The feature selection problem often occurs in pattern recognition and, more specifically, classification. Although these patterns could contain a large number of features, some of them could prove to be irrelevant, redundant or even detrimental to classification accuracy. Thus, it is important to remove these kinds of features, which in turn leads to problem dimensionality reduction and could eventually improve the classification accuracy. In this paper an approach to dimensionality reduction based on differential evolution which represents a wrapper and explores the solution space is presented. The solutions, subsets of the whole feature set, are evaluated using the \( k \)-nearest neighbour algorithm. High quality solutions found during execution of the differential evolution fill the archive. A final solution is obtained by conducting \( k \)-fold cross-validation on the archive solutions and selecting the best one. Experimental analysis is conducted on several standard test sets. The classification accuracy of the \( k \)-nearest neighbour algorithm using the full feature set and the accuracy of the same algorithm using only the subset provided by the proposed approach and some other optimization algorithms which were used as wrappers are compared. The analysis shows that the proposed approach successfully determines good feature subsets which may increase the classification accuracy.

Keywords: classification, differential evolution, feature subset selection, \( k \)-nearest neighbour algorithm, wrapper method.

1. Introduction

The problem of selecting the features to use during the classification of patterns is very common. When a problem originates from the real world, it is often very hard to tell the difference between the necessary and differentiating features from the ones that are not. At a first glance, it may seem that a higher number of features guarantees a better classification accuracy, but that is often not the case. Through the use of all available features, some redundant, unnecessary or even detrimental ones could be included, which would eventually cause poor classification results and reduce the classification accuracy. It is not always possible to know up-front which features should be included in the classification process and which should be left out. For the stated reason, various methods have been developed which evaluate the features. These approaches can be divided into Ranked Selection (RS) methods (e.g., Wu et al., 2009; Balakrishnan et al., 2008), and into Feature Subset Selection (FSS) methods that use either filters (Ferreira and Figueiredo, 2012) or wrappers, where search methods such as Differential Evolution (DE) (Khushaba et al., 2008), Genetic Algorithms (GAs) (Yusof et al., 2012), ant colony optimization (Kubir et al., 2012) and particle swarm optimization (Chuang et al., 2011) are applied.

This paper proposes a wrapper based FSS approach. The classifier used is the \( k \)-Nearest Neighbour (\( k \)-NN) algorithm and the DE algorithm adapted for binary space operation is used as a wrapper. The proposed approach includes an archive which stores a selected number of solutions obtained through the process of optimization, i.e., good quality solutions found during the DE run. A final solution is then obtained from the archive by performing a \( k \)-fold cross-validation of its solutions and selecting the best one. This process is expected to yield a more refined subset selection. The archive alongside with the \( k \)-fold cross-validation attempts to ensure the selection of the most salient features from the set.

The remainder of the paper is organized as follows. In Section 2 the classification problem is defined, the method of feature selection is described and a brief overview of related work is given. It also describes the \( k \)-NN algorithm, which is used as a classifier due to its
simplicity and good performance. A brief description of DE, as well as a couple of approaches to its adaptation to binary optimization problems, is given in Section 3. The proposed approach, which combines DE, adapted for operation in the binary space, and the k-NN algorithm, is described in Section 4. Section 5 presents the experiment set-up and the results of the experimental analysis conducted on several standard datasets.

2. Classification and feature selection

Classification belongs to the area of pattern recognition and it is its typical representative. The classification problem can be defined as follows. If a pattern set and it is its typical representative. The classification problem proportionally expands and with it the calculation complexity of the relation between feature vectors.

The classification accuracy depends greatly on the method used, but also on the underlying problem, i.e., the characteristics of the data on which the classification method is applied. Since each feature vector is represented by a continent, “0” or “AB” for blood type). As the number of features increases, the dimensionality of the problem proportionally expands and with it the calculation complexity of the relation between feature vectors.

An increased number of features does not always guarantee a better classification quality nor does it improve differentiation between the classes. Additional features are often introduced in order to better describe the problem, and one of the consequences could be the introduction of irrelevant, redundant and even features that are detrimental to classification accuracy (Dash and Liu, 1997). To avoid this problem, an attempt is made to reduce the number of features used in classification and this process is called feature selection (Javed et al., 2012).

The selection is performed through four formal steps: generating a subset, subset evaluation, the break criterion and result evaluation (Dash and Liu, 1997), and two ways of conducting this process are commonly used. The first one, RS, is the process of selection where weights are given to all of the features. After that, they are ranked and a predetermined number of the most highly ranked features is used. The second one, FSS, is the process of selection where various methods are used to determine the best subset of the full feature set. In this paper, an FSS approach is used, and the goal is to determine one subset of the feature set that maximizes the result of the fitness function. An attempt is made to find a binary vector \( o^* \) which is used as a mask on the full feature vector to yield a reduced feature set that will be used for the classification. The yielded reduced feature set must achieve the highest possible classification accuracy as described by an objective function \( f \), which is formally shown in

\[
o^* \in O = \arg \max_{b \in \{0,1\}^n} f(S,I,b). \tag{2}
\]

Here \( O \) represents the set of binary vectors that yield a reduced feature vector for which the objective function \( f(S,I,b) \) reaches a global maximum, where \( I \) is the set of input patterns to be classified. It is clear that this problem can also be treated as a minimization problem if the classification error is observed. Additionally, it is important to mention that FSS methods are commonly divided into filter and wrapper ones. Filter methods are somewhat faster and represent a form of preprocessing that incorporates subset or individual feature evaluation without the inclusion of a classifier. Ferreira and Figueiredo (2012) gave an example of filter usage where they showed filter effectiveness on feature vector sets of up to \( 10^5 \) features. They used two different filters. The first one is based on the idea that the importance of a particular feature is proportional to its dispersion. The second one uses a redundancy measure between the features.

In wrapper methods the algorithm that selects the features uses a classification algorithm for evaluation. Accordingly, wrapper methods are more precise but computationally more complex (Wang et al., 2011), and they also heavily depend on the data selected for classifier development. Since these data guide the selection, they can lead to over-fitting (Loughrey and Cunningham, 2004). A broad spectrum of various wrappers is used in today’s approaches. For example, the forward and the backward floating search and their combinations are commonly used, where one feature is added or reduced at a time, depending on the classification accuracy, but also some advanced methods based on the aforementioned ones which take into account feature dependencies (Michalak and Kwaśnicka, 2006).
Evolutionary algorithms are also used. For instance, Raymer et al. (2000) employed a GA as a wrapper and a k-NN algorithm as a classifier. The dimensionality reduction problem was approached by these authors by using weight factors and one or more bit masks for feature selection. Kubir et al. (2011) presented a hybrid GA for feature selection. To enable a more refined search space exploration, they built a local search algorithm based on the difference and informative nature of features calculated based on correlation information in the GA. The classifier used was an ANN. Currently, hybrid approaches combining filter and wrapper methods or even RS and FSS are becoming more and more popular. One such method was presented by Hsu et al. (2011). Firstly they combined the F-Score filter, which calculates discriminatory possibilities of each feature, and the information gain filter, which selects features that contain more information. Through this filter combination they select the candidates and create a new, reduced feature set. After that a wrapper method is used for FSS. Sequential floating search is used as a wrapper and an SVM with a radial basis function kernel as the classifier.

The k-nearest neighbour algorithm is a classifier that works based on the classes of k nearest vectors from S, which are nearest to the input vector based on some distance measure (Zhu et al., 2007). Distance measures used to determine the relation between two samples can be distance functions such as Euclidean, Mahalanobis, Minkowski, Manhattan and others. In this paper the Euclidean distance is used as the distance measure. The algorithm parameter k determines the number of neighbours from S to be used when the label of the input vector is chosen. As stated by Garcia et al. (2010), the most common way to determine the value of k is through the process of cross-validation. Most often, k is selected as an odd number between 1 (where the classifier is reduced to the Nearest Neighbour (NN) classifier) and several tens. The k-NN algorithm is one of simpler classifiers, but due to its simplicity and good performance it is often used in classification problems. In time it has been improved and enhanced according to the needs of various problems. An overview of different versions is given by, e.g., Bhatia and Vandana (2010) or Jiang et al. (2007).

3. Differential evolution

Differential Evolution, or briefly DE (Storn and Price, 1997; Price et al., 2005; Xinjie and Mitsuo, 2010), is a simple but effective search method for continuous optimization problems. According to Xinjie and Mitsuo (2010), DE represents a direction based search that maintains a vector population of candidate solutions. Like other usual Evolutionary Algorithms (EAs), it uses mutation, crossover and selection. The key part of DE, which differentiates it from standard EAs, is the mutation operator that perturbs the selected vector according to the scaled difference of the other two members of the population. The operation of DE is shown with pseudo-code as Algorithm 1.

**Algorithm 1.** DE in pseudo-code.

1: Initialization and parameter setting
2: while termination condition not met do
3:   for all population member—vector v_i do
4:     create mutant vector u_i
5:     crossover v_i and u_i to create trial vector t_i
6:   end for
7:   for all population member—vector v_i do
8:     if f(t_i) ≤ f(v_i) then
9:       v_i ← t_i
10:   end if
11: end for
12: end while

The population of size NP contains vectors and each vector v_i, of dimensionality D, consists of real-valued parameters, v_i = (v_i1, ..., v_iD) ∈ ℝ^D, for i = 1, ..., NP. Usually the population is initialized with vectors of values obtained randomly in the interval [v_ub, v_lb], where v_ub and v_lb represent the lower and upper bound, respectively. In each generation, a new population is created through mutation and crossover. This new population is composed of the so-called trial vectors t_i. For each member of the current population, v_i (called the target vector), a new corresponding mutant vector u_i is formed using mutation. The mutation is conducted according to

\[ u_i = v_r1 + F \cdot (v_r2 - v_r3) \]  

(3)

Here u_i is a mutant while v_r1, v_r2 and v_r3 are population vectors selected randomly with the condition i ≠ r1 ≠ r2 ≠ r3, and F ∈ [0, ∞) is the scale factor which represents a parameter of the algorithm. After the mutation, crossover occurs between the target vector v_i and the corresponding mutant u_i, creating a trial vector t_i. The crossover is done as follows:

\[ t^j_i = \begin{cases} u^j_i & \text{if } U[0, 1] \leq \text{CR or } j = r_j, \\ v^j_i & \text{otherwise} \end{cases} \]  

(4)

for j = 1, ..., D. Here t_i is a trial vector obtained through crossover, U[0, 1] is a variable with its value randomly selected from the interval [0, 1] with uniform distribution, r_j is a random variable with the value from the set {1, ..., D}, while CR ∈ [0, 1) is the crossover rate and represents a parameter of the algorithm. The described crossover is called the binomial crossover.

Once the trial vector population has been created, vectors that transfer over to the next generation, i.e., which
will constitute the new population, are selected. A given trial vector \( t \), replaces the corresponding target vector \( v_i \), if it is of equal or lesser cost, according to the given objective/fitness function.

Due to its simplicity, DE is a very popular search method that has been successfully applied to various problems. Here, the classic DE is described that is commonly denoted by DE/rand/1/bin (Storn and Price, 1997). However, various other variants/strategies are presented in the literature. A very comprehensive overview of different DE strategies, as well as application areas, is given by Das and Suganthan (2011).

### 3.1. Differential evolution within the binary space.

Differential evolution was originally proposed for solving continuous optimization problems. Primarily, because of the nature of the mutation operator, DE is not directly applicable to discrete optimization problems. Still, it is possible to use it for discrete optimization, and the literature (e.g., Lichtblau, 2012; Vegh et al., 2011; Zhang et al., 2008) proposes different ways of achieving the aforementioned. Several different approaches for applying DE to binary optimization problems have been proposed (Engelbrecht and Pampara, 2007).

Angle Modulated DE (AMDE), proposed by Pampara et al. (2006), boils down the basic problem (in the binary space) to a simpler one in the continuous space. AMDE optimizes the coefficients of a function \( h(x) \), given in (5), in the continuous space which will be used for the creation of a binary vector. AMDE reduces the \( D \)-dimensional problem in the binary space to a 4-dimensional problem in continuous space,

\[
 h(x) = \sin(2\pi(x-a) \cdot b \cdot \cos(2\pi(x-a) \cdot c)) + d. \tag{5}
\]

Here \( x \) is an independent variable while \( a, b, c, \) and \( d \) are the function coefficients that determine its shape (usually constrained to the range \([-1,1])\). The binary vector is obtained by sampling the given function in equal intervals. If the function has a positive value in the given interval, a 1 is inserted, and a 0 otherwise.

An alternative, simple and intuitive approach is to use vectors of \( D \) dimensions, i.e., vectors of the same size as the problem in the binary space. The real-valued parameters are constrained to the range \([0,1]\) and a binary vector is then derived from it by setting a 0 if the corresponding real-valued parameter is less than 0.5, or, otherwise, by setting a 1. We adopt this approach in the proposed algorithm since it proved, in our preliminary analysis, to yield better solutions than AMDE. Also, it resulted in a more stable algorithm.

In both of the aforementioned approaches the quality of each solution, i.e., the population member, is determined based on the evaluation of the obtained binary vector, using a fitness function that is defined in the binary space.

### 4. Proposed approach

As mentioned earlier, in this paper an approach is proposed based on FSS using a wrapper method. Accordingly, the proposed approach, hereinafter referred to as DE-\( k \)-ANN, combines the DE and \( k \)-NN algorithms. The former is used as a wrapper in the search for a subset of the given feature set of good quality, and the latter is used for the needs of evaluating found solutions. The employed DE for FSS, DE-\( k \)-NN, is based on DE/rand/1/bin as previously described in Section 3.1 In DE-\( k \)-NN the scale factor \( F \) is varied randomly as proposed by Das et al. (2005). More precisely, the scale factor used during mutation of a particular vector has a uniform random value in the range \([0.5,1] - F = 0.5 \cdot (1 + U(0,1))\). According to Das et al. (2005), the randomly varied scale factor should help maintain the population diverse throughout the search. Also, DE-\( k \)-NN incorporates an archive of fixed size which is used to store good quality solutions found during the optimization process, i.e., search. The mode of operation of DE-\( k \)-NN is shown in Fig. 1.

The archive size was set to 50 to include a variety of good quality solutions found during the search. The population is initialized randomly in DE-\( k \)-NN with \( v_i = U(0,1) \) for every \( i = 1,\ldots,NP \) and \( j = 1,\ldots,n \). Also, the first 50 solutions are copied to the archive, unless the population is smaller, in which case all population members are copied to the archive and the rest \((50-NP)\) is randomly generated. The population of trial vectors is created according to the selected strategy, DE/rand/1/bin. During the selection of vectors that will comprise the new population, a new binary vector is formed using the process described in Section 3.1 for each member of the current population and the trial vector population. The size of the binary vector corresponds to the number of features for the given classification problem. The binary vector, \( b = (b_1,\ldots,b_n) \in \{0,1\}^n \), determines which features will be used \((1)\), and which will not \((0)\) during classification. This vector is evaluated with the following negative value fitness function (Yang et al., 2011),

\[
 g(b_i) = o(b_i) - \lambda \cdot p(b_i) , \tag{6}
\]

where \( o(b_i) \) is the percentage of the classification accuracy using the \( k \)-NN algorithm, \( p(b_i) \) is the feature.
subset size, i.e., the number of features used, and \( \lambda \) is a penalty factor which has, in this paper, a value of 0.001. Larger values of \( g(b) \) indicate small feature subsets that yield high classification accuracy.

If any of the binary vectors contains only 0s, it is penalized, i.e., its fitness is set to a relatively very high constant value (in our case this value is 1000) to ensure the elimination of such a solution from transferring into the next generation.

Each time a trial vector is generated, it is considered for inclusion in the archive. A trial vector enters the archive only if it is better than the current worst and different from all the vectors present in the archive. The vectors in the archive are differentiated via the Hamming distance, which must be at least 1 (between the considered vector and each archive vector) for a vector to be granted access.

Once the DE execution is finished, a final solution is obtained from the archive. The motivation for using the archive is the fact that the optimization process tends to over-fit to training data resulting in loss of generality. As described, the archive contains distinct solutions of good quality, and the final solution is chosen from among them. The choice is made based on the quality of each solution obtained from a \( k \)-fold cross-validation. The \( k \)-fold cross-validation is a commonly employed cross-validation method that makes clever use of the available data. Thus, it should provide a more general evaluation of feature subsets compared with a single \( k \)-NN run. The archive solution that performed best in the cross-validation is chosen as the final solution. A motivation for this approach stems from the fact that it is possible to achieve similar classification quality with various feature subsets. Adding cross-validation and forcing the archive to hold only distinct feature vectors, but only those of high quality, provides an option to deduce which of them performs best on various test sets generated by the cross-validation. A solution that prevails should prove to be good in general. This way, the knowledge about the feature vectors that perform best is maintained throughout the search, and only a step of post-evaluation is added. It is worth mentioning that using the \( k \)-fold cross-validation during the optimization process would be computationally too demanding, but applying it to the archive solutions does not produce a significant computational overhead since the archive is relatively small.

5. Experimental analysis

With the purpose of evaluating the effectiveness of the proposed approach, DE-\( k \)-NN, an experimental analysis was conducted on several datasets. All datasets were obtained from the UCI machine learning repository (Frank and Asuncion, 2010), except the Texture (Alcalá-Fdez et al., 2011) dataset. The analysis is based on the evaluation of classification accuracy applying the \( k \)-NN algorithm with the full feature set and using the same measure and classifier with the use of the reduced feature set.

The selected datasets and their characteristics after preprocessing are shown in Table [1]. The sets were chosen to cover various parameter values in order to test different input pattern cases, i.e., the influence of the number of instances, feature set size and similar on the feature selection process results. The datasets were preprocessed to remove the features that are known to have no meaning for the classification, such as the instance name, the ordinal number of an instance and similar. Instances with missing features were not considered.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data-type</th>
<th># inst</th>
<th># feat</th>
<th># class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast Cat. Wis. (Or.)</td>
<td>Int.</td>
<td>683</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Dermatology</td>
<td>Cat., Int.</td>
<td>258</td>
<td>34</td>
<td>6</td>
</tr>
<tr>
<td>Glass id</td>
<td>Real</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Image seg.</td>
<td>Real</td>
<td>2310</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>Int., Real</td>
<td>351</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>Musk vers. 1</td>
<td>Int.</td>
<td>347</td>
<td>160</td>
<td>1</td>
</tr>
<tr>
<td>Liver MN</td>
<td>Real</td>
<td>360</td>
<td>90</td>
<td>15</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>Real</td>
<td>197</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>Sonar</td>
<td>Real</td>
<td>1600</td>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>Spambase</td>
<td>Int., Real</td>
<td>4597</td>
<td>57</td>
<td>2</td>
</tr>
<tr>
<td>Statlog (Veh. Silh.)</td>
<td>Int., Real</td>
<td>946</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>Texture</td>
<td>Int., Real</td>
<td>5500</td>
<td>40</td>
<td>11</td>
</tr>
</tbody>
</table>

5.1. Genetic algorithm used for comparison

Genetic algorithms (Xinjie and Mitsuo, 2010) are optimization and search methods inspired by genetics and natural selection, originally proposed for binary optimization problems. GAs have been successfully applied to a wide variety of problems (Yan et al., 2013; Martinović and Bajer, 2011). In order to better evaluate the performance of DE-\( k \)-NN, it was compared to a GA. A generational GA with binary tournament selection, one-point crossover and bit-flip mutation was implemented. Also, elitism was incorporated. A high-level outline of the GA used is given in Algorithm [2].

Binary tournament selection without replacement was chosen because it is easy to implement, has a small time complexity and exhibits a relatively small selection pressure. One-point crossover and bit-flip mutation were chosen since they are employed in the simple GA (Eiben and Smith, 2003; Xinjie and Mitsuo, 2010); also, according to Eiben and Smith (2003), bit-flip mutation is the most common mutation operator for binary encodings. Elitism was incorporated since it can significantly enhance GA performance, and it was implemented as follows. The best individual in the current population replaces the worst in the new (offsprings) population if it does not contain an equally good or better individual.

In the GA, the population is initialized randomly, i.e.,
where performance of DE-one core was utilized during algorithm execution. The OS. Due to the sequential algorithm implementation only (Intel E5800 @ 3.20 GHz), 4 GB of RAM and Windows 7 was carried out on a computer with a dual core processor normalized as 5.2. Experiment set-up. The experimental analysis NN was compared with a number of optimization algorithms which were used as wrappers. More precisely, it was compared with a GA (as described in Section 5.1), DE/rand/1/bin (adapted for the binary feature space the same way as DE-\(k\)-NN, and hereinafter referred to as stdDE) and AMDE. The aforementioned algorithms were implemented in the C# programming language. Before the start of the experiment, all data were normalized as

\[
N^j_i = \begin{cases} 
  1 & \text{if } \Delta = 0 \\
  1 + \frac{9}{\Delta} (X^j_i - \min_{s=1,\ldots,m} X^j_s) & \text{otherwise,}
\end{cases}
\]

where

\[
\Delta = \max_{s=1,\ldots,m} X^j_i - \min_{s=1,\ldots,m} X^j_s,
\]

and \(N^j_i\) is the normalized value of the feature \(X^j_i\), where \(i = 1,\ldots,m\) and \(j = 1,\ldots,n\). If the difference between the maximum and minimum values is 0, then it is clear that the value of this feature is equal for all feature vectors in the dataset. The normalized value is then set as 1 (for all feature vectors in the set) and it does not have any influence on the classification. Accordingly, it is expected that such a feature will be eliminated. Through normalization the values of all features were converted to values from the interval \([1, 10]\), which reduced the influence of the difference between different features on the classification results, as described by Raymer et al. (2000). In other words, the prevailing influence of one feature is disabled if the only reason for this influence is the range of its values. For instance, the influence of human height would be greater than that of width just because a person is usually taller than wider. In the case of non-numerical data, other distance measures can be employed (Li and Li, 2010), or several binary attributes can be added (1 representing the existence and 0 non-existence of the attribute), enabling the normalization of the feature and creating the same distance among categories. Once normalized, the dataset is divided into three proper subsets. The first subset contains 50% of the total feature vectors and represents the training set. These are the feature vectors with known class labels based on which the classification is performed. The second set contains 25% of the total feature vectors and represents the tuning set. These are the feature vectors used by the algorithms to determine which features are rejected, i.e., this set enables the \(k\)-NN algorithm to determine the quality of the solutions found by the wrapper algorithms. The third subset is the test subset which contains 25% of the vectors from the dataset. This subset is used to evaluate the final solution, i.e., the feature subset obtained by the one of the algorithms. Feature vectors selected for any of these sets are chosen from the initial dataset of feature vectors randomly, while maintaining the class distribution of the original set.

The parameter values for all algorithms used in the experimental analysis, obtained through extensive preliminary analysis, are displayed in Table 2. Furthermore, even though the proposed approach was designed with the general case in mind, the value of parameter \(k\) for the \(k\)-NN algorithm for a given input dataset is 1, thus creating its special case, the nearest neighbour algorithm. This parameter was used to alleviate the influence of parameter determination on the achieved results, since it should be determined through experimentation and the focus of this research was not classifier performance.

The value of \(k\) used in the \(k\)-fold cross-validation of archive solutions was set based on the size of the dataset used. Accordingly, its value was 3 for datasets containing less than 500 instances, 5 for datasets containing less than 1000 instances, and 10 for datasets containing 1000 or more instances. This way, the folds were of reasonable size. The union of both the training and tuning subsets was used for the creation of the folds in order to provide the largest possible number of data for the cross-validation. The population size and the maximum iteration number were 50 and 300, respectively, and were the same for all algorithms. Also, the termination condition was the same for all algorithms. More precisely, the algorithm execution is terminated if it reaches the assumed maximum number of iterations or earlier, if in 30 consecutive runs a solution of higher quality was not found. Relatively small values were used to keep the total

---

**Algorithm 2. GA: high-level outline.**

1: Initialization and parameter setting  
2: while termination condition not met do  
3: while offspring population not complete do  
4: select 2 distinct parents from current population  
5: cross over parents with probability \(p_c\) to produce 2 offspring  
6: mutate offspring with probability \(p_m\)  
7: evaluate offspring  
8: end while  
9: replace current population with offspring population  
10: apply elitism  
11: end while
number of evaluations on a reasonable level since each evaluation is time consuming. Since all the employed algorithms are stochastic search methods, ten independent runs were carried out for each algorithm and dataset. For each of these solutions (feature subsets), as well as for the full feature set, the NN algorithm was executed only once because it is a deterministic one.

### 5.3. Results and discussion

The first part of the experimental analysis results is presented in Table 2 and Fig. 2. The results show the datasets used, and display the mean classification accuracy (\(\mu\)) that each of the employed algorithms achieved. The table also displays the standard deviation (\(\sigma\)), maximum (\(\text{bst}\)), minimum (\(\text{wst}\)) and range (calculated as \(\text{bst} - \text{wst}\)) of the classification accuracy for each algorithm. Also, a statistical analysis of the pairwise comparison of the performance in terms of the resulting classification accuracy of DE-\(k\)-NN with the other wrappers considered is shown in Table 2. The table includes the sum of ranks (\(W\)), the obtained \(p\)-value, and the corresponding 95\% confidence interval. The statistical analysis was performed using the Mann-Whitney U (Wilcoxon rank-sum test) test—two-sided test, provided by the R software environment for statistical computing (R Core Team, 2013). This test was chosen since, according to Trawiński et al. (2012), it is more sensible than the \(t\)-test when the number of observations is small (10 in our case).

As can be seen from the table, the proposed approach shows promising results. In most (7 out of 12) test cases it outperforms the other feature selection algorithms (wrappers considered) and yields a higher classification accuracy than the NN algorithm. According to Table 2, the higher performance of DE-\(k\)-NN compared with the other wrappers is in most cases statistically significant. The performance improvements are most notable on the Breast Can. Wis. (Or.), Glass identification, and Image segmentation datasets. The improvements are shown to be statistically significant. In several cases (e.g., Spambase and Parkinsons), considerably higher performance is achieved compared with the other wrappers, and it is shown to be statistically significant compared with two of the three utilized wrapper methods. In several cases the deterministic NN algorithm shows the best performance, but even then the difference is small and DE-\(k\)-NN outperforms other tested wrappers. The accuracy it achieves is close to that of the NN but is attained with far fewer features. Since classifier development has to be performed only once, the cost should prove its worth in the long run since using fewer features use less time to classify an unknown sample. It should be noted that the costs involved might not only be related to time consumption, so the performance and speed should be weighted on the case-to-case basis. The performance of the wrapper depends on the fitness function guiding the search process, and the over-fitting occurring due to the adaptation of the solutions to the data used in classifier development could lead to under-performance on the independent set of data used for testing. Based on these remarks, it can be concluded that, as discussed in Section 1, each of the datasets used contains some irrelevant or redundant features and/or some that are detrimental to the classification accuracy.

The second part of the results is shown in Table 2 and Fig. 3. They represent feature reduction results and display the mean number of features (\(\mu\)) that each of

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE-kNN</td>
<td>(C = 0.95, F = 0.5 \cdot (1 + U(0, 1)))</td>
</tr>
<tr>
<td>GA</td>
<td>(p_c = 0.9, p_m = 0.03)</td>
</tr>
<tr>
<td>stdDE</td>
<td>(C = 0.95, F = 0.3)</td>
</tr>
<tr>
<td>AMDE</td>
<td>(C = 0.95, F = 0.25)</td>
</tr>
</tbody>
</table>

Table 2. Algorithm parameters used.
the employed algorithms reduced the full set to. The table also contains the standard deviation (σ), minimum (bst), maximum (wst) and range (bst − wst) of the number of features for each algorithm. Since the goal of the proposed approach is not to achieve the minimum amount of features, but to generate the best general solution, it is understandable that the proposed approach is not producing feature vectors with minimum features. However, it still gives reasonable reduction and in some cases (e.g., Statlog Vehicle Silhouettes) even achieves the best results with the smallest feature subset. When compared with the full feature set, the result is substantial for each of the wrappers considered since the average feature number across all datasets is more than halved. All datasets display a high degree of feature reducibility, and that fact is even more evident on datasets with higher dimensional data (e.g., Libras Movement).

It is interesting to observe that the ratio of feature set size and the average size of the feature subset found by the DE-kNN and other algorithms varies for different datasets. If the dataset is quite large, DE-kNN will not necessarily discard a substantial amount of the features and vice versa, which is noticeable from the Glass identification, Parkinsons and Libras Movement datasets. This is understandable since the reduction can be pursued up to a certain level that depends on the data in the dataset and the fitness function that determines the quality of an individual feature subset, which is given in the proposed approach, and for the other considered wrappers as well. Furthermore, the proposed approach evaluates candidates not based on the reduction, but on their performance on k-fold cross-validation. Although the candidates are all fairly reduced, it is quite possible that although the most reduced feature set performs exquisitely on the tuning set, it is not so good in general. Therefore the additional features included in the solution by DE-kNN in regard to the sets provided by other tested wrappers is justified, since the achieved accuracy is higher for the independent test set containing previously unknown data.

It is worth noting that it is possible to achieve the same classification accuracy with feature subsets of different sizes, and also different classification accuracy with subsets of the same size. In the latter case, even though these solutions have the same amount of features, they are not necessarily the same features. The aforementioned is not strange since the search is guided by the samples contained in the tuning subset of the dataset which are different than the ones in the independent test subset. That signifies the possibility of different feature subsets achieving the same or very similar classification accuracy on the tuning subset, but a different accuracy on

### Table 4. Statistical analysis of the pairwise comparison of DE-kNN with the other wrappers.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DE-kNN</th>
<th>DE-kNN vs. GA</th>
<th>DE-kNN vs. AMDE-kNN</th>
<th>DE-kNN vs. DE-kNN</th>
<th>DE-kNN vs. GA</th>
<th>DE-kNN vs. AMDE-kNN</th>
<th>DE-kNN vs. DE-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.C.W. Or.</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Statlog</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Musk 1</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Dermatology</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>B.C.W. Or.</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Statlog</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Musk 1</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Dermatology</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>B.C.W. Or.</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Statlog</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Musk 1</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Dermatology</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Fig. 2. Comparison of the classification accuracy of the NN algorithm using the full feature set and the average classification accuracy of the NN using the solutions found by wrapper algorithms.

Fig. 3. Number of employed features per algorithm for each of the datasets used.
The final part of the analysis is given in Table 6 and Fig. 4. The results represent the execution time of each of the algorithms per dataset. Both show the average number of wrapper evaluations (since it archives top number of solutions found throughout DE execution) and a method of post-evaluation using archived and a method of post-evaluation using.

As can be deduced from the presented data, the proposed approach has a slightly longer execution time. This is most evident on datasets with most instances. The proposed approach has a slightly longer execution time.

Table 5. Feature reduction analysis.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AMDE</th>
<th>std AMDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.C.W. Or.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Dermatology</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Glass</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Image seg.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Irisphere</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Libras Mov.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Parkinson</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Spambase</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Staring V.S.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Texture</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
</tbody>
</table>

Table 6. Execution time analysis.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DE-kNN</th>
<th>std DE-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.C.W. Or.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Dermatology</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Glass</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Image seg.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Irisphere</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Libras Mov.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Parkinson</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Spambase</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Staring V.S.</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Texture</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>test</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>wst</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>range</td>
<td>-0.00</td>
<td>-0.00</td>
</tr>
</tbody>
</table>
Experimental analysis was carried out on several standard datasets of varying sizes and numbers of features. The results of the analysis show the usefulness of this approach because in almost every case the results are better than the ones of the full feature set and achieve higher accuracy than other tested wrapper methods. The promising results achieved by the proposed method were statistically evaluated by a pairwise comparison between it and the other utilized wrappers using the Mann–Whitney U test. It was shown that the differences in performance were in most cases statistically significant. This means that, alongside the problem complexity reduction (through the reduction of features used), classification accuracy is improved.

Future work includes potential improvements of the proposed solution through combining it with some filter method to discard some features up-front and to reduce the execution time or by using a more advanced classifier such as an artificial neural network. It should also focus on adjusting the solution using other methods of post-evaluation, other than \( k \)-fold cross-validation and parameter optimization to achieve reduced running times while maintaining the solution quality.

Acknowledgment

This work was supported by the research project grant no. 165-0362980-2002 of the Ministry of Science, Education and Sports of the Republic of Croatia. The authors would like to thank the anonymous reviewers for their useful comments that helped improve this paper.

References


A differential evolution approach to dimensionality reduction for classification needs


Goran Martinović is a full professor of computer science. He obtained his B.Sc.E.E. degree from the Faculty of Electrical Engineering, J.J. Strossmayer University of Osijek, in 1996. In 2000 and 2004, he obtained his M.Sc. and Ph.D. degrees in computer science, both from the Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include distributed computer systems, fault-tolerant systems, real-time systems, artificial intelligence and medical informatics. He is a member of the IEEE, ACM, IACIS, Cognitive Science Society, KOREMA and IEEE SMC Technical Committee on Distributed Intelligent Systems.

Dražen Bajer received the Bachelor and Master degrees in computer engineering from the Faculty of Electrical Engineering, J.J. Strossmayer University of Osijek in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree at the Faculty of Electrical Engineering. His research interests are computational intelligence methods and their applications, and unsupervised classification. He is an IEEE graduate student member.

Bruno Zorić received the Bachelor and Master degrees in computer engineering from the Faculty of Electrical Engineering, J.J. Strossmayer University of Osijek in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree at the Faculty of Electrical Engineering. His research interests are supervised classification and affective computing. He is an IEEE graduate student member.

Received: 6 January 2013
Revised: 20 August 2013
Re-revised: 27 November 2013