation is an under-explored area in instance reduction, and just a few approximation approaches exist (e.g. windowing [4] or surrogate models [41]).

Bearing in mind the elevated computational cost of the fitness function, we propose a simple and yet effective domain-specific MC approach for instance reduction. The proposed MC approach is composed of a novel domain-specific implementation of local search hybridised with a global evolutionary operator. The local search exploits the logic of the Generalised Pattern Search that performs an implicit variable decomposition technique and perturbs the elements of a candidate solution one by one [40]. In contradistinction with existing population-based approaches that create new solutions perturbing multiple variables at once, we exploit the fact that the proposed local search produces candidate solutions that are only "slightly" different w.r.t the previous fitness evaluation. Based on this fact, we devise a mechanism to drastically reduce the cost of the objective function when using the NN algorithm as base classifier. The global search operator is a simple resampling mechanism followed by crossover while an elite memory slot retains the solution with the best performance. The key idea lies in keeping a single-point approach to highly accelerate the objective function evaluation while using a global operator to avoid getting stuck in local optima.

The remainder of this article is organised in the following way. Section 2 provides the background about instance reduction, formalises it as an optimisation problem and provides an explanation why the problem is unavoidably large scale and why calculation of the objective function is computationally expensive. Section 3 describes and justifies the proposed method. Section 4 presents the experimental setup while Section 5 shows and discusses the results. Finally, Section 6 provides the conclusion of this study.

2. Problem Formulation and Challenges Associated with it

In a supervised classification problem, the data is usually split into training (**TR**) and test (**TS**) sets. Each instance belongs to a class w, which is known for **TR** and unknown for **TS**. Both datasets can be viewed as a

matrix in which instances $\mathbf{I_i}$ are displayed on the rows whilst features $\mathbf{f_i}$ are shown on the columns:

$$\mathbf{TR} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & ... & \mathbf{f_m} \\ \mathbf{I_1} & a_{11} & a_{12} & ... & a_{1m} \\ \mathbf{I_2} & a_{21} & a_{22} & ... & a_{2m} \\ ... & ... & ... & ... & ... \\ \mathbf{I_l} & a_{l1} & a_{l2} & ... & a_{lm} \end{bmatrix} \tag{1}$$

The main purpose of an instance reduction technique is to clean and compress **TR** into a reduced set **RS**, by either selecting or generating new representative instances, so that, it preserves and provides valuable information for a machine learning algorithm to learn useful insights about a classification problem. Thus, the resulting **RS** should satisfy several conditions such as well-representing the distributions of the classes, significantly reducing in size to minimise the required storage, which would be beneficial to the posterior classification phase.

$$\mathbf{RS} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & ... & \mathbf{f_m} \\ \mathbf{I_1} & b_{11} & b_{12} & ... & b_{1m} \\ \mathbf{I_2} & b_{12} & b_{22} & ... & b_{2m} \\ ... & ... & ... & ... & ... \\ \mathbf{I_p} & b_{p1} & b_{p2} & ... & b_{pm} \end{bmatrix} \tag{2}$$

with $p \ll l$. In this study we choose to treat p as a parameter that signifies the compression of the data with respect to the size of the entire set of training set (number of rows of **TR**). More specifically, we use the reduction rate $\frac{l}{p}$ as a parameter of our problem. In both matrices **TR** and **RS** each row is associated with its class label, that is each instance $\mathbf{I_i}$ is assigned to its class on the basis of its features.

2.1. Evaluation of an **RS**

The development of many data pre-processing techniques such as instance reduction was initially motivated by the imprecision and inefficiencies of the well-known nearest neighbour(s) (NN) algorithm [13]. These weaknesses have turned into strengths and made the NN rule a core algorithm to preprocess raw data [53]. Thus, most instance reduction techniques verify how well a candidate matrix **RS** represents the entire training dataset, **TR**, by using the NN algorithm as base classifier. To do so, this approach essentially checks

		1	2	3	---	p
Distance matrix	1	0.93	0.22	0.35		1.03
	2					
	3					
	---					
	l	0.21	1.08	1.08	---	1.50

Figure 1: Distance matrix of l instances in **TR** and p instances in **RS**. The instance at the first row is verified by instance at column 2, while the instance at the last row is checked by the one at column 1. Blue entries represent the shortest distance among the neighbours.

how well we can classify, the large dataset **TR** using the small dataset **RS** as training data, and consists of the following steps. The Euclidean distance between each instance $\mathbf{I_i}$ (row vector) of **RS** and each instance $\mathbf{I_j}$ (row vector) of **TR** is calculated. This process yields $l \times p$ distance computations. Typically, the nearest neighbours (smallest distances) are computed "on the fly", just by keeping the shortest distance and instance ID/number, and any intermediate distance computations are disregarded. As part of the strategy we will devise in Section 3.2, we could store all computed distances on a *distance matrix*; Figure 1 shows an example of a distance matrix. An entry $D_{i,j}$ of the distance matrix in position i, j indicates the distance of the i^{th} instance in **TR** to the j^{th} instance in the **RS**:

$$D_{i,j} = \sqrt{(b_{i,1} - a_{j,1})^2 + (b_{i,2} - a_{j,2})^2 + \ldots + (b_{i,m} - a_{j,m})^2}.$$

When the distance matrix **D** is calculated, for each row (i.e. each instance of **TR**), the smallest entry is detected, e.g. 0.22 in the first row of Figure 1, and that instance is given the class label w of the closest instance in **RS**. When all instances in **TR** have been classified, there are different ways to assign a score (objective function) to the performance of **RS** [2, 59].

As we are dealing with mostly balanced datasets, in this study we use the Accuracy Rate *Acc* [2, 59], that is

$$Acc = \frac{\text{number of correct classifications by means of } \mathbf{RS}}{\text{total number of examined samples}}$$

Thus, for an input **RS** the objective function value is *Acc*, that is

$$f(\mathbf{RS}) = Acc.$$

Algorithm 1 describes step-by-step the calculation of the objective function based on the distance matrix.

Algorithm 1 Objective Function

INPUT matrices $\mathbf{TR} = [a_{i,j}]$ and $\mathbf{RS} = [b_{i,j}]$
Build the matrix of Euclidean distances $\mathbf{D} = [D_{i,j}]$
for each row of the matrix **D do**
    Find the smallest number and save its row and column indices
    Select, from **TR** and **RS**, the instances corresponding to the calculated indices
    Check the corresponding labels
    if the labels coincide **then**
        Update the number of correct classifications
    end if
end for
Calculate *Acc*
OUTPUT the objective function value *Acc*

Of course, since higher values of *Acc* correspond to a better classification, the objective function needs to be maximised. The maximisation occurs within a $(p \times m)-$dimensional space where each variables can continuously vary in a normalised interval. Hence, the search for the optimal solution occurs in the set $[0, 1]^{p \times m}$.

2.2. Computational Cost of the Objective Function

Regardless of the score used to measure the quality of **RS**, the procedure described above requires the calculation of $l \times p$ Euclidean distances. This operation can be computationally expensive especially when large datasets are under examination. The two main problems that arise when tackling larger datasets are runtime (due to the large number of distance computations required) and memory consumption (e.g. when the size of **TR** does not allow us to store it in main memory). However, the required runtime to evaluate candidate solutions tends to be the most important factor to enable instance reduction of large datasets.

In the literature, two popular types of approach to accelerate the processing of instance reduction techniques are:

- **divide-and-conquer:** the execution of instance reduction approaches is parallelised, splitting the training data into a number of chunks, typically through big data technologies, see [55]. Whilst they are necessary when the **TR** set does not fit in main memory, the main limitation of this approach is that it does not address the computational complexity of the problem, but its processing time, by using additional computational resources. In addition, a trade-off between the number of splits and the accuracy that can be obtained exist, and must be experimentally found for the dataset at hand.

- **approximation:** to reduce the complexity of the objective function, the quality of **RS** may also be estimated by an approximation function. For example, a windowing approach that uses a different partition of **TR** to evaluate an **RS** at each iteration of a search algorithm [4]. Other more sophisticated approximation (also known as surrogate) models to reduce the number of evaluations for instance reduction algorithms have been recently investigated [25, 41]. While this approach reduces the runtime, its main limitation is that an approximated objective function may mislead the search of the optimisation algorithm.

In the present paper, we propose a new mechanism that while exploiting the structure of the optimisation algorithm allows a substantial reduction of the computational complexity (i.e. number of distance computations) of the objective function without approximations, see Section 3. Thus, the goal of this work is not to tackle big datasets and the memory limitations associated to it, but to devise a very fast and reliable instance reduction process that could be couple together with the approaches provided in [55] when very big datasets need to be addressed.

3. Single-Point Memetic Structure with Accelerated Local Search for Instance Reduction

From the description in Section 2, we may characterise Instance Reduction as an optimisation problem with the following considerations:

- the problem is large-scale and its number of variables ($p \times m$) can be extremely high depending on the size of the dataset

- due to the large number of variables, the problem is likely to be hard to solve and the fitness landscape could be highly multimodal

- even if it were multimodal, an excessive exploitation of the basin of attraction may yield an overfitted solution, that is a solution that performs well on the training set but not on the test set

- each objective function call (or fitness evaluation) is computationally expensive due to calculation of multiple Euclidean distances

In order to address the Instance Reduction problem, a domain-specific MC approach that takes into account the considerations above is here proposed. The proposed MC approach, namely Single-Point Memetic Structure with Accelerated Local Search (SPMS-ALS) is population-less and designed according to the bottom-up logic reported in [22]. SPMS-ALS perturbs a single solution and makes use of one more memory slot to store the elite solution, that is the best solution ever found. A novel domain-specific accelerated local search implementation is here proposed. Section 3.1 describes the local search operator employed in SPMS-ALS while Section 3.2 illustrates how the local search logic is exploited to accelerate the calculation of the objective function. The proposed SPMS-ALS makes also use of a simple global search operator illustrated in Section 3.3. Finally, Section 3.4 discusses and justifies the design of SPMS-ALS.

3.1. Local Search Operator

With the purpose of effectively describing SPMS-ALS, let us slightly redefine the notation. As introduced in Equation 2, $\mathbf{RS} = [b_{i,j}]$ is a matrix of size $p \times m$ which can be rewritten as a vector $\mathbf{x}$ of length $n = p \times m$ containing all the rows of $\mathbf{RS}$ arranged sequentially:

$$\mathbf{x} = (b_{11}, b_{12}, \ldots, b_{1m}, b_{21}, b_{22}, \ldots, b_{2m}, \ldots, b_{p1}, b_{p2}, \ldots, b_{pm}) = (x_1, x_2, \ldots, x_n)$$

where x_i represents the design variables of the optimisation problem.

Let $\mathbf{e^i}$ be the i^{th} versor (that is a vector of modulus equal to 1) of a basis in an n-dimensional space, that is a vector whose elements are all zeros except from the i^{th} element which is one [37]:

$$\mathbf{e^i} = (0, 0, \ldots, 1, \ldots, 0, 0)$$

The local search works on the candidate solution $\mathbf{x}$ to locally improve it. The following greedy implementation of a Generalised Pattern Search has

been used, see [40]. The algorithm perturbs each feature value of an instance at a time in its feasible range and then check if any improvement is found. Specifically, let $\mathbf{x}$ be the base vector (the best solution found at the time), for each design variable i from 1 to n the algorithm explores at first

$$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$$

where $\mathbf{x^t}$ is a trial vector and the scalar ρ is the step-size (exploratory radius). For each index i, the algorithm attempts to explore the opposite orientation of the direction identified by $\mathbf{e^i}$ if the first attempt fails, that is

$$\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$$

As a remark, the asymmetric step-size is designed to avoid to revisit the same solution (vector), see [40]. As soon as $\mathbf{x^t}$ outperforms $\mathbf{x}$, that is $f(\mathbf{x^t}) \geq f(\mathbf{x})$, the trial vector $\mathbf{x^t}$ replaces the base vector $\mathbf{x}$.

Note that when applying the above perturbations, the resulting values in the vector x^t could be outside of the bounds $[x_{low}, x_{high}]$. On the basis of preliminary tests we employed a toroidal handling of the bounds, i.e. for $x_i \in [x_{low}, x_{high}]$, if $x_i > x_{high}$ it is reinserted by reassignment:

$$x_i = x_{low} + \left((x_i - x_{high}) \lfloor \frac{(x_i - x_{high})}{(x_{high} - x_{low})} \rfloor (x_{high} - x_{low}) \right)$$

while if $x_i < x_{low}$ it is reinserted by reassignment

$$x_i = x_{high} - \left((x_{low} - x_i) - \lfloor \frac{(x_{low} - x_i)}{(x_{high} - x_{low})} \rfloor (x_{high} - x_{low}) \right)$$

where the parentheses $\lfloor \cdot \rfloor$ indicate the truncation to the lower integer.

As an example, if we are in the range [0,1], and the resulting value x_i of the perturbation is 1.1, the toroidal handling will begin from the beginning of the range, producing a 0.1. Conversely, if x_i were to be below 0, e.g. -0.1, this circular handling would provide 0.9. This ensures that the investigated values are within the range. Also, by forcing the perturbation to go to the other side of the bound, we increase the exploratory abilities of the method before reducing the radius ρ. This strategy provided good results in preliminary tests in comparison with other alternatives. If after the entire exploration along the n directions no improved solution $\mathbf{x^t}$ is found, then the radius ρ is reduced by a reduction rate. The local search is interrupted when either a

budget condition is met or when the radius ρ is smaller than a pre-arranged precision. For sake of clarity Algorithm 2 shows the local search operator used in SPMS-ALS.

Algorithm 2 Local Search of the family of Pattern Search used by SPMS-ALS

INPUT x
while local budget and precision conditions are not met **do**
    $\mathbf{x^t} = \mathbf{x}$
    for $i = 1 : n$ **do**
        $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$
        Apply toroidal handling of the bounds
        if $f(\mathbf{x^t}) \geq f(\mathbf{x})$ **then**
            $\mathbf{x} = \mathbf{x^t}$
        else
            $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$
            Apply toroidal handling of the bounds
            if $f(\mathbf{x^t}) \geq f(\mathbf{x})$ **then**
                $\mathbf{x} = \mathbf{x^t}$
            end if
        end if
    end for
    if x has **not** been updated **then**
        reduce ρ
    end if
end while
RETURN x

3.2. Accelerated Local Search

The proposed local search makes use of the search logic outlined in Algorithm 2 and integrates within it a domain-specific procedure to reduce the computational time of the algorithm. As highlighted in Section 2, when the NN algorithm is used as based classifier, most of the high computational cost of the Instance Reduction problem is due to the calculation of $l \times p$ Euclidean distances. However, the local search moves

$$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$$

and

$$\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$$

affect only one design variable that is only one entry of the **RS** matrix.

As a consequence, if we build a distance matrix **D** associated with $\mathbf{x^t}$, this differs by only one column from the matrix **D** associated with **x**. When the objective function $f(\mathbf{x^t})$ is calculated according to Algorithm 1, there is no need to recompute $l \times p$ Euclidean distances since $l \times (p-1)$ elements have already been computed and appropriately stored.

Thus, when Algorithm 2 is applied, each objective function call requires the calculation of only l Euclidean distances. This fact can be effectively represented as the modified objective function used by the local search outlined in Algorithm 3.

Algorithm 3 Objective Function $f(\mathbf{x^t})$ of the Accelerated Local Search

1: **INPUT** matrix $\mathbf{TR} = [a_{i,j}]$, matrix **D** associated with the base vector **x**, and trial vector $\mathbf{x^t}$
2: Build the matrix $\mathbf{RS} = [b_{i,j}]$ from $\mathbf{x^t}$
3: Update the matrix of Euclidean distances $\mathbf{D} = [D_{i,j}]$ by recalculating the l elements of the pertinent column
4: **for** each row of the matrix **D do**
5: Find the smallest number and save its row and column indices
6: Select, from **TR** and **RS**, the instances corresponding to the calculated indices
7: Check the corresponding labels
8: **if** the labels coincide **then**
9: Update the number of correct classifications
10: **end if**
11: **end for**
12: Calculate *Acc*
13: **OUTPUT** the objective function value *Acc*

Our proposed local search performs once at the beginning the objective function call as in Algorithm 1 and then integrates Algorithm 3 into each $f(\mathbf{x^t})$ function call for the rest of its execution.

3.3. Evolutionary Global Search Operator

At the beginning of the optimisation, a matrix **RS** (i.e. Equation 2) is randomly sampled from the matrix **TR** (i.e. Equation 1) and from **RS** the corresponding base vector $\mathbf{x}$ constructed and inputted into the local search operator. The local search is continued until the stopping criteria conditions on budget and precision are met. The local search returns a (possibly improved) solution $\mathbf{x}$. The best solution ever found is saved and stored in an elite slot and called $\mathbf{x^e}$. Then, a new solution $\mathbf{x^r}$ is generated by randomly sampling a new **RS** matrix from **TR** and constructing the corresponding vector. A uniform crossover is applied to $\mathbf{x^r}$ and $\mathbf{x}$ to generate a new trial vector $\mathbf{x^t}$.

In order to explain the functioning of this crossover, let us consider a candidate solution $\mathbf{x}$ and let us remind it that it corresponds to a matrix **RS** whose rows are instances and columns are features. By applying a matrix partitioning we may represent **RS** as a vector of row vectors

$$\mathbf{RS} = \begin{bmatrix} \mathbf{I_1} \\ \mathbf{I_2} \\ \dots \\ \mathbf{I_p} \end{bmatrix}$$

Similarly, we may consider the random solution $\mathbf{x^r}$ and represent the corresponding $\mathbf{RS^r}$ matrix

$$\mathbf{RS^r} = \begin{bmatrix} \mathbf{I_1^r} \\ \mathbf{I_2^r} \\ \dots \\ \mathbf{I_p^r} \end{bmatrix}$$

whose instances are randomly selected from **TR**.

The proposed crossover generates a trial vector $\mathbf{x^t}$ by randomly selecting some rows from **RS** and some rows from $\mathbf{RS^r}$. Each row of the resulting matrix $\mathbf{RS^t}$ has a gene-resampling probability Gr to be selected from **RS** and $1 - Gr$ probability to be selected from $\mathbf{RS^r}$. It must be remarked that a crossover that perturbs single elements of **RS** instead of entire rows would yield a candidate solution which could be noisy (i.e. not have the right class label), and therefore, not meaningful from a classification point of view.

The gene-resampling probability Gr expresses the rate of the instances in **RS** which are replaced by other instances sampled from **TR**. Algorithm 4 describes the crossover mechanism.

Algorithm 4 Crossover between $\mathbf{x}$ and $\mathbf{x^r}$

INPUT base vector $\mathbf{x}$ and random vector $\mathbf{x^r}$
Build the matrices $\mathbf{RS} = [\mathbf{I_i}]$ and $\mathbf{RS^r} = [\mathbf{I_i^r}]$
$\mathbf{RS^t} = [\mathbf{I_i^t}] = \mathbf{RS}$
for $i = 1 : p$ **do**
    Generate a random number $rand$
    if $rand < Gr$ **then**
        $\mathbf{I_i^t} = \mathbf{I_i^r}$
    end if
end for
From $\mathbf{RS^t}$ calculate $\mathbf{x^t}$
OUTPUT the trial vector $\mathbf{x^t}$

The local and global search operators are repeated until the global budget conditions are met. The framework of the proposed SPMS-ALS is illustrated in Algorithm 5.

3.4. Algorithmic Design

The proposed SPMS-ALS follows a bottom-up strategy as suggested in [22]: we implemented within the algorithmic operators the necessary countermeasures to address each challenge associated with the problem.

The structure of the local search has been selected to address the large scale nature of the instance reduction problem, that is for a large dataset, the matrix **RS** can easily have hundreds if not thousands of rows. The proposed local search perturbs the variables separately and thus implicitly performs a variable decomposition. Approaches of this type have been proved effective for large scale problems, see [47, 56, 28].

This observation was reported in the experimental study in [8]. Large scale problems are by no means easier than low-dimensional problems. However, since in practice the computational budget cannot grow exponentially with the problem dimensionality only a very limited portion of the decision space is explored. Under these experimental conditions, the algorithm "sees" the problem as separable: average Pearson and Spearman coefficients of the variables approach zero independently on the problem when the number of dimensions grows, see [8].

The high computational cost of each function call is addressed by the acceleration mechanism outlined above: only the elements of one column of

Algorithm 5 Framework of the SPMS-ALS for Instance Reduction

Randomly generate a base vector $\mathbf{x}$ in $[0,1]^n$ and calculate $f(\mathbf{x})$ according to Algorithm 1
Assign the elite $\mathbf{x^e} = \mathbf{x}$
while global budget conditions are met **do**
    Apply the Accelerated Local Search to the base vector $\mathbf{x}$ according to Algorithm 2 with the objective function $f(\mathbf{x^t})$ calculated according to Algorithm 3
    if $f(\mathbf{x}) \geq f(\mathbf{x^e})$ **then**
        Update the elite $\mathbf{x^e} = \mathbf{x}$
    end if
    Randomly generate a vector $\mathbf{x^r}$ in $[0,1]^n$
    Apply Crossover between $\mathbf{x}$ and $\mathbf{x^r}$ according to Algorithm 4 and generate a new trial vector $\mathbf{x^t}$
    Calculate $f(\mathbf{x^t})$ according to Algorithm 1
    if $f(\mathbf{x^t}) \geq f(\mathbf{x^e})$ **then**
        Update the elite $\mathbf{x^e} = \mathbf{x^t}$
    end if
    Assign $\mathbf{x} = \mathbf{x^t}$
end while

the Euclidean matrix $\mathbf{D}$ and not those of the entire matrix are calculated at each function call. The population-less structure of SPMS-ALS has also been chosen taking into consideration the computational cost. The proposed SPMS-ALS naturally devotes most of the computational budget (in terms of function calls) to the local search. On the contrary, the global search operator performs only sporadic function calls. This logic perfectly suits the needs of reducing the computational cost since the global operator requires the expensive objective function as in Algorithm 1 the local search operator uses its computationally cheaper version as in Algorithm 3.

In order to address the multimodality of the fitness landscape and prevent that the algorithm converges to a suboptimal solution, we combined the Accelerated Local Search with the simplistic global search described above. It must be noted that the global search makes use of part of decision variables (genotype) of previously improved solutions. The elitism guarantees that previously detected promising solutions are available at the end of the run. Furthermore, the gene-resampling mechanism, happening at the in-

stance level (considering the rows as building blocks) complements the local search that happens at the level of the elements of **RS**.

At last, the restarting local search logic combined with a limited local search budget is an important countermeasure to prevent from overfitting: an excessive local search budget is likely to yield an overly specialised solution that performs poorly when the solution is tested on a new dataset. This characteristic is experimentally analysed in Section 5.3.

4. Experimental Framework

This section presents the used datasets (Section 4.1) and is followed by introducing several instance reduction techniques that will be used for comparison with our proposal (Section 4.2). Finally, the parameter configuration is explained (Section 4.3).

4.1. Datasets

In the experimental study, we have examined 40 small and 17 medium multi-class datasets from the KEEL dataset repository [54]. The properties of these datasets including name (**Dataset**), the number of samples (**Samp**), the number of attributes (**Att**), the number of classes (**%Class**) are summarised in Table 1.

As defined in [51], small datasets have less than 2000 instances while medium datasets have at least 2000 instances. Each dataset is partitioned using a 10-fold stratified cross-validation (10-fcv) procedure, see [45]. Thus, the performance of each dataset is reported by an average of the 10 folds. All of the experiments with these datasets have been conducted on computers at which each has 2 x 20 core processors (Intel Xeon Gold 6138 20C 2.0GHz CPU) and 192 GB RAM.

4.2. Comparison Algorithms

In order to understand the benefits of the proposed MC approach, we first define two baselines:

- Nearest Neighbour (1NN): we use the NN algorithm (k=1) employing the entire **TR** for training, without any pre-processing. The performance of the NN in **TR** is calculated following a *leave-one-out* validation scheme. This serves of a baseline to understand the benefits of instance reduction.

- Local Search Instance Reduction (LSIR): the local search presented and used in [41]. LSIR is essentially the basic pattern search shown in Algorithm 2 without any acceleration, that is the local search by using the basic fitness function as in Algorithm 1.

Table 1: Summary description for small (Sample < 2000) and medium (Sample >= 2000) datasets.

Dataset	**Samp**	**Att**	**Class**	**Dataset**	**Samp**	**Att**	**Class**
Abalone	4174	8	28	Monks	432	6	2
Appendicitis	106	7	2	Movement_libras	360	90	15
Australian	690	14	2	Newthyroid	215	5	3
Autos	205	25	6	Nursery	12960	8	5
Balance	625	4	3	Page-blocks	5472	10	5
Banana	5300	2	2	Penbased	10992	16	10
Bands	539	19	2	Phoneme	5404	5	2
Breast	286	9	2	Pima	768	8	2
Bupa	345	6	2	Ring	7400	20	2
Car	1728	6	4	Saheart	462	9	2
Chess	3196	36	2	Satimage	6435	36	7
Cleveland	297	13	5	Segment	2310	19	7
Contraceptive	1473	9	3	Sonar	208	60	2
Crx	125	15	2	Spambase	4597	57	2
Dermatology	366	33	6	Spectheart	267	44	2
Ecoli	336	7	8	Splice	3190	60	3
Flare-solar	1066	9	2	Tae	151	5	3
German	1000	20	2	Texture	5500	40	11
Glass	214	9	7	Thyrod	7200	21	3
Haberman	306	3	2	Tic-tac-toe	958	9	2
Hayes-roth	133	4	3	Titanic	2201	3	2
Heart	270	13	2	Twonorm	7400	20	2
Hepatitis	155	19	2	Vehicle	846	18	4
Housevotes	435	16	2	Vowel	990	13	11
Iris	150	4	3	Wine	178	13	3
Led7digit	500	7	10	Wisconsin	683	9	2
Lymphography	148	18	4	Yeast	1484	8	10
Magic	19020	10	2	Zoo	101	16	7
Mammographic	961	5	2				

In addition, we test the performance of the proposed approach against the current state-of-the-art in instance reduction. SPMS-ALS belongs to the family of positioning adjustment methods (see [51]), which are, to date, the best performing instance reductions methods in the literature and follow a similar algorithmic structure to the proposed approach. In [41], we showed the classification performance of the local search against the entire family

of positioning adjustment methods. For the sake of simplicity, here we only report the comparison against the most competitive methods. SPMS-ALS can be categorised as a pure instance generation approach, as we perform a continuous search. Thus, we choose the following metaheuristics instance generation methods to compete against SPMS-ALS:

- Scale Factor Local Search Differential Evolution (SFLSDE): this memetic approach optimises the positioning of prototypes using an implementation of differential evolution [52].
- Particle Swarm Optimisation (PSO): this algorithm modifies the position of an initial set using PSO rules, aiming to maximise the classification performance [36].

Additionally, to compare against more recent meta-heuristics, we have adapted a recent metaheuristic, proposed for the continuous domain, to tackle instance reduction.

- Linear Population Size Reduction of the Success-History based Adaptive Differential Evolution (LSHADE) [49]: this approach is developed from Success-History based Adaptive Differential Evolution (SHADE) [48] and Adaptive Differential Evolution with Optional External Archive JADE [62]. It makes use of success-history and also applies the population size reduction to progress the search. Note that this metaheuristic has not been previously used for instance reduction, but due to its similarity to JADE, we used the design ideas from [52] to adapt it to solve the instance reduction problem.

These approaches evolve a population of solutions, whilst our method only evolves a single solution (or more precisely two solutions the trial solution $\mathbf{x^t}$ and the elite $\mathbf{x^e}$). However, similar to our method, both approaches start off from a random (stratified) subset of the training set $\mathbf{TR}$ (one for each individual of their population), which keeps the original distribution of instances per class. Thus, the reduction rate is also defined by a parameter that determines how much we want to reduce $\mathbf{TR}$.

As a further remark, while PSO, SFLSDE were existing metaheuristics that have been adapted to solve the instance generation problem, the proposed SPMS-ALS has been expressly designed to solve this problem effectively in terms of accuracy efficiently in terms of runtime. This design approach follows the bottom-up design logic of MC [22, 38] and can be observed in both accelerated local search logic and crossover operator. Another

remark is that LSHADE was used in the continuous domain to solve benchmark functions, this metaheuristic design is first time adapted to solve the instance generation problem in this study.

In [52], the authors showed that using a random selection as initialisation mechanism is not usually appropriate, and the hybridisation of an instance selection step followed by instance generation was suggested to replace this random initialisation. More specifically, the use of a Steady-State Memetic Algorithm (SSMA) [18] demonstrated empirically to provide an excellent starting point, which means a good selection of instances per class and a good reduction rate (automatically determined by the instance selection step). To the best of our knowledge, the hybrid instance reduction algorithm, SSMA-SFLSDE [52], remains the best performing method for Instance Reduction in both accuracy and reduction rate. To establish a fair comparison against it, we will also hybridise the proposed MC approach and the local search with SSMA (see Section 5.5).

Whilst the hybrid instance selection/generation method, SSMA-SFLSDE has not been outperformed to date, in order to assess the potential of the proposed approach, we add a comparison with recently published algorithms belonging to the family of instance selection. We included an approach based on local sets [27] and a method based on instance ranking [10]. Note that these methods follow a completely different approach to produce a reduced set from the training set. Thus, we cannot set up the same computational budget that we do for the rest of the comparison algorithms, as they do not follow an optimisation-based approach (Section 4.3.1).

The study in [27] contains three instance selection methods, namely Local Set Smoother (LSSm), Local Set Core (LSCo) and Local Set Border (LSBo). LSSm aims at achieving the highest accuracy regardless of the reduction, while LSCo seeks at obtaining the highest reduction with acceptable accuracy. LSBo addresses both accuracy and reduction rate with the same priority. For this reason, LSBo has been selected for comparison against the proposed memetic approach.

The main idea of [10] is to exploit the relationship among members in the training set by computing a rank for each element. A rank of an instance introduces the correlation between itself and others in the training set. Instances with higher ranks are likely to be selected compared to those holding a low rank. In Section 5.4.2, we report the performance of Ranking-based Instance Selection (RIS1) as it showed in [10] to display the best performance, among multiple variants, w.r.t both measures of accuracy and reduction rate.

4.3. Parameter Settings

This section presents the parameter configuration for all the methods employed in this study, including the accelerated local search outlined in Algorithms 2 and 3 and the entire memetic framework SPMS-ALS shown in Algorithm 5. Subsection 4.3.1 focuses on the computational budget while Subsection 4.3.2 discusses the other parameters.

4.3.1. Computational Budget

In the previous studies on instance reduction, the computational budget has usually been set empirically to a number of iterations (in search-like algorithms), which remained fixed for all datasets [52, 41]. Just like in scalability studies for ordinary optimisation problems, in the instance reduction problem, the complexity of the search space grows exponentially with the problem size, [8]. On the other hand, instance reduction poses a further challenge that is the risk of overfitting and underfitting. An incorrect local search budget allocation is likely to lead to overfitting in small datasets and underfitting appears in larger datasets. In order to overcome this challenge and propose a standard for setting the computational budget, we have here conducted an extensive experimental study. Note that keeping the same number of evaluations through the different comparison methods will also help establish a fairer comparison (which has not been the case in previous studies).

Among the various properties of a dataset, the number of instances in training data, i.e. the number of rows l of the matrix $\mathbf{TR}$ and the number of features (**Features**) in an instance at each dataset are the two important factors that define the size of the problem and need to be considered when the budget is allocated. We acknowledge that other factors may be also required into consideration such as the number of classes or the ratio of samples among classes. However, this simple yet effective approach of parameter setting has proven to highly reduce the unnecessary allocated number of evaluations and thus can help mitigate overfitting in small datasets and underfitting in larger ones. Since RIS1 and LSBo do not perform any evaluation of their reduced set against the training set, we cannot apply a computational budget.

In the original setting based on forty small datasets, SFLSDE [52] and LSIR [41] use approximately 20,000 and 30,000 evaluations, respectively. We took these values as a reference and set three levels in our experimental study: lower, comparable, and greater than the reference ones. Table 2 displays, for all the algorithms considered in this study that employ local search, the three local search budgets scenarios. From the total number of evaluations

presented in Table 2, we split the evaluations into two parts when SSMA is included. SSMA takes $3 \times l$ evaluations in Setting 1 and $5 \times l$ in Setting 2 and 3. The budget allocated to SSMA is indicated as SSMA_Eval. The rest of the evaluations is used for instance generation methods (LSIR, SPMS-ALS, PSO, SFLSDE, LSHADE).

Table 2: Changing the number of evaluation considering training size and features for fairer comparison.

	Computational Budget		
Algorithm	Setting 1	Setting 2	Setting 3
SSMA	SSMA_Eval = $3 \times l$	SSMA_Eval = $5 \times l$	SSMA_Eval = $5 \times l$
SFLSDE	$2.5 \times$ **Features** $\times l$ − SSMA_Eval	$5 \times$ **Features** $\times l$ − SSMA_Eval	$10 \times$ **Features** $\times l$ − SSMA_Eval
LSIR			
SPMS-ALS			
PSO			
LSHADE			

4.3.2. Parameters

The proposed SPMS-ALS contains some parameters to set to coordinate global and local search. In particular, the following parameters are fundamental to coordinate the interruption of accelerated local search and restart of global search.

- $N_{\max}$: the maximum number of times the local search accepts a new trial solution $\mathbf{x^t}$ with the same objective function (Acc) as that of the previous trial solution i.e. maximum number of search moves allowed on a plateau
- ρ_{Red}: the reduction rate of the exploratory step ρ after the same fitness has been calculated $N_{\max}$ times
- ρ_{Thr}: the threshold after which the local search is interrupted
- Gr: the gene-resampling probability as in Algorithm 4

Since small datasets have only few samples per class, a large Gr value is required to make a significant refresh of the candidate solution. On the contrary, medium datasets inherently pose a highly multivariate problems. Hence smaller Gr values result into a major alternation of the candidate solution. We may consider this effect analogous to the setting of the crossover

rate in Differential Evolution with respect to the number of dimensions of the problem, see [39]. On the other hand, numerous configurations have been examined to find a set of parameters that can guarantee a robust performance of SPMS-ALS on both small and medium datasets. In this study, we report the performance of SPMS-ALS using the following parameters: initial radius $\rho = 0.4$, $N_{\max} = 3$, $\rho_{Red} = 0.25$, $\rho_{Thr} = 0.005$ and $Gr = 0.5$ for small and 0.05 for medium datasets, respectively. Apart from the budget condition, which is investigated for all the comparison algorithms as described above, the rest of the parameters for all the algorithms are established as recommended by the authors. All the details are presented in Table 3. Following the experimental setup in [51], the reduction rate parameter is set to 95% for small size datasets, and 98% for medium datasets.

Table 3: Parameters used for comparison algorithms

Algorithms	Parameter setting
SFLSDE	PopulationSize = 40, iterSFGSS =8, iterSFHC = 20, Fl = 0.1, Fu = 0.9
LSIR	initial $\rho = 0.4$
SSMA	Population = 40, Cross = 0.5, Mutation=0.001
PSO	SwarmSize = 40, C1 = 1, C2 = 3, Vmax = 0.25, Wstart = 1.5, Wend = 0.5
NN	k = 1, Euclidean distance
RIS	Thresholds = [0.0, 0.1, 0.2, ..., 0.9, 1.0]
LSBo	–
LSHADE	ArchiveSize = 1.4, PopulationSize = 40, MemorySize = 5

5. Analysis of results

In this section, we analyse the results obtained from different sets of experiment, divided into multiple subsections, to empirically examine the individual effect of each component we propose in our algorithm. In the analysis, our aims are:

- To understand how well LSIR works in different settings of evaluations (Subsection 5.1). We discuss multiple aspects of LSIR such as the change in performance measured by accurate rate to see the overfitting or underfitting effects on the learning process. In addition, the number of evaluations that has been used and saved for each dataset is reported.

- To measure the actual savings in terms of runtime when the proposed acceleration is integrated within LSIR (Subsection 5.2). We report in detail the absolute and percentage figures of the runtime savings.

- To examine the performance enhancement due to the proposed memetic components (Subsection 5.3), in comparison with the local search. We report the accuracy rate depending on the number of evaluations and we analyse the statistical significance of the improvements.

- To compare the performance of SPMS-ALS with the state-of-the-art techniques in the family of instance reduction, with a focus on instance generation (Subsection 5.4.1) considering the 1NN rule as a baseline and recent instance selection methods (Subsection 5.4.2). In addition, the average runtime required by each algorithm is contrasted to highlight the substantial computational saving in the proposed method.

- To establish a fair comparison between the proposed approach and the state-of-the-art algorithm in the family of instance reduction with hybrid instance selection and generation algorithm, SSMA-SFLSDE, using the same memetic instance selection algorithm as initialisation mechanism (Subsection 5.5).

- To contextualise the results presented in this paper by comparing the performance of SPMS-ALS with a recently proposed classifier (obRaF(H)) which represents a robust algorithm in the field of classification [24] (Subsection 5.6).

For the sake of space, this section will only present summary results, and all the detailed results can be found in the Supplementary Material and the associated GitHub repository[1].

5.1. LSIR Running with Different Computational Budgets

The Local Search LSIR as Shown in Algorithm 2 has been run with the three budget settings outlined in Table 2 to understand the influence of the budget allowance in the performance of this algorithm. Its average classification performance in the training and test phase is displayed in Table 4 on small and medium datasets. Note that the reported performance is obtained from using Algorithm 1 changing **TR** by **TS** to evaluate LSIR in the test phase. Analysing these average results and the detailed results in the supplementary material, we can make the following comments:

[1]https://github.com/lehoanglam20000/SPMS-ALS

- In the training phase the computational budget has a major impact on the performance. However, while the performance grows consistently from Setting 1 to Setting 3, this improvement is not major when the performance of Setting 2 and Setting 3 are compared. This may infer that changing each feature value cannot help the search seek a better solution after a certain number of function calls.

- Regarding the test performance, we observe that the results are dramatically different to those achieved during the training phase: Setting 1 achieves most of the wins in test data overall. In small datasets, Setting 1 has 22 wins out of 40, while Setting 2 and 3 win 14 and 11 times, respectively. In medium datasets, Setting 1 has 9 wins out of 17, while Setting 2 and 3 win 6 and 8 times, respectively. We conclude that overfitting is likely to happen for LSIR (possibly due to its exploitative structure) in Setting 2 and 3. This tendency appears evident in small size datasets.

Table 4: Average training and test performance in different settings of LSIR over small and medium datasets.

	Training		Test	
	Small	Medium	Small	Medium
Setting 1	0.8521 ± 0.0128	0.9005 ± 0.0039	0.7411 ± 0.0605	0.8612 ± 0.0139
Setting 2	0.8657 ± 0.0128	0.9049 ± 0.0039	0.7419 ± 0.0614	0.8610 ± 0.0133
Setting 3	0.8693 ± 0.014	0.9052 ± 0.0041	0.7415 ± 0.0607	0.8609 ± 0.0136

In summary, we can conclude that this local search does not seem to benefit from using a larger budget and, as possibly expected, may be falling into local optima, which do not generalise well in terms of classification performance. This is especially noticeable in medium size datasets in which the average training performance does not seem to increase much in respect to the number of evaluations.

To further illustrate the behaviour of the LSIR approach with respect to the number of evaluations, Table 5 shows the effect of its stopping criteria. More specifically, when LSIR does not succeed at enhancing upon the trial solution $\mathbf{x^t}$ (see Algorithm 2), the exploratory step decreases by the factor ρ_{Red} until a threshold value ρ_{Thr} is met. When these conditions are met the run of LSIR is interrupted. Table 5 displays the computational budget saving caused by the interruption of the run. The savings are shown for small and medium datasets and for each of the setting under consideration. For each

configuration of dataset and setting the number of function calls used by the algorithm is also shown.

Table 5: Number of evaluations used and saved by LSIR in different settings and datasets.

	Small datasets			Medium datasets		
	Used	Saved	(%) Saved	Used	Saved	(%) Saved
Setting 1	15398	117	0.76	290777	0	0.00
Setting 2	30795	562	1.83	581554	54094	9.30
Setting 3	61591	2966	4.82	1163108	588637	50.61

Table 5 shows that Setting 1 mostly uses up the allocated number of evaluations, whilst Setting 2 saves 1.83% and 9.3% in small and medium datasets, respectively. Setting 3 spends most of the evaluations in small datasets but only consumes nearly half of the allocated number of evaluations. These figures may help optimise the number of evaluations used for each dataset based on their size and features. On the other hand, the allocation of a very large budget to the local search (like Setting 3) may not be always beneficial, and as mentioned above, the algorithm seems to get trapped into local optima.

5.2. Runtime Reduced in the Accelerated Version of LSIR

This subsection reports the runtime used by LSIR and how much it is reduced from its accelerated version using Algorithm 2 and the fitness in Algorithm 3, here referred to as Accelerated Local Search for Instance Reduction (ALSIR).

Of course, the time required by the local search depends directly on the allocated budget and when the stopping criteria is reached. It is also important to remember that ALSIR always provides exactly the same classification performance as LSIR, this is because ALSIR focuses on accelerating the execution of the proposed method but it does not change the behaviour of the algorithm at all. The objective of the section is therefore to show how much we can accelerate LSIR with the proposed acceleration strategy.

Details of the runtime of both LSIR vs ALSIR in small and medium datasets, respectively, with respect to each setting of the number of evaluations can be found in the Supplementary material. Table 6 summarises the average runtime for each setting and the average percentage of time saved by the proposed acceleration. On average in small datasets, the objective function of the accelerated local search as in Algorithm 3 enables a time reduction from at least 66% to 84.29%. However, the average runtime saved in

medium datasets settles around 90% in the three settings. Thus, the larger the dataset the more we can benefit from the proposed acceleration strategy, as distance computations become the most dominant part of the execution of the local search.

Table 6: Average runtime (in seconds) saved in different settings of LSIR and ALSIR, smaller values are in bold.

	Small datasets			Medium datasets		
	LSIR	ALSIR	(%) Time saved	LSIR	ALSIR	(%) Time saved
Setting 1	6.98	**2.35**	66.25	8676.67	**856.05**	90.13
Setting 2	19.47	**3.60**	81.54	15957.94	**1508.85**	90.54
Setting 3	37.68	**5.92**	84.29	18061.49	**1784.64**	90.12

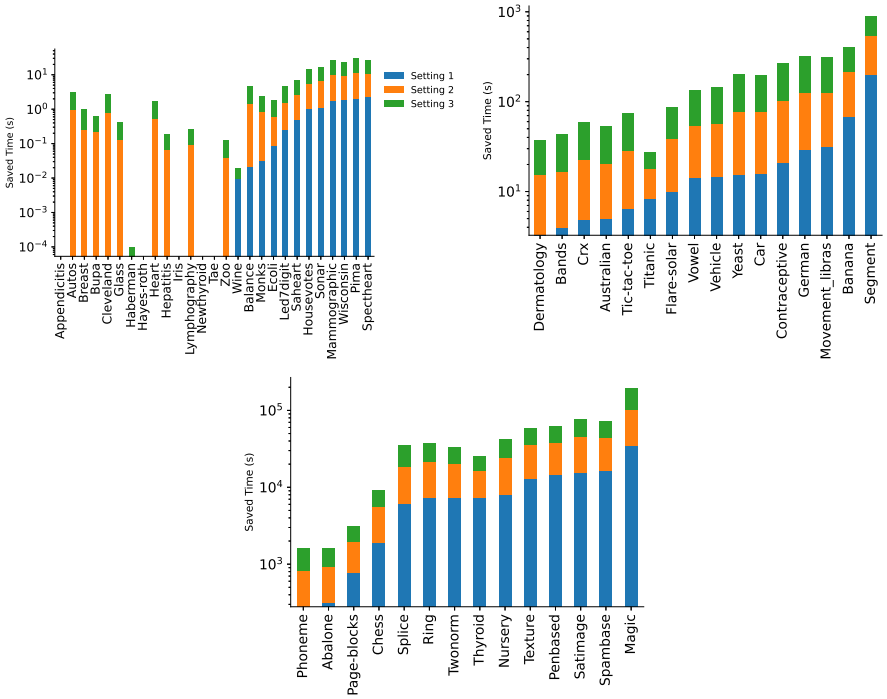


Figure 2: Runtime saved across 57 datasets, sorted by the ascending order of time gaps in Setting 1.

To illustrate the runtime reduction depending on the dataset size, Figure 2 depicts the difference in runtime between LSIR and ALSIR for all the datasets, providing a graphical representation of the average time saving for each dataset. In order to enhance the readability of the diagram, the

logarithmic scale has been used. Those datasets which appear to have no value represent those scenarios where the search can be completed is less than a second. Hence, the acceleration may not be essential in these cases.

5.3. Validation of the Memetic Framework of SPMS-ALS

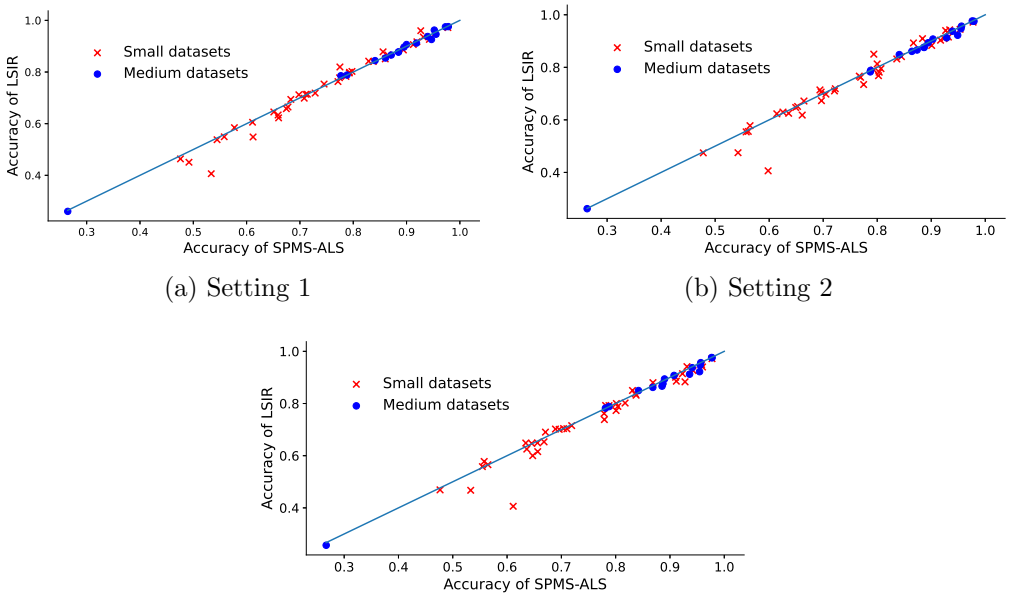
In this section, we compare the performance of LSIR and SPMS-ALS to demonstrate the effectiveness of the proposed memetic framework. Table 7 provides a full summary of this comparison, presenting the average accuracy values (over all the datasets) and the corresponding standard deviations in the three settings of computational budget in both training and test phases. The best average results in training and test are highlighted in bold face.

Furthermore, the Wilcoxon test [57] is also applied to detect the statistical differences between the two methods. The corresponding p-values are also shown in the last column of Table 7. When one algorithm significantly outperforms the other, the p-value is less than the confidence level 0.05. We highlight in italic these p-values.

Table 7: Comparison in average training and test performance between LSIR and SPMS-ALS over small and medium datasets. Wilcoxon p-value is obtained from the comparison between SPMS-ALS and LSIR.

		SPMS-ALS				LSIR				Wilcoxon
		TRAINING		TEST		TRAINING		TEST		p-value
SMALL	Evaluations	Acc	Std	Acc	Std	Acc	Std	Acc	Std	
	Setting 1	0.8598	0.0130	0.7477	0.0633	0.8521	0.0128	0.7411	0.0605	0.1015
	Setting 2	0.8665	0.0122	0.7512	0.0625	0.8657	0.0128	**0.7419**	0.0614	0.0867
	Setting 3	**0.8733**	0.0110	**0.7549**	0.0615	**0.8693**	0.0140	0.7415	0.0607	*0.0132*
MEDIUM	Setting 1	0.9126	0.0034	0.8625	0.0127	0.9005	0.0039	**0.8612**	0.0139	>0.2
	Setting 2	0.9129	0.0033	0.8626	0.0126	0.9049	0.0039	0.8610	0.0133	>0.2
	Setting 3	**0.9199**	0.0028	**0.8668**	0.0110	**0.9052**	0.0041	0.8609	0.0136	*0.0577*

Numerical results in Table 7 show that for both training and test phases, the memetic framework outperforms on a regular basis LSIR. We may observe that in training phase and small datasets, SPMS-ALS slightly outperforms LSIR while the difference in performance is larger for medium datasets. According to our interpretation, this shows the effectiveness of the global search component in complex spaces: while the local search exploits the space and is likely to achieve a suboptimal point (we may see that different computational budgets do not yield major changes in LSIR performance), the crossover allows the search a further chance to detect a solution closer to the global optimum. The results in the test phase display a consistent better performance of the memetic framework across the datasets. This finding can be



(a) Setting 1

(b) Setting 2

(c) Setting 3

Figure 3: Accuracy scatter plots over 40 small and 17 medium datasets in the test phase.

interpreted as a better performance of SPMS-ALS in terms of overfitting: the deterministic and exploitative nature of LSIR may lead to overfitting while the degree of randomisation introduced by the crossover-based global search element reduces the risk of overfitting hence improving upon the performance of the algorithm in test phase. Finally we may observe that SPMS-ALS statistically outperforms LSIR in Setting 3 in small datasets and shows improved progress in medium datasets. This fact is expected since longer runs tend to be more stable and thus be associated with lower standard deviation values. On the contrary, with Setting 1 and 2 we are more likely to observe "lucky" or "unlucky" runs that may jeopardise the statistical significance of the results.

The test results are also graphically presented in Figure 3 which contains scatter plots of the accuracy of the methods. Each point compares the test performance of SPMS-ALS and LSIR algorithm on a single dataset. The accuracy of SPMS-ALS is shown on the x-axis position of the point, while that of LSIR is on the y-axis position. Thus, points below the $y = x$ line correspond to datasets for which SPMS-ALS achieves better performance

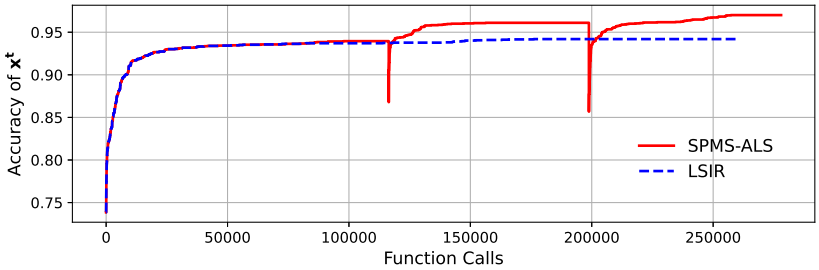


Figure 4: Functioning (Accuracy) of SPMS-ALS and LSIR on the Chess dataset.

than the compared algorithm. In most of the cases, the points are plotted on or below the separating line, inferring greater performance of SPMS-ALS. In this plot, we can also see that the biggest improvements have been made in small datasets, but in turn, there are a few datasets in which SPMS-ALS performs slightly worse. However, in medium size datasets the improvements are less significant, especially in settings 1 and 2, but consistently better.

In order to emphasise the different behaviour of LSIR vs SPMS-ALS we plot in Figure 4 the accuracy of the trial solution $\mathbf{x^t}$ against function calls (evaluations) of the two algorithms on the Chess dataset, using Setting 1. To show the functioning of the crossover the plot of SPMS-ALS refers to the local solution (and not the elite). We may observe that the crossover functions as a restart which then quickly reaches a solution with a good performance.

In conclusion, whilst the local search seemed to get stuck after a number of evaluations, the proposed MC approach, despite its simplicity, benefits from larger computation budgets, outperforming the local search.

5.4. Comparison with the State-of-the-art Methods in Instance Reduction

This section consists of two sub-sections: Section 5.4.1 covers the comparison of our proposal with related instance generation methods; Section 5.4.2 presents the comparison with recently published instance selection methods.

5.4.1. Comparison against similar instance generation techniques

In order to compare the performance of SPMS-ALS against that of the other global optimisers for instance generation (PSO [36], SFLSDE [52], and the adapted LSHADE [49]), we will focus on the maximum number of evaluations (Setting 3) for all the algorithms. As a baseline, we also include the 1NN algorithm as a comparison algorithm.

Table 8 summarises the performance of the comparison algorithm in all (57) datasets (small and medium). We have employed the Friedman procedure [20] plus a Holm post-hoc test to perform a ranking-based statistical analysis on the performance of the algorithms for small and medium datasets, respectively. The last two columns of Table 8 provide the results of these tests, including the rankings and the resulting p-values. Note that the control method will obtain the lowest ranking, and therefore, the p-value shows if the differences are significant comparing the control algorithm against the rest of the methods.

As shown in Table 8, LSHADE and SFLSDE are reported as the control method in small and medium datasets, respectively, since they hold the smallest ranking values. In small datasets, our proposal SPMS-ALS ranks third after SFLSDE, while it ranks second in the medium datasets and its ranking value is not far away from that of the control algorithm.

The Holm post-hoc test is used to detect if there is any significant statistical differences between the control algorithm (LSHADE and SFLSDE) with respect to the remaining methods. Considering a level of significance of $\alpha = 0.05$, LSHADE statistically outperforms only 1NN in small datasets, and PSO in medium datasets. The statistical tests have not reported significant differences between our proposal and the control algorithm in either small or medium datasets.

Table 8: Summary of the performance of SPMS-ALS against SFLSDE, PSO, LSHADE and 1NN for instace reduction over 57 datasets. The best performance in the column is shown in bold.

		TRAINING		TEST		Friedman+Holm	
	Algorithm	Acc	Std	Acc	Std	Ranking	p_{Holm}
SMALL	LSHADE	0.8401	0.0165	0.7541	0.0612	**2.425**	–
	SFLSDE	0.8480	0.0092	**0.7615**	0.0634	2.525	0.7773
	SPMS-ALS	**0.8733**	0.0110	0.7549	0.0615	3.125	0.1017
	PSO	0.8147	0.0156	0.7414	0.0606	3.175	0.1017
	1NN	0.7369	0.0088	0.7369	0.0088	3.750	**0.0007**
MEDIUM	SFLSDE	0.8887	0.0048	0.8608	0.0122	**2.177**	–
	SPMS-ALS	**0.9199**	0.0028	**0.8668**	0.0110	2.353	0.7448
	LSHADE	0.8859	0.0138	0.8503	0.0147	3.294	0.1180
	1NN	0.8316	0.0045	0.8316	0.0045	3.294	0.1180
	PSO	0.8537	0.0066	0.8319	0.0137	3.882	**0.0066**

According to our interpretation, in training phase an exploitative action guarantees a better performance of the algorithm especially in high dimensions [8]. However, the exploitative pressure should be counterbalanced by a certain degree of randomisation to prevent the algorithm from overfitting

and pay off with a deteriorated performance in test phase. This feature of the instance generation problem makes it especially suitable to be tackled by memetic frameworks. Albeit reasonable, the excessively exploratory nature of PSO does not appear to effectively address the large dimensional space.

In comparison with the baseline, 1NN, which uses all the data to classify the test set, we have conducted the Wilcoxon test to conduct a pairwise comparison to our method. Although SPMS-ALS shows better average performance , the Wilcoxon test compute p-value 0.0415 for small datasets and >0.2 for medium datasets. These numeric p-values indicates that our algorithm statistically outperform 1NN in small dataset, but has no significant different in medium datasets, considering a level of significance of $\alpha = 0.05$.

Finally, Table 9 displays the average runtime of the four global optimisers for the small and medium datasets, respectively. On average, the runtime spent for small datasets is 6.25s, saving 92.52%, 83.93% and 28.17% with respect to LSHADE, SFLSDE and PSO, respectively. For medium datasets, the percentage of saving time is slightly higher than what it did in small datasets, but the absolute value is more meaningful. Specifically, SPMS-ALS only consumes roughly 5000s on average, while SFLSDE and PSO experience about 35000s, and LSHADE used up to approximate 160.000s. In other words, for medium datasets, SPMS-ALS achieves similar if not better results of SFLSDE and LSHADE in one seventh and less than one thirtieth of the runtime, respectively.

Table 9: Comparison of the runtime (in seconds) consumed in SPMS-ALS and other approaches. Min values are in bold.

	Small datasets	Medium datasets
SFLSDE	38.89 ± 1.31	34738.82 ± 188.55
PSO	19.63 ± 0.80	34941.65 ± 188.86
LSHADE	83.63 ± 4.12	159009.10 ± 1154.75
SPMS-ALS	**6.25** ± 0.39	**5006.93** ± 405.33

5.4.2. Comparison against recent Instance Selection

As mentioned in Section 4.2, two recent instance selection algorithms LSBo [27] and RIS1 [10] have been selected for comparison with our proposal. This section reports the experimental results of the these two algorithms against SPMS-ALS over 57 datasets with reference to test performance and reduction rate.

Details of the classification performance of RIS1, LSBo and SPMS-ALS in the test phase on small and medium datasets can be found in the Supple-

mentary Material, while the summary information is displayed at Table 10. Overall, RIS1 obtains a majority of wins in both small and medium datasets, SPMS-ALS ranks second and LSBo lies at the lowest position. Particularly, RIS1 has 25 wins (18 small and 7 medium), SPMS-ALS achieves the best results 18 times (15 small and 3 medium), while LSBo obtains 15 wins (8 small and 7 medium). However, the average test performance of LSBo and SPMS-ALS are the highest for small and medium datasets, respectively.

The overall goal of instance reduction is to reduce the original dataset as much as possible whilst keeping (or improving) the accuracy. Therefore, to establish a fairer comparison between the two instance selection methods and our method, we will also provide an additional metric to consider both test accuracy and reduction as equally important. Following [52], we simply multiply the accuracy in test **Acc** and the reduction rate **Red** to form a new metric **Acc*Rec**. Table 10 presents the overall average performance in accuracy **Acc**, reduction rate **Red** and both metrics **Acc*Rec**. Furthermore, Table 11 provides the set of rankings and p-values obtained from Friedman+Holm tests for the three contrasted algorithms.

In the previous section, all algorithms (PSO, SFLSDE, LSHADE, and SPMS-ALS) yield the same reduction rate. In particular, in this experiment, SPMS-ALS fixes the rate up to 95% and 98% for small and mediums datasets, respectively. However, LSBo and RIS1 do not specify the reduction rate as a parameter, but their reduction depends on a particular dataset. RIS1 yields an average reduction rate of 59.06% and 70.52% in small and medium datasets, respectively; whilst LSBo reduces 78.59% and 87.39% in small and medium datasets. Thus, both algorithms achieve a smaller reduction rate than the instance generation approach investigated in this paper.

Table 10: Summary of the performance of SPMS-ALS against RIS1 and LSBo considering **Acc** in the test phase, **Red** and **Acc*Red** measures for instance reduction over 57 datasets. The best performance in the column is shown in bold.

	Algorithm	Acc (Test)	Std	Red	Acc*Red
SMALL	RIS1	0.7319	0.0583	0.5906	0.4499
	LSBo	**0.7605**	0.0576	0.7859	0.6064
	SPMS-ALS	0.7549	0.0615	**0.9500**	**0.7172**
MEDIUM	RIS1	0.7972	0.0141	0.7052	0.6071
	LSBo	0.8540	0.0099	0.8739	0.7639
	SPMS-ALS	**0.8668**	0.0110	**0.9800**	**0.8495**

On average, the test performance of LSBo is the highest on small datasets,

while SPMS-ALS reports the highest average on medium datasets. However, apart from having the best reduction rate, SPMS-ALS obtains the best balance between accuracy and reduction. The results from the non-parametric tests (Friedman+Holm) in Table 11 reveal the advantage of SPMS-ALS looking at the Ranking and p-value columns. The p-values reported for the comparison in terms of accuracy column do not reflect any significant differences between the three methods in either small or medium datasets. However, when the reduction rate is taken into consideration the proposed technique stands out significantly.

Table 11: Friedman+Holm statistical test results in both **Acc** and **Acc*Rec** metrics for small and medium datasets. The best performance in the column is shown in bold.

	Acc			Acc * Red		
	Algorithm	Ranking	p-value	Algorithm	Ranking	p-value
SMALL	SPMS-ALS	**1.912**	–	SPMS-ALS	**1.175**	–
	RIS1	1.938	0.9110	LSBo	2.000	**0**
	LSBo	2.150	0.2882	RIS1	2.825	**2.25E-04**
MEDIUM	LSBo	**1.882**	–	SPMS-ALS	**1.176**	–
	RIS1	2.000	0.7316	LSBo	2.059	**1.01E-02**
	SPMS-ALS	2.117	0.4927	RIS1	2.765	**4.00E-06**

5.5. Hybridisation with Instance Selection

As stated in Section 4.2, to perform a fair comparison against hybrid instance reduction method SSMA-SFLSDE [52], the initialisation process of the proposed algorithm must be replaced with a smarter approach. In particular, we use the same instance selection algorithm, SSMA [18], as tested in [52]. This section compares LSIR, SFLSDE, LSHADE and SPMS-ALS after using SSMA as initialisation. The resulting algorithms are indicated as SSMA-LSIR, SSMA-SPMS-ALS, SSMA-LSHADE and the state-of-the-art algorithm SSMA-SFLSDE [52]. The detailed accuracy results for small and medium datasets in both training and test can be found in the Supplementary Material. Table 12 shows the average results obtained from the compared algorithms in conjunction with SSMA and the ranking plus p-values from the Friedman + Holm test.

The detailed results show that for small datasets and in training phase SSMA-SPMS-ALS outperforms SSMA-SFLSDE and SSMA-LSHADE, and is outperformed by SSMA-LSIR. However these results are not confirmed in test phase where SSMA-LSHADE achieves the best performance, SSMA-SPMS-ALS the third best performance after SSMA-SFLSDE, and SSMA-LSIR the worst performance over the four algorithms considered in this section. This

ranking is statistically significant and confirmed by the Friedman + Holm test. According to our interpretation, the deterministic and exploitative local search logic in LSIR causes overfitting. The restriction of the search space caused by SSMA increases the risk of overfitting. In the proposed memetic framework, the resampling and crossover mechanism seems to mitigate the overfitting.

Our interpretation is confirmed by the results for medium datasets. Since the search space is naturally large, the exploitative local search is beneficial [8] and is improved by the memetic framework. Hence, SSMA-SPMS-ALS achieves the best performance in training phase. This ranking is confirmed in test phase where SSMA-SPMS-ALS slightly outperforms SSMA-SFLSDE and is established as the control algorithm in the Friedman test.

In summary, and similar to what we saw when comparing against purely instance generation methods, the proposed memetic framework can obtain a very competitive classification performance, especially in larger datasets, whilst reducing drastically the required runtime.

Table 12: Summary performance between four hybrid models over 57 datasets.

		TRAINING		TEST		Friedman + Holm	
	Algorithm	Acc	Std	Acc	Std	Ranking	p_{Holm}
SMALL	SSMA-LSHADE	0.8687	0.0101	**0.7792**	0.0570	**2.000**	–
	SSMA-SFLSDE	0.8684	0.0108	0.7767	0.0594	2.200	0.4884
	SSMA-SPMS-ALS	0.8727	0.0134	0.7670	0.0574	2.700	0.0306
	SSMA-LSIR	**0.8911**	0.0148	0.7642	0.0604	3.100	0.0004
MEDIUM	SSMA-SPMS-ALS	**0.9264**	0.0033	0.8700	0.0107	**2.265**	–
	SSMA-SFLSDE	0.9059	0.0040	0.8675	0.0125	2.441	1.0000
	SSMA-LSHADE	0.9069	0.0035	**0.8706**	0.0118	2.647	1.0000
	SSMA-LSIR	0.9245	0.0039	0.8682	0.0127	2.647	1.0000

At last we report some considerations about future improvements that can be applied. We will investigate the extension of our approach to big data frameworks [55]. In addition, we plan to expand our approach to new promising classifiers, such as Heterogeneous oblique random forest [24]. An initial comparison with this kind of classifier can be found in the Supplementary Material. Further investigation is however required to perform an appropriate instance reduction for those classifiers.

5.6. Contextualising the results and limitations of the proposal

Experimental results in Sections 5.1 to 5.5, show that the proposed SPMS-ALS and its hybrid form, SSMA-SPMS-ALS, are effective at reducing the size

of the training data whilst maintaining, or even improving, the performance of the base classifier; in our case, the 1NN rule. The goal of this section is to contextualise the classification results presented in this paper with the 1NN as base classifier, and let the reader know where we are going in our future research. To do so, we compare the results of the proposed SPMS-ALS and 1NN against the popular Random Forest (RaF) algorithm and a state-of-the-art classifier also based on Trees, obRaF(H) [24].

It is important to note that the classification performance of a classifier is not only influenced by the pre-processing techniques but also its inherent robustness. For example, an ensemble classifier is likely to outperform a single model [46], or tree-based approaches handle categorical attributes better than the NN classifier. Thus, whilst the reader may expect the results of RaF and obRaF(H) to be superior to the ones presented with the 1NN rule, we believe it is beneficial to still observe the performance gap and understand potential future research lines for preprocessing technique for more robust classifiers.

To establish a fair comparison against methods, we have re-run SPMS-ALS over the 121 UCI datasets used in [16, 24], following the exact same experimental framework, including depth of a tree (i.e. 57), number of trees (i.e. 500), and a 4-fold cross validation scheme. Details of the comparison on each single dataset can be found in the Supplementary Material, while the summary information and results of the statistical test are displayed in Table 13. As expected, both RaF and obRaF(H) display a higher performance than NN-based results. On the other hand, the proposed data reduction does not only reduce the storage need but also becomes much more efficient in terms of runtime as we only preserve 2 to 5% of the training data. For this reason, it is essential to highlight that the contribution of our proposal lies in the reduction of the training set, as a preprocessing technique, whilst maintaining (or improving) the classification performance of 1NN.

Table 13: Summary the average performance of 4-fold cross validation between the basic models (1NN and RaF) and their improved versions (SPMS-ALS and obRaF(H)) over 121 datasets.

			Friedman + Holm	
	TRAINING	TEST	Ranking	p_{Holm}
obRaF(H)	–	**0.8336**	146.24	–
RaF(Scikit-learn)	0.9892	0.8286	173.13	0.05
SPMS-ALS	0.8926	0.7597	324.99	0.03
1NN	0.7487	0.7534	325.64	0.02

The reader might wonder if the result of the preprocessing performed by the technique proposed in this paper could be used directly by any other classifier like RaF or obRaF(B). Although this pre-processing approach is intended for 1NN, as highlighted in [5] the resulting set could potentially be used by any other classifiers. However, it is not straightforward to directly use the reduced set obtained from SPMS-ALS in another classifier. We have performed some preliminary experiments using the resulting reduced set as training data for RaF in 89 small datasets (from the set of 121). The average results in the test phase sets at 0.5656, whilst it is 1.000 on training. This suggests that RaF is overfitting the training data with the parameters used (e.g. depth of the trees, or number of trees). This could be expected as the reduction on small datasets may end up having as few as 2-15 samples in some extremely small datasets. Whilst NN technique would work well with such amount of data, Tree-like technique will not. Thus, as future work we plan to explore the interaction between the proposed SPMS-ALS and more robust classifiers like obRaF(B), for example, by adding it as base classifier or fine tuning the parameters to use smaller training datasets without overfitting.

6. Conclusion

This paper proposes a single-point MC approach for instance reduction. The proposed algorithm is composed of a novel accelerated local search and a crossover based global search. The local search is deterministic and exploitative belonging to the family of Pattern Search methods whilst the global search is stochastic, based on resampling and crossover. By making some considerations about the functioning of the NN classifier in instance generation and exploiting the search logic of Pattern Search, the local search has been redesigned and implemented in an accelerated version. The accelerated local search uses most of the calculations performed at the previous step and thus lead to a major saving in terms of runtime with respect to the existing algorithms in the literature.

Numerical results performed with and without instance selection as initialisation mechanism show that the proposed MC approach tends to be slightly worse than only one instance reduction algorithm in small datasets. On the other hand, on medium datasets, the proposed MC approach achieves the best accuracy performance in both training and test phases. These results are extremely valuable when we consider that the proposed approach is up to seven times faster than the other algorithms.

Besides the proposed domain-specific MC approach this article offers an extra contribution about experimentalism in data reduction. More specifically, in this paper we perform a thorough parameter setting of the computational budget and display the results in multiple scenarios. These results aim to offer some guidelines to data scientists to set their experimental conditions in a fair and effective manner to detect the desired trade-off between accuracy and runtime.

7. 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. All experiments in this study were conducted with the High Performance Computing system at the University of Nottingham. We thank George Darmiton da Cunha Cavalcanti and Rodolfo Jose de Oliveira Soares, the authors of paper [10] for the provided code and their support to run the experiment.

References

[1] G. Acampora, V. Loia, M. Gaeta, Exploring e-Learning Knowledge Through Ontological Memetic Agents, IEEE Computational Intelligence Magazine 5 (2) (2010) 66–77.

[2] E. Alpaydin, Introduction to Machine Learning, MIT press, 2014.

[3] J. E. Amaya, C. Cotta, A. J. Fernández, P. García-Sánchez, Deep memetic Models for Combinatorial Optimization Problems: Application to the Tool Switching Problem, Memetic Computing 12 (1) (2020) 3–22.

[4] 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, 1021–1031, 2004.

[5] J. R. Cano, F. Herrera, M. Lozano, Using Evolutionary Algorithms as Instance Selection for Data reduction in KDD: an experimental study, IEEE Transactions on Evolutionary Computation 7 (6) (2003) 561–575.

[6] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, M. Sumner, A Fast Adaptive Memetic Algorithm for On-line and Off-line Control Design of PMSM Drives, IEEE Transactions on System Man and Cybernetics-part B, Special Issue on Memetic Algorithms 37 (1) (2007) 28–41.

[7] F. Caraffini, F. Neri, M. G. Epitropakis, HyperSPAM: A Study on Hyper-heuristic Coordination Strategies in the Continuous Domain, Information Sciences 477 (2019) 186–202.

[8] F. Caraffini, F. Neri, G. Iacca, Large Scale Problems in Practice: The Effect of Dimensionality on the Interaction Among Variables, in: G. Squillero, K. Sim (Eds.), Applications of Evolutionary Computation, Springer, 636–652, 2017.

[9] F. Caraffini, F. Neri, B. Passow, G. Iacca, Re-sampled Inheritance Search: High Performance Despite the Simplicity, Soft Computing 17 (12) (2014) 2235–2256.

[10] G. D. Cavalcanti, R. J. Soares, Ranking-based Instance Selection for Pattern Classification, Expert Systems with Applications 150 (2020) 113269.

[11] Chang-Yong Lee, Xin Yao, Evolutionary Programming Using Mutations Based on the Levy Probability Distribution, IEEE Transactions on Evolutionary Computation 8 (1) (2004) 1–13.

[12] X. Chen, Y. Ong, M. Lim, K. C. Tan, A Multi-Facet Survey on Memetic Computation, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 591–607.

[13] T. M. Cover, P. E. Hart, Nearest Neighbor Pattern Classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.

[14] A. Elola, J. Del Ser, M. N. Bilbao, C. Perfecto, E. Alexandre, S. Salcedo-Sanz, Hybridizing Cartesian Genetic Programming and Harmony Search for Adaptive Feature Construction in Supervised Learning Problems, Applied Soft Computing 52 (2017) 760–770.

[15] L. Feng, Y. Ong, M. Lim, I. W. Tsang, Memetic Search With Interdomain Learning: A Realization Between CVRP and CARP, IEEE Transactions on Evolutionary Computation 19 (5) (2015) 644–658, ISSN 1941-0026.

[16] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?, The Journal of Machine Learning Research 15 (1) (2014) 3133–3181.

[17] I. Fister, A. Iglesias, A. Gálvez, J. Del Ser, E. Osaba, I. F. Jr., M. Perc, M. Slavinec, Novelty Search for global Optimization, Applied Mathematics and Computation 347 (2019) 865–881.

[18] S. García, J. R. Cano, F. Herrera, A Memetic Algorithm for Evolutionary Prototype Selection: A Scaling up Approach, Pattern Recognition 41 (8) (2008) 2693–2709.

[19] S. García, J. Derrac, J. Cano, F. Herrera, Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435.

[20] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power, Information Sciences 180 (10) (2010) 2044–2064.

[21] A. Gupta, Y.-S. Ong, Memetic Computation, Adaptation, Learning, and Optimization, Springer, 2019.

[22] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, M. H. Lim, Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration, Information Sciences 188 (2012) 17–43.

[23] N. D. Jana, J. Sil, S. Das, Continuous Fitness Landscape Analysis Using a Chaos-Based Random Walk Algorithm, Soft Computing 22 (2018) 921–948.

[24] R. Katuwal, P. N. Suganthan, L. Zhang, Heterogeneous Oblique Random Forest, Pattern Recognition 99 (2020) 107078.

[25] H. L. Le, D. Landa-Silva, M. Galar, S. Garcia, I. Triguero, EUSC: A Clustering-based Surrogate Model to Accelerate Evolutionary Undersampling in Imbalanced Classification, Applied Soft Computing 101 (2021) 107033.

[26] M. N. Le, Y. S. Ong, Y. Jin, B. Sendhoff, Lamarckian Memetic Algorithms: Local Optimum and Connectivity Structure Analysis, Memetic Computing Journal 1 (3) (2009) 175–190.

[27] E. Leyva, A. González, R. Pérez, Three New Instance Selection Methods Based on Local Sets: A Comparative Study with Several Approaches from a Bi-objective Perspective, Pattern Recognition 48 (4) (2015) 1523–1537.

[28] X. Li, X. Yao, Cooperatively Coevolving Particle Swarms for Large Scale Optimization, Evolutionary Computation, IEEE Transactions on 16 (2) (2012) 210–224.

[29] P. López-García, E. Onieva, E. Osaba, A. D. Masegosa, A. Perallos, GACE: A Meta-heuristic Based in the Hybridization of Genetic Algorithms and Cross Entropy Methods for Continuous Optimization, Expert Systems with Applications 55 (2016) 508–519.

[30] L. Ma, J. Li, Q. Lin, M. Gong, C. A. Coello Coello, Z. Ming, Cost-Aware Robust Control of Signed Networks by Using a Memetic Algorithm, IEEE Transactions on Cybernetics (2020) 1–14To appear.

[31] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, Z. Zhu, A Survey on Cooperative Co-Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation 23 (3) (2019) 421–441.

[32] K. M. Malan, A. P. Engelbrecht, A Survey of Techniques for Characterising Fitness Landscapes and Some Possible Ways Forward, Information Sciences 241 (2013) 148 – 163.

[33] A. D. Martinez, E. Osaba, I. Oregi, I. F. Jr., I. Fister, J. Del Ser, Hybridizing differential evolution and novelty Search for multimodal Optimization Problems, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019, 1980–1989, 2019.

[34] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Tech. Rep. 826, Caltech, 1989.

[35] P. Moscato, M. Norman, A Competitive and Cooperative Approach to Complex Combinatorial Search, Tech. Rep. 790, Caltech, 1989.

[36] L. Nanni, A. Lumini, Particle Swarm Optimization for Prototype Reduction, NeuroComputing 72 (4-6) (2009) 1092–1097.

[37] F. Neri, Linear Algebra for Computational Sciences and Engineering, Springer, second edn., 2019.

[38] F. Neri, C. Cotta, Memetic Algorithms and Memetic Computing Optimization: A Literature Review, Swarm and Evolutionary Computation 2 (2012) 1–14.

[39] F. Neri, G. Iacca, E. Mininno, Disturbed Exploitation Compact Differential Evolution for Limited Memory Optimization Problems, Information Sciences 181 (12) (2011) 2469 – 2487.

[40] F. Neri, S. Rostami, Generalised Pattern Search Based on Covariance Matrix Diagonalisation, SN Comput. Sci. 2 (3) (2021) 171.

[41] F. Neri, I. Triguero, A Local Search with a Surrogate Assisted Option for Instance Reduction, in: P. A. Castillo, J. L. J. Laredo, F. F. de Vega (Eds.), Applications of Evolutionary Computation, vol. 12104 of *Lecture Notes in Computer Science*, Springer, 578–594, 2020.

[42] Q. H. Nguyen, Y. S. Ong, L. M. Hiot, N. Krasnogor, Adaptive Cellular Memetic Algorithms, Evolutionary Computation 17 (2) (2009) 231–256.

[43] R. Nogueras, C. Cotta, Studying Self-balancing Strategies in Island-based Multimemetic Algorithms, Journal of Computational and Applied Mathematics 293 (2016) 180–191.

[44] E. Özcan, B. Bilgin, E. E. Korkmaz, A Comprehensive Analysis of Hyper-heuristics, Intelligent Data Analysis 12 (1) (2008) 3–23.

[45] P. Refaeilzadeh, L. Tang, H. Liu, Cross-Validation, in: L. LIU, M. T. ÖZSU (Eds.), Encyclopedia of Database Systems, Springer US, Boston, MA, 532–538, 2009.

[46] L. Rokach, Ensemble-based Classifiers, Artificial Intelligence Review 33 (1) (2010) 1–39.

[47] R. Ros, N. Hansen, A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity, in: G. Rudolph, T. Jansen, N. Beume, S. Lucas, C. Poloni (Eds.), Parallel Problem Solving from Nature – PPSN X, Springer Berlin Heidelberg, Berlin, Heidelberg, 296–305, 2008.

[48] R. Tanabe, A. Fukunaga, Success-history Based Parameter Adaptation for Differential Evolution, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, 71–78, 2013.

[49] R. Tanabe, A. S. Fukunaga, Improving the Search Performance of SHADE Using Linear Population Size Reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 1658–1665, 2014.

[50] C. Ting, R. Liaw, T. Wang, T. Hong, Mining Fuzzy Association Rules Using a Memetic Algorithm Based on Structure Representation, Memetic Computing 10 (1) (2018) 15–28.

[51] I. Triguero, J. Derrac, S. García, F. Herrera, A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification, IEEE Transactions on Systems, Man, and Cybernetics–Part C 42 (1) (2012) 86–100.

[52] I. Triguero, S. García, F. Herrera, Differential Evolution for Optimizing the Positioning of Prototypes in Nearest Neighbor Classification, Pattern Recognition 44 (4) (2011) 901–916.

[53] 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.

[54] I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesús, L. Sánchez, F. Herrera, KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining, International Journal of Computational Intelligence Systems 10 (2017) 1238–1249, ISSN 1875-6883.

[55] 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, ISSN 0925-2312.

[56] L.-Y. Tseng, C. Chen, Multiple trajectory Search for Large Scale Global Optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 3052–3059, 2008.

[57] F. Wilcoxon, Individual Comparisons by Ranking Methods, Biometrics Bulletin 1 (6) (1945) 80–83.

[58] D. R. Wilson, T. R. Martinez, Reduction Techniques for Instance-Based Learning Algorithms, Machine Learning 38 (3) (2000) 257–286.

[59] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Elsevier, 2017.

[60] X. Yao, Y. Liu, G. Lin, Evolutionary Programming Made Faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.

[61] A. A. Zaher, R. Berretta, N. Noman, P. Moscato, An Adaptive Memetic Algorithm for Feature Selection Using Proximity Graphs, Computational Intelligence 35 (1) (2019) 156–183.

[62] J. Zhang, A. C. Sanderson, JADE: Adaptive Differential Evolution with Optional External Archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.

[63] Z. Zhu, S. Jia, Z. Ji, Towards a Memetic Feature Selection Paradigm [Application Notes], IEEE Computational Intelligence Magazine 5 (2) (2010) 41–53.