A Local Search with a Surrogate Assisted Option for Instance Reduction

Ferrante Neri[0000−0002−6100−6532] and Isaac Triguero[0000−0002−0150−0651]

Computational Optimisation and Learning (COL) Lab,
School of Computer Science, University of Nottingham, United Kingdom
{ferrante.neri,isaac.triguero}@nottingham.ac.uk

Abstract. In data mining, instance reduction is a key data pre-processing step that simplifies and cleans raw data, by either selecting or creating new samples, before applying a learning algorithm. This usually yields to a complex large scale and computationally expensive optimisation problem which has been typically tackled by sophisticated population-based metaheuristics. Unlike the recent literature, in order to accomplish this target, this article proposes the use of a simple local search algorithm and its integration with an optional surrogate assisted model. This local search, in accordance with variable decomposition techniques for large scale problems, perturbs an n-dimensional vector along the directions identified by its design variables one by one.
Empirical results in 40 small data sets show that, despite its simplicity, the proposed baseline local search on its own is competitive with more complex algorithms representing the state-of-the-art for instance reduction in classification problems. The use of the proposed local surrogate model enables a reduction of the computationally expensive objective function calls with accuracy test results overall comparable with respect to its baseline counterpart.

Keywords: Instance Reduction · Instance Generation · Computationally Expensive Problems · Surrogate Assisted Algorithms · Local Search · Pattern Search.

1 Introduction

Data science is a discipline that studies methods to store and manage data with the aim of extracting knowledge from it [6]. A typical problem in data science is to have a very large raw data set which requires pre-processing to enable data mining techniques to learn from a more manageable data set that is free of noise, redundant or irrelevant samples. In order to overcome this issue, a normal practice consists of selecting some instances and discarding others, or creating artificial samples that better represent the original training data.

However, it is fundamental to properly select or generate those instances. *Instance reduction* techniques, either selection [7] or generation [28], have to allow still to extract the required knowledge. In other words, we would like to

simplify the original data set and keep it as informative as it is when it contains all the data, or even better if noisy data is removed appropriately [14].

Instance reduction can be formulated as an optimisation problem and be addressed by search algorithms. The pure selection of instances can be seen as a binary space search problem [1]. The generation of new representative instances, however, can be expressed as a continuous space search problem. The latter approach turned out to be more flexible, but also more complex [30]. In both cases, Evolutionary Algorithms (EAs) have excelled in comparison with other approaches [7, 28]. EAs for instance generation are based on optimising the location of a subset of instances [18, 30].

Note that most instance reduction algorithm were originally designed to enhance the performance of the Nearest Neighbour classifier (NN) [5], but the resulting pre-processed data set could be used, in principle, by any classifier [1]. In this work, we are focused on instance generation for NN classification, also known as prototype generation.

Two major challenges are associated to the instance reduction problem: the high dimensionality of the problem and the high cost of each objective function evaluation, which typically consists of classifying the training data. The first challenge is addressed by using an exploitative operator which can be embedded within heuristic frameworks. Some examples under the umbrella name of Memetic Algorithms are proposed in [8, 9]. A comparison reporting the advantages of the extra local search is reported in [20]. In the recent literature, these problems are currently being addressed by using distributed approaches in big data platforms [33], but population-based approaches keep taking a long time to pre-process the data. Thus, there is a need for simpler and faster, yet powerful, search algorithms.

This article also explicitly addresses the second challenge by proposing a technique to limit the cost of instance reduction within the optimisation process. More specifically, this article proposes the use of a local search algorithm for large scale problems and a surrogate (approximated) local model to reduce the number of objective function calls. To the best of our knowledge, this is the first local search proposed for instance generation, and the use of surrogate models has been often neglected. The proposed local search samples the points in its neighbourhood and makes use of them to build a multi-variable (local) linear model. The resulting surrogate assisted local search [11, 26, 25, 22] alternates the use of the true objective function with the approximation given by the surrogate model. A mechanism to ensure that wrong search directions are suggested by the surrogate model has been implemented: the algorithm checks the promising points provided by the surrogate model before accepting a new base point.

The remainder of this article is organised in the following way. Section 2 describes the instance reduction 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. Details about the implementation and linear regression

model are also included. Section 4 displays the algorithmic results. Finally, Section 5 provides the conclusion of this study.

2 Problem Formulation

Let **TR** be a training data set and **TS** a test set for a supervised classification problem. Both data sets can be viewed as a matrix whose rows are the instances and columns are the features:

$$\mathbf{TR} = \begin{pmatrix} & \mathbf{F_1} & \mathbf{F_2} & \dots & \mathbf{F_m} \\ \mathbf{I_1} & a_{11} & a_{12} & \dots & a_{1m} \\ \mathbf{I_2} & a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbf{I_l} & a_{l1} & a_{l2} & \dots & a_{lm} \end{pmatrix}$$

Each instance belongs to a class ω. For the **TR** set the class ω is known, while it is unknown for **TS**. The objective of an instance reduction algorithm is to provide a reduced set **RS** of instances, which are either selected or generated from the examples of **TR**,

$$\mathbf{RS} = \begin{pmatrix} & \mathbf{F_1} & \mathbf{F_2} & \dots & \mathbf{F_m} \\ \mathbf{I_1} & b_{11} & b_{12} & \dots & b_{1m} \\ \mathbf{I_2} & b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbf{I_i} & b_{i1} & b_{i2} & \dots & b_{im} \end{pmatrix}$$

with $i \ll l$ that still allows the data representation of **TR**. **RS** should be created to efficiently represent the distributions of the classes. The size of RS should be significantly reduced to minimise the information that requires storing, and speed up the posterior classification phase.

We may, equivalently, represent the matrix **RS** as a vector $\mathbf{x}$ of length $n = i \times m$ whose elements are the rows of **TR** arranged sequentially

$$\mathbf{x} = (b_{11}, b_{12}, \dots, b_{1m}, b_{21}, b_{22}, \dots, b_{2m}, \dots, b_{i1}, b_{i2}, \dots, b_{im}) = (x_1, x_2, \dots, x_n)$$

The objective function $f(\mathbf{x})$ will measure how well the resulting RS exemplifies the original training data TR. To do so, in the literature, RS is inferred using the TR matrix as representative information of the problem, assuming that this will allow us to classify the elements of TS. In particular, this objective function simply calculates the classification accuracy (i.e. number of correct classifications regarding the total number of instances classified) using RS as training data, and TR as test data.

2.1 Computational Cost of the Objective Function

The exact computational cost of the objective function depends on the particular classifier that is being used. Most of the instance reduction literature focused

their efforts on improving the well-known NN classifier, because it is one of the most affected classifiers by the size of the training data.

Focusing on the NN rule as base classifier, calculating the accuracy of RS consists of computing the Euclidean distance between all elements of TR against all elements of RS and determine which is the closest instance in RS for each element of TR. The class label of the closest instance is used as prediction.

This intuitively shows that the cost of the objective function will be very high when the size of TR is very big. The complexity of instance reduction models is $O((i \cdot m)^2)$ or higher, and best performing methods are based on EAs [30].

Current research is typically focused on the use of divide-and-conquer approaches, implemented with big data technologies, to parallelise the execution of instance reduction approaches. We can also find an approximation strategy, called windowing [31], which estimates the fitness value of RS using a random subset of TR at every iteration of the search (this reduces significantly the cost, but could mislead the search). However, the use of more sophisticated surrogate models to reduce the number of evaluations for instance reduction algorithms has been neglected.

3 A Local Search for Instance Reduction

This section presents the proposed method, outlines its theoretical and implementation aspects and justifies the choices made. More specifically, Subsection 3.1 presents the structure of the baseline Local Search, Subsection 3.2 describes the multivariable linear model used in this study, Subsection 3.3 outlines the surrogate assisted technique to build and use the surrogate model with the original objective function, and finally, Subsection 3.4 provides a justification of the algorithmic choices made.

3.1 Baseline Local Search

The proposed algorithm is based on a greedy local search [13, 21] of the family of Pattern Search algorithms [27]. The algorithm perturbs each variable (of $\mathbf{x}$) at the time and replaces the current best point with a better one as soon as an improved solution is found. Along the directions identified by each variable, the algorithm attempts to move one step in one oriented direction and then half step in the opposite oriented direction if the first attempt fails. More specifically, the algorithm explores at first

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

where the scalar ρ is the step-size (exploratory radius) defined by the user and $\mathbf{e^i}$ is the i^{th} versor, i.e. a vector composed of zeros and only a one in the i^{th} position. Then if this exploration fails, the algorithm attempts to explore

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

Algorithm 1 Baseline Local Search used for Instance Reduction (LSIR)

INPUT x
while local budget condition **do**
  $\mathbf{x^t} = \mathbf{x}$
  {**Exploration**}
  for $i = 1 : n$ **do**
    $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$
    if $f(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
      $\mathbf{x} = \mathbf{x^t}$
    else
      $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$
      if $f(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
        $\mathbf{x} = \mathbf{x^t}$
      end if
    end if
  end for
  if x has **not** been updated **then**
    $\rho = \frac{\rho}{2}$
  end if
end while
RETURN x

Algorithm 1 shows the pseudocode of the baseline Local Search for Instance Reduction (LSIR) used in this study.

For the experiments carried out in this paper, 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

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

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

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

$\forall i$. The parentheses $\lfloor \rfloor$ indicate the truncation to the lower integer.

3.2 Linear Multivariable Surrogate Model

In order to approximate the objective function f and generate a surrogate function $\tilde{f}$, a multivariable linear regression with least square method is implemented, see [10, 12]. For the sake of clarity, we built a local surrogate linear model

$$\tilde{f}(\mathbf{x}) = c_0 + c_1 x_1 + c_2 x_2 + c_n x_n = \sum_{j=1}^{n} c_j x_j + c_0.$$

In order to identify the $n + 1$ parameters $c_0, c_1, c_2, \ldots c_n$ the least square method has been applied.

The method processes a sample of $n + 1$ observation vectors

$$\mathbf{x^1}, \mathbf{x^2}, \ldots \mathbf{x^{n+1}}$$

where

$$\mathbf{x^j} = (x_{j1}, x_{j1}, \ldots, x_{jn})$$

and the corresponding function values

$$\begin{array}{l} y_1 = f\left(\mathbf{x^1}\right) \\ y_2 = f\left(\mathbf{x^2}\right) \\ \ldots \\ y_j = f\left(\mathbf{x^j}\right) \\ \ldots \end{array}$$

In order to find the parameters $c_0, c_1, c_2, \ldots c_n$ we have to minimise the following function Δ

$$\Delta = \sum_{j=1}^{n+1} \left(y_j - \left(c_0 + \sum_{i=1}^{n} c_i x_{ji} \right) \right)^2 .$$

Thus, we have to calculate the partial derivatives of Δ with respect to $c_0, c_1, \ldots c_n$. The derivative with respect to c_0 and c_1 are, respectively

$$\frac{\partial \Delta}{\partial c_0} = -2 \left(\sum_{j=1}^{n+1} y_j - \left(c_0 (n+1) + c_1 \sum_{j=1}^{n+1} x_{j1} + \ldots + c_n \sum_{j=1}^{n+1} x_{jn} \right) \right)$$

$$\frac{\partial \Delta}{\partial c_1} = -2 \left(\sum_{j=1}^{n+1} x_{j1} y_j - \left(c_0 \left(\sum_{j=1}^{n+1} x_{j1} \right) + \ldots + c_1 \left(\sum_{j=1}^{n+1} x_{j1}^2 \right) + \ldots + c_n \left(\sum_{j=1}^{n+1} x_{j1} x_{jn} \right) \right) \right)$$

The derivative with respect to the generic coefficient c_i is

$$\frac{\partial \Delta}{\partial c_i} = -2 \left(\sum_{j=1}^{n+1} x_{ji} y_j - \left(c_0 \left(\sum_{j=1}^{n+1} x_{ji} \right) + \ldots + c_i \left(\sum_{j=1}^{n+1} x_{ji}^2 \right) + \ldots + c_k \left(\sum_{j=1}^{n+1} x_{ji} x_{jk} \right) + \ldots \right) \right) .$$

By simultaneously equating the derivatives to 0, we obtain the system of linear equations $\mathbf{Lc} = \hat{\mathbf{y}}$, that is

$$\begin{pmatrix} (n+1) & \sum_{j=1}^{n+1} x_{j1} & \sum_{j=1}^{n+1} x_{j2} & \ldots & \sum_{j=1}^{n+1} x_{jn} \\ \sum_{j=1}^{n+1} x_{j1} & \sum_{j=1}^{n+1} x_{j1}^2 & \sum_{j=1}^{n+1} x_{j1} x_{j2} & \ldots & \sum_{j=1}^{n+1} x_{j1} x_{jn} \\ \sum_{j=1}^{n+1} x_{j2} & \sum_{j=1}^{n+1} x_{j2} x_{j1} & \sum_{j=1}^{n+1} x_{j2}^2 & \ldots & \sum_{j=1}^{n+1} x_{j2} x_{jn} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \sum_{j=1}^{n+1} x_{jn} & \sum_{j=1}^{n+1} x_{jn} x_{j1} & \sum_{j=1}^{n+1} x_{jn} x_{j2} & \ldots & \sum_{j=1}^{n+1} x_{jn}^2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \ldots \\ c_n \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n+1} y_j \\ \sum_{j=1}^{n+1} x_{j1} y_j \\ \sum_{j=1}^{n+1} x_{j2} y_j \\ \ldots \\ \sum_{j=1}^{n+1} x_{jn} y_j \end{pmatrix} .$$

The solution of this system of linear equation is the set of parameters $\mathbf{c}$ which allow the construction of the surrogate model $\tilde{f}(\mathbf{x})$.

3.3 The Proposed Surrogate Local Search for Instance Reduction

With reference to Algorithm 1, each exploration in the **for** loop samples at least n and at most $2n$ trial points $\mathbf{x^t}$ in the neighbourhood of the current best point $\mathbf{x}$. The proposed Surrogate Assisted Local Search for Instance Reduction (SALSIR) exploits this logic by storing the visited points in a data structure $Surr$, that is a list where each entry is a point $\mathbf{x}$ and the corresponding $f(\mathbf{x})$:

$$Surr(k) = (\mathbf{x}, f(\mathbf{x})).$$

The data structure $Surr$ is filled until it contains n entries. Since the starting point is also inserted in $Surr$, $(n+1)$ points are available. These points are used to build a surrogate model $\tilde{f}(\mathbf{x})$.

For the remaining function calls, the LS uses the surrogate model $\tilde{f}(\mathbf{x})$ instead of the computationally expensive objective function $f(\mathbf{x})$. However, to ensure that wrongly estimated search directions do not jeopardise the functioning of the algorithm, when a solution estimated by the surrogate model outperforms the current best solution, its actual objective function value is checked. This increases the cost of the algorithm (reduces the advantages of the surrogate model [19]) . On the other hand, this strategy enhances the reliability of the search.

If the moves failed in all directions, the exploratory radius is halved and the search repeated in a closer neighbourhood of $\mathbf{x}$. The pseudocode of this algorithm is shown Algorithm 2. We highlighted that the main loop of the algorithm is divided into parts: in the first the surrogate model is built while in the second the surrogate model is used as an alternative to the objective function.

3.4 Motivation of the Proposed Design

This section justifies the algorithmic choices and in particular answers to the following two questions.

1. *Why did we choose this algorithmic structure for this problem?*
2. *Why did we choose a multivariable linear model as a surrogate?*

To address the first question, we have to consider that the optimisation problem under examination besides being computationally expensive is large scale. In data science, it is very likely to have a large volume of data and matrix **RS** above can easily have still hundreds if not thousands of rows.

For this reason, we selected a LS component that is especially suited for large problems as it is the main element of the algorithm proposed in [34] and then used as a LS in [36] and modified as a stand-alone LS within other frameworks, see e.g. [3, 4].

Techniques that perturb the variables separately, just like that used in this article, are known to be effective for large scale problems, see [24, 17, 15]. This observation was reported in the experimental study in [2]. 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 dimensionality grows, see [2].

This study is one of the reasons behind the decision of using a linear surrogate model (second question above).

Algorithm 2 The Proposed Surrogate Assisted Local Search for Instance Reduction (SALSIR) Algorithm

INPUT x
while local budget condition **do**
  $k = 1; i = 1$
  $Surr = []$ {**Initialise the surrogate list**}
  $\mathbf{x^t} = \mathbf{x}$
  {**Build the surrogate**}
  while $k \leq (n+1)$ **do**
    $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$
    $Surr(k) = (\mathbf{x^t}, f(\mathbf{x^t}))$
    $k = k + 1$
    if $f(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
      $\mathbf{x} = \mathbf{x^t}$
    else
      $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$
      $Surr(k) = (\mathbf{x^t}, f(\mathbf{x^t}))$
      $k = k + 1$
      if $f(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
        $\mathbf{x} = \mathbf{x^t}$
      end if
    end if
    $i = i + 1$
  end while
  Use $Surr$ to build the multivariable linear model $\tilde{f}(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j + c_0$
  {**Use the surrogate model to reduce the function calls**}
  for i=i:n **do**
    $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$
    if $\tilde{f}(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
      Calculate $f(\mathbf{x^t})$
      {**Ensure that the surrogate does not mislead the search**}
      if $f(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
        $\mathbf{x} = \mathbf{x^t}$
      end if
    else
      $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$
      if $\tilde{f}(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
        Calculate $f(\mathbf{x^t})$
        {**Ensure that the surrogate does not mislead the search**}
        if $f(\mathbf{x^t}) \leq f(\mathbf{x})$ **then**
          $\mathbf{x} = \mathbf{x^t}$
        end if
      end if
    end if
  end for
  if x has **not** been updated **then**
    $\rho = \frac{\rho}{2}$
  end if
end while
RETURN x

From the perspective of the interaction among variables, a complex model is unnecessary in the highly multivariate domain since both the objective functions and surrogate model (for the limited budget) would appear separable. The second reason is that, to our knowledge, there are no studies on the fitness landscape of the instance reduction problem. Although many studies propose many algorithms to achieve the reduction of instances, we do not know yet the features of the problem, e.g. how multimodal it is, and we do not know how the landscape depends on the specific data set. Hence, the simplistic approach of using a local linear model is a natural choice, see [22, 16]. In the present paper, we propose a local surrogate model that is designed to work in a limited portion of the decision space by using the neighbour points visited by the local search algorithm, see [11, 37, 23, 35].

4 Experimental study

This section describes the experimental setup and presents the numerical results of our study. Subsection 4.1 provides a description of the experimental framework while Subsection 4.2 displays, analyses, and interprets the results achieved against a number of algorithms for instance reduction previously proposed in the literature. Finally Subsection 4.3 analyses the benefits and drawbacks of SALSIR with respect to its baseline counterpart.

4.1 Experimental Framework

For the proposed study, in order to test the viability of the use of a local search for instance reduction, we have chosen 40 small data sets from the KEEL data set repository [32] with less than 2,000 instances (based on [7, 28]). Table 1 outlines the main features of these data sets. For each data set, the total number of examples (#Ex.), number of attributes (#Atts.), and number of classes (#ω.) are shown. These data sets are partitioned using a ten fold cross-validation scheme (10-fcv).For each data set, n can be computed as $n = 0.9 \times \#\text{Ex.} \times \#\text{Atts.}$

In order to evaluate the proposed methods, the following two measures have been used:

Classification accuracy:

$$Acc = \frac{N_{cc}}{N_I}$$

where N_{cc} is the number of correct classifications and N_I is the total number of instances. The classification is performed by the NN classifier using the resulting RS. The results in training ($TrainAcc$) and test ($TestAcc$) partitions are reported.

Reduction rate:

$$Red = 1 - \frac{size(\mathbf{RS})}{size(\mathbf{TR})}$$

where $size(RS)$ and $size(TR)$ are the sizes of reduced and training sets, respectively, that is the number of rows of the two matrices. This index measures the reduction of storage requirements achieved by an instance reduction algorithm.

Table 1: Brief description of the classification data sets used in this study

Data Set	#Ex.	#Atts.	$\#\omega$	Data Set	#Ex.	#Atts.	$\#\omega$
appendicitis	106	7	2	housevotes	435	16	2
australian	690	14	2	iris	150	4	3
autos	205	25	6	led7digit	500	7	10
balance	625	4	3	lymphography	148	18	4
bands	539	19	2	mammographic	961	5	2
breast	286	9	2	monks	432	6	2
bupa	345	6	2	movement_libras	360	90	15
car	1,728	6	4	newthyroid	215	5	3
cleveland	297	13	5	pima	768	8	2
contraceptive	1,473	9	3	saheart	462	9	2
crx	125	15	2	sonar	208	60	2
dermatology	366	33	6	spectheart	267	44	2
ecoli	336	7	8	tae	151	5	3
flare-solar	1,066	9	2	tic-tac-toe	958	9	2
german	1,000	20	2	vehicle	846	18	4
glass	214	9	7	vowel	990	13	11
haberman	306	3	2	wine	178	13	3
hayes-roth	133	4	3	wisconsin	683	9	2
heart	270	13	2	yeast	1484	8	10
hepatitis	155	19	2	zoo	101	17	7

Various instance reduction methods representing the state-of-the-art have been used for comparison with the proposed LSIR and SALSIR. In this study, We focused on the family of positioning adjustment methods (see [28]), which are the best performing instance reduction methods in the literature and follow a working logic similar to that of the proposed local search algorithms.

In order to compare the methods, we used as a benchmark the NN rule employing the entire TR set for training. In addition, we compared against the entire set of the positioning adjustment-based methods reviewed in [28]. We also included two advanced instance reduction algorithms: an incremental Differential Evolution (IPADE) [29], and a hybrid instance selection and instance generation algorithm (SSMA-SFLSDE) which is the current state-of-the-art according to [30]. Hence, 17 algorithms in total are considered in this study.

For the proposed LSIR method, and its surrogate variant, the search is started with a random subset of 5% of the rows of **TR** as suggested in [28]. Both LSIR and SALSIR are stopped either when $\rho < 10^{-5}$ or when $100 \times n$ objective function calls have been performed. Table 2 presents in greater detail the configuration parameters for IPADE, SSMA-SFLSDE and the proposed methods. Regarding the other comparison methods and related parameters we used the setup suggested in `https://sci2s.ugr.es/pr/pgtax/experimentation`.

Table 2: Parameters of the optimisation algorithms used for instance reduction.

Algorithm	*Parameters*
LSIR	Evaluations = $100 \times n$, $\rho = 0.4$ Reduction Rate = 0.95
SALSIR	Evaluations = $100 \times n$, $\rho = 0.4$ Reduction Rate = 0.95
IPADE	PopulationSize = 50, iterations of Basic DE = 500
	iterSFGSS =8, iterSFHC=20, Fl=0.1, Fu=0.9
SSMA-SFLSDE	PopulationSFLSDE= 40, IterationsSFLSDE = 500,
	iterSFGSS =8, iterSFHC=20, Fl=0.1, Fu=0.9
NN	Number of neighbors = 1, Euclidean distance.

4.2 Results

Table 3 provides the average results of reduction rate, training and test accuracy on the 40 data sets used in this paper. For each type of result, the algorithms are ranked from the best to the worst. The NN algorithm is highlighted in bold as the benchmark method.

Table 3: Average Results on 40 small data sets

Red		*TrainAcc*		*TestAcc*	
PSCSA	0.9858	LSIR	0.8667	SSMASFLSDE	0.7845
IPADE	0.9798	SSMASFLSDE	0.8651	PSO	0.7501
AVQ	0.9759	SALSIR	0.8410	IPADE	0.7446
LVQTC	0.9551	HYB	0.8309	LSIR	0.7415
SSMASFLSDE	0.9547	ENPC	0.8247	SALSIR	0.7346
MSE	0.9520	PSO	0.8238	**NN**	**0.7326**
LVQPRU	0.9503	IPADE	0.7883	MSE	0.7237
DSM	0.9491	MSE	0.7566	ENPC	0.7167
VQ	0.9491	**NN**	**0.7369**	HYB	0.7153
PSO	0.9491	LVQTC	0.7327	LVQPRU	0.6997
LSIR	0.9488	LVQPRU	0.7304	LVQTC	0.6981
SALSIR	0.9488	AMPSO	0.7227	AMPSO	0.6903
LVQ3	0.9488	DSM	0.7036	DSM	0.6810
AMPSO	0.9430	LVQ3	0.6931	LVQ3	0.6763
ENPC	0.7220	AVQ	0.6869	PSCSA	0.6682
HYB	0.4278	PSCSA	0.6787	AVQ	0.6672
NN	**0.0000**	VQ	0.6614	VQ	0.6549

Table 3 shows that the proposed LSIR and SALSIR achieve the best and the third best training accuracy result, and are ranked forth and fifth in terms of test accuracy. The reduction rates of LSIR and SALSIR are comparable with those of the other methods that use a reduction rate parameter of 5%. It must be remarked that despite the low number of evaluations and a simple local search strategy, the proposed LSIR algorithm provides the highest train accuracy. This indicates that LSIR may be incurring in overfitting of the training data sets. Particularly interesting is the comparison with PSO, which also starts off from the same random set of instances (5%). PSO does not seem to find an RS that fits that well the training data. This turns out to be in its favour as it reduces the overfitting of the training data, providing a higher test result. This may suggest that an even lower number of evaluations may prevent our algorithm from overfitting the data.

Table 4 presents the average test classification accuracy results (from the 10-fcv), for the proposed methods and the NN rule. The best result for each data set is highlighted in the bold face.

We can observe that for the majority of the datasets (29 out of 40), both proposed methods outperform the benchmark NN. This means that the methods are not only able to reduce the size the training data by 95%, but also are also able to improve the performance of the NN classifier. In the remaining cases the data reduction process may deteriorate the performance of the NN algorithm (e.g. on aut data set). This may be due to overfitting of the training data, or

Table 4: Classification accuracy in test of the proposed methods against the NN benchmark rule

Data Set	NN	LSIR	SALSIR	Data Set	NN	LSIR	SALSIR
appendicitis	0.7936	**0.8118**	0.8018	housevotes	**0.9216**	0.9149	0.9147
australian	0.8145	0.8261	**0.8464**	iris	0.9333	0.9467	**0.9733**
aut	**0.7474**	0.5434	0.49	led7digit	0.4020	**0.704**	0.688
bal	0.7904	**0.8718**	0.8463	lym	0.7387	0.7491	**0.755**
bands	0.6309	0.6811	**0.6848**	mammographic	0.7368	0.794	**0.8013**
bre	0.6535	**0.6896**	0.6789	monks	0.7791	**0.869**	0.8505
bupa	0.6108	**0.6452**	0.6143	movement_libras	0.8194	0.5778	0.5639
car	0.8565	0.**9143**	0.8762	newthyroid	**0.9723**	0.9632	0.9584
cleveland	0.5314	0.5483	**0.5644**	pima	**0.7033**	0.6903	0.6488
contraceptive	0.4277	**0.48**	0.4657	saheart	0.6449	**0.6926**	0.6687
crx	0.7957	**0.8362**	0.8217	sonar	**0.8555**	0.7776	0.706
dermatology	**0.9535**	0.9209	0.91	spectfheart	0.6970	**0.7943**	0.7862
ecoli	**0.8070**	0.7917	0.798	tae	0.4050	0.5188	**0.525**
flare-solar	0.5554	0.6473	**0.6594**	tic-tac-toe	0.7307	**0.7641**	0.7349
german	0.7050	**0.706**	0.677	vehicle	**0.7010**	0.6868	0.681
glass	0.7361	0.6071	**0.628**	vowel	**0.9939**	0.6253	0.5677
haberman	0.6697	0.686	**0.7384**	wine	**0.9552**	0.933	0.9333
hayes-roth	0.3570	0.4687	0.5701	wisconsin	0.9557	0.9313	**0.9571**
heart	0.7704	**0.8111**	0.7778	yeast	0.5047	**0.5552**	0.5546
hepatitis	**0.8075**	0.8054	0.7608	zoo	**0.9281**	0.8786	0.905

when the (random) original selection of instances per class was not suitable for these data sets. Thus, our local search strategy could benefit from a preliminary instance selection step before optimising the location of the instances, as proposed in [30].

In order to understand the significance of the provided results, we applied the Wilcoxon test to establish a fair comparison with the state-of-the-art. Table 5 displays the result of this comparison.

Table 5: Summary of the Wilcoxon test. The symbol •indicates that the method in the row outperforms the method in the column. The symbol ∘indicates that the method in the column outperforms the method of the row. Upper and diagonal of level significance 0.9 and 0.95, respectively

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)
NN (1)	-	•		•		•	•	•	•	•	∘		•	∘	∘	∘	
LVQ3 (2)	∘	-	∘		∘	•			∘		∘	∘		∘	∘	∘	∘
MSE (3)		•	-	•	•	•	•		•		∘	•	•	∘	∘	∘	
DSM (4)	∘		∘	-	∘	•			∘		∘	∘		∘	∘	∘	∘
LVQTC (5)		•		•	-	•	•				∘			∘	∘	∘	∘
VQ (6)	∘	∘	∘	∘	∘	-		∘	∘	∘	∘	∘	∘	∘	∘	∘	∘
AVQ (7)	∘		∘		∘		-	∘	∘	∘	∘	∘		∘	∘	∘	∘
HYB (8)	∘					•		-			∘			∘	∘	∘	∘
LVQPRU (9)	∘	•	∘	•		•	•		-		∘			∘	∘	∘	∘
ENPC (10)	∘					•				-	∘			∘	∘	∘	∘
PSO (11)	•	•	•	•	•	•	•	•	•	•	-	•	•		∘		•
AMPSO (12)		•	∘			•					∘	-	•	∘	∘	∘	∘
PSCSA (13)			∘								∘		-	∘	∘	∘	∘
IPADE (14)	•	•	•	•	•	•	•	•	•	•		•	•	-	∘		•
SSMA-SFLSDE (15)	•	•	•	•	•	•	•	•	•	•	•	•	•	•	-	•	•
LSIR (16)		•		•	•	•	•	•	•	•		•	•		∘	-	•
SALSIR (17)		•		•	•	•	•	•	•	•	∘	•	•	∘	∘		-

Table 5 highlights that the hybrid SSMA-SFLSDE algorithm remains to be the best algorithm, outperforming all the other methods. However, it should be remarked that SSMA-SFLSDE is composed of two population-based metaheuristics (a binary search to select relevant instances, and an adjustment of the position based on differential evolution). The selection of an appropriate number of instances per class is a well-known issue for instance generation techniques [29, 30], and the instance selection mechanism of SSMA-SFLSDE helps it to reduce overfitting and improve test accuracy. In this preliminary study, LSIR and SALSIR are naively used without a careful selection of instances per class.

More generally, LSIR and SALSIR are remarkably simpler than all the metaheuristic-based algorithms used in this study, and perform only a local search of the decision space. Despite these limitations, we observe that the proposed methods are competitive with a way more complex population-based metaheuristics such as PSO and IPADE. This study can be viewed as a stepping stone towards the generation of a hybrid algorithm that employs LSIR and SALSIR.

4.3 The effect of the surrogate

Regarding the performance of LSIR and SALSIR, we should make two considerations. On the one hand, Table 3 shows that LSIR appears to outperform SALSIR on both test and training accuracy. On the other hand, training results suggest that the surrogate variant suffers from overfitting less than the its baseline counterpart. However, the Wilcoxon test finds significant differences between the two algorithms (in the test phase) at a level of significance $\alpha = 0.9$ and that LSIR tends to outperform SALSIR. As an example of this fact, Fig. 1 shows the convergence plot on a single partition of the Bupa data. We can observe that both algorithms progress steadily but LSIR marginally outperforms SALSIR. This result was expected since a surrogate assisted algorithm often deteriorates the performance of its counterpart that uses only the true objective function, see e.g. [11, 19].

From the perspective of the computational saving, in the case depicted in Fig. 1, SALSIR saved 1991 evaluations with respect to LSIR. Since the purpose of a surrogate assisted algorithm is to reduce the number of objective function calls, we reported in Fig. 2 a histogram displaying the total number of evaluations and the number of saved evaluations by the surrogate variant. On average, around 15% of the evaluations have been saved.

5 Conclusion

This paper proposed a local search algorithm for addressing the large scale challenges imposed by instance reduction problems. The proposed local search is also endowed with a local surrogate model to mitigate the computational cost generated by objective function calls. The proposed local search algorithm can potentially be used within optimisation frameworks, such as portfolios, hyper-heuristics, and memetic algorithms. Numerical results indicate that the use of

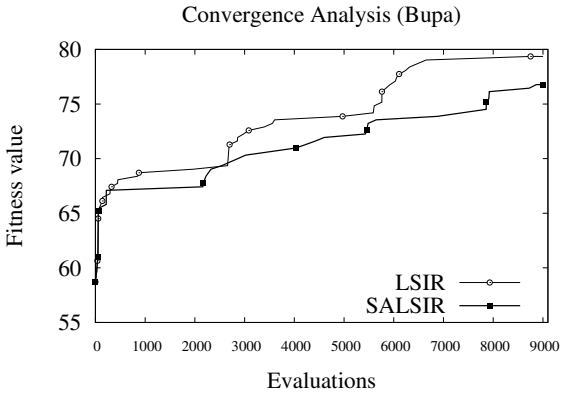

Fig. 1: Convergence plot example on one partition of Bupa data set

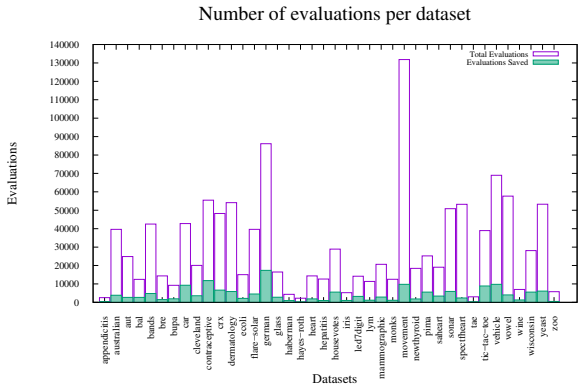

Fig. 2: Objective function calls saved by SALSIR (average 15%)

local search algorithms is a promising subfield of optimisation for addressing instance reduction problems. The proposed local search, despite its algorithmic naivety, outperformed numerous classical algorithms for instance reductions and is competitive with sophisticated population-based metaheuristics representing the state-of-the-art. The comparison between the versions with and without surrogate assisted model shows that the proposed surrogate design/implementation allows for an approximately 15% saving on the number of objective function calls, with a relatively small loss in accuracy. As a future work, we will investigate the integration of the proposed local search within advanced instance reduction algorithms to address larger classification problems.

References

1. Cano, J.R., Herrera, F., Lozano, M.: Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. IEEE Transactions on Evolutionary Computation **7**(6), 561–575 (Dec 2003)
2. Caraffini, F., Neri, F., Iacca, G.: Large scale problems in practice: The effect of dimensionality on the interaction among variables. In: Squillero, G., Sim, K. (eds.) Applications of Evolutionary Computation. pp. 636–652. Springer (2017)
3. Caraffini, F., Neri, F., Iacca, G., Mol, A.: Parallel memetic structures. Information Sciences **227**(0), 60 – 82 (2013)
4. Caraffini, F., Neri, F., Picinali, L.: An analysis on separability for memetic computing automatic design. Information Sciences **265**, 1–22 (2014)
5. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1), 21–27 (1967)
6. Dhar, V.: Data science and prediction. Commun. ACM **56**(12), 64–73 (2013)
7. García, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(3), 417–435 (2012)
8. García, S., Cano, J.R., Herrera, F.: A memetic algorithm for evolutionary prototype selection: A scaling up approach. Pattern Recognition **41**(8), 2693 – 2709 (2008)
9. García-Pedrajas, N., de Haro-García, A., Prez-Rodrguez, J.: A scalable memetic algorithm for simultaneous instance and feature selection. Evolutionary Computation **22**(1), 1–45 (2014)
10. Hidalgo, B., Goodman, M.: Multivariate or multivariable regression? Am J Public Health **103**, 39–40 (2013)
11. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61 – 70 (2011)
12. Jobson, J.D.: Multiple Linear Regression, pp. 219–398. Springer (1991)
13. Krasnogor, N.: Toward Robust Memetic Algorithms. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms, pp. 185–207. Studies in Fuzzines and Soft Computing, Springer, Berlin, Germany (2004)
14. Krawczyk, B., Triguero, I., García, S., Woźniak, M., Herrera, F.: Instance reduction for one-class classification. Knowledge and Information Syst. **59**(3), 601–628 (2019)
15. Li, X., Yao, X.: Cooperatively Coevolving Particle Swarms for Large Scale Optimization. Evolutionary Computation, IEEE Transactions on **16**(2), 210–224 (2012)
16. Lim, D., Jin, Y., Ong, Y.S., Sendhoff, B.: Generalizing Surrogate-assisted Evolutionary Computation. IEEE Transactions on Evolutionary Computation **14**(3), 329–355 (2010)
17. Lin, S.F., Cheng, Y.C.: A separability detection approach to cooperative particle swarm optimization. In: Proceedings of the International Conference on Natural Computation. pp. 1141–1145 (2011)
18. Nanni, L., Lumini, A.: Particle swarm optimization for prototype reduction. Neurocomputing **72**(4-6), 1092–1097 (2008)
19. Neri, F., Garcia, X.d.T., Cascella, G.L., Salvatore, N.: Surrogate Assisted Local Search on PMSM Drive Design. COMPEL: International Journal for Computation and Mathematics in Electrical and Electronic Engineering **27**(3), 573–592 (2008)
20. Nguyen, P.T.H., Sudholt, D.: Memetic algorithms beat evolutionary algorithms on the class of hurdle problems. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1071–1078. GECCO '18, ACM (2018)

21. de Oca, M.A.M., Cotta, C., Neri, F.: Local search. In: Neri, F., Cotta, C., Moscato, P. (eds.) Handbook of Memetic Algorithms, pp. 29–41. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
22. Ong, Y.S., Nair, P.B., Lum, K.Y.: Max-Min Surrogate-Assisted Evolutionary Algorithm for Robust Design. IEEE Trans. Evol. Comp. **10**(4), 392–404 (2006)
23. Regis, R.G.: Surrogate-assisted particle swarm with local search for expensive constrained optimization. In: Korošec, P., Melab, N., Talbi, E.G. (eds.) Bioinspired Optimization Methods and Their Applications. pp. 246–257. Springer (2018)
24. Ros, R., Hansen, N.: A simple modification in cma-es achieving linear time and space complexity. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) Parallel Problem Solving from Nature – PPSN X. pp. 296–305. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
25. Tenne, Y., Goh, C.K.: Computational Intelligence in Expensive Optimization Problems. Springer Publishing Company, Incorporated (2010)
26. Tong, H., Huang, C., Liu, J., Yao, X.: Voronoi-based efficient surrogate-assisted evolutionary algorithm for very expensive problems. In: IEEE Congress on Evolutionary Computation. pp. 1996–2003 (2019)
27. Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization **7**(1), 1–25 (1997)
28. Triguero, I., Derrac, J., García, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics–Part C **42**(1), 86–100 (2012)
29. Triguero, I., García, S., Herrera, F.: IPADE: Iterative prototype adjustment for nearest neighbor classification. IEEE Transactions on Neural Networks **21**(12), 1984–1990 (2010)
30. Triguero, I., García, S., Herrera, F.: Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. Pattern Recognition **44**(4), 901–916 (2011)
31. Triguero, I., Peralta, D., Bacardit, J., Garcia, S., Herrera, F.: A combined mapreduce-windowing two-level parallel scheme for evolutionary prototype generation. In: IEEE Congr. on Evolutionary Computation. pp. 3036–3043 (2014)
32. Triguero, I., Gonzlez, S., Moyano, J.M., Garca, S., Alcal-Fdez, J., Luengo, J., Fernndez, A., del Jess, M.J., Snchez, L., Herrera, F.: Keel 3.0: An open source software for multi-stage analysis in data mining. International Journal of Computational Intelligence Systems **10**, 1238–1249 (2017)
33. Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F.: MRPR: A mapreduce solution for prototype reduction in big data classification. Neurocomputing **150**, 331–345 (2015)
34. Tseng, L.Y., Chen, C.: Multiple trajectory search for Large Scale Global Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 3052–3059 (2008)
35. Wang, Y., Yin, D., Yang, S., Sun, G.: Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints. IEEE Transactions on Cybernetics **49**(5), 1642–1656 (2019)
36. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput. **15**(11), 2175–2185 (2011)
37. Zhou, Z., Ong, Y.S., Lim, M.H., Lee, B.S.: Memetic algorithm using multi-surrogates for computationally expensive optimization problems. Soft Computing **11**(10), 957–971 (2007)