Multi/Many-Objective Optimization in Feature Selection

Duarte Côrte-Real Rolim
duarte.rolim@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
November 2017

***Abstract*—Feature selection, the removal process of non-essential variables in a dataset, is a crucial step in any machine learning algorithm since it not only simplifies the model but also increases the predictor's performance. However, admitting that the removal of unnecessary features does not improve all performance metrics simultaneously, different applications require distinct classifier's performance metrics. Additionally, it might be advantageous to use diverse metrics for the process of finding good feature subsets.**

This work has three major contributions relating to binary classification using a wide-set of wrapper performance metrics and multi-class classification dividing it into several binary sub-problems: firstly, a relationship analysis between wrapper's performance metrics is made, comparing and conjecturing which are made redundant by each other; The second contribution is a first study on the sets of classifier's performance metrics' performance in feature selection, testing if the inclusion of more than 2 objectives is beneficial; Lastly, a feature selection decision interface was built, which aids in the solution selection process.

The first analysis shows that less than a handful of the tested performance metrics for binary classification is not simultaneously improved in the feature selection process. Using state of the art multi-objective algorithms, results suggest a better performance, in terms of convergence and diversity, of feature selection when using a high number of objectives in binary classification, despite some being redundant. In relation to multi-class classification, only diversity is improved when dividing it into several binary sub-problems and using accuracy to each one.

***Index Terms*—Feature Selection; Wrapper Evaluation; Evolutionary Computation; Multi-Objective Optimization; Decision Interface.**

I. Introduction

Society rapidly evolves into a future where computers will be able to act and react without being explicitly programmed to do so. Hence, computers should be able to acquire data and learn with it, making them suited to predict situations.

Artificial intelligence algorithms are being applied to innumerable everyday problems such as social media, internet security, and market analysis, working in cooperation with decision makers. These algorithms are usually trained by example, i.e., data is acquired in order to train a classifier.

Nowadays data collection and storage are available effortlessly, allowing researchers and scientists to gather and store enormous datasets, frequently prioritizing the storage of all variables, disregarding their importance. There are also applications where variable relevance is unknown *a priori* and all variables end up being collected. Consequently, most datasets became contaminated with redundant, noisy or simply not relevant variables. The data collection and storage evolution was of such magnitude, that only two decades ago a dataset with more than 20 variables was considered large-scale [1], while nowadays that designation is used when dealing with thousands of variables.

Feature selection is the exclusion process of those unnecessary variables. Fewer features results in a simpler, more transparent model, quicker to train and test, and easier to understand. Additionally, removing redundant and noisy variables improves performance by increasing the predictive value of data, avoiding the curse of dimensionality [2].

Feature selection is a complex duty not only considering the enormous search space for medium-sized datasets, which makes feature selection an NP-hard problem [3], but also due to the intricate interactions between features. These interactions might make features useless by themselves be beneficial when paired with others, and apparently redundant ones can also be advantageous when combined [2].

Any feature selection algorithm is characterized by two main traits, a search procedure, and an evaluation function. The first is simply the operation of searching for promising subsets, and the latter is how each subset is evaluated.

A. Search Procedure

The search procedure is one of three types. Exhaustive is the brute-force metric evaluating all $2^N - 1$ possible solutions, with N being the total number of features. This is impractical even for small sized datasets (for example, $N = 20$ requires 1048575 combinations). Heuristic search procedures are straightforward algorithms that respond to exhaustive's disadvantage by only trying a maximum of N^2 combinations [4]. In each generation, the features yet to be selected (rejected) are considered for selection (rejection). For their simplicity and robustness, these methods are undoubtedly the most used [5]. The colossal search space and intricate feature interaction suggests that the best option is randomized methods (also known as meta-heuristics). Its ability to escape local optimums, a big disadvantage of heuristics, and the fact that most of this algorithms are population-based, i.e., a single run produces several solutions, are major advantages. Nevertheless, they're slower and more complex and considering their stochasticity, different runs will likely not result in the same population.

B. Evaluation Function

The process of ranking feature subsets in order to compare them can be divided into four kinds. The simplest kind, filter, includes model-free methods that assign a score based on a statistical value of the dataset to each subset. Despite being

very fast and simple to implement, most of these are unable to consider the relationship between features, especially when the rank is for each feature rather for a whole subset. In wrapper methods, a subset's score is the performance of a classifier trained and tested using only the features included in the subset. These are quite slower because they involve training and testing a machine, but usually show better results than filter methods, despite the possibility of over-fitting. The third type of methods combines the strengths of both filter and wrapper approaches. A model is trained and tested, but simultaneously it's able to understand which features best contribute to the model's performance. Therefore the training and the ranking part of the methods cannot be separated. They are more complex and heavy than all previously discussed algorithms, and also more recent. Finally, hybrid methods use filter and wrapper independently. Firstly, a combination of filter methods are used to reduce the original set of features, and then wrapper evaluation occurs to the remaining features to find the best subset.

C. Motivation and Contribution

Despite slower and more complex, or perhaps as a consequence of it, combining **randomized search** procedures, or meta-heuristics, with **wrapper evaluation** is the most promising approach to feature selection in the current state of technology. Randomized search is undeniably the best approach to successfully explore such an immense search space, for its stochastic nature and ability to avoid local optimums, and is known for its high performance in NP-hard problems. Wrapper approach to feature subset evaluation is the most reliable one, despite the computational costs, being the only approach truly reactive to feature's interactions.

However, not only classifier's performance should be considered when choosing between solutions with exact same performance the one using less features is obviously preferred. Moreover, different classifier's applications might require distinct performance metrics. For example, in an classifier design to help in the grading process of an exam, a professor might only want to confirm an A, and therefore only the classifier's performance in that grade is a concern. Another professor might want a more general classifier.

The reasoning behind using several classifier's performance metrics is, therefore, twofold: firstly, it's advantageous to present the decision makers with several objectives, allowing them to prioritize and select based on their preferences. Additionally, using several performance metrics in the search procedure might allow the algorithm to reach better and more diverse solutions.

This work's three main contributions are a deep study on wrapper's performance metrics, studying redundancy and conflicts, followed by a study of whether it's beneficial to use several of those metrics in the randomized feature selection process. Finally, a decision making interface is built to help the user examine and select one or more solutions among all offered.

D. Background

Using evolutionary computation in feature selection dates back to the 80's [6], but only gained popularity in the last decade with the ever-growing size of datasets. The advantages of adding other objectives and using multi-objective algorithm to tackle feature selection have been a topic for some years, with most research consisting of stating feature selection as a bi-objective problem of number of features and some performance metric such as accuracy. This is true both for binary classification [7]–[10] and for multi-class classification [11]. Advantages of using more performance metrics in binary classification are suggested by [12], that showed an increase in performance when using recall and specificity in comparison to using only accuracy. However, that work as not been persued and the advantages of using more metrics were not discussed, despite works such as [13] and [14] using several objectives for binary and multi-class classification, respectively.

II. Subset Evaluation

Traditionally, in the field of feature selection, the classifiers used include decision trees, support vector machines, and k-nearest neighbours [2], [15]. These classifiers do not have inherent tacit feature selection techniques such as neural networks. Decision trees were chosen for exhibiting a great accuracy/computational time trade-off [16]. Nevertheless, a classifier in feature selection needs not have exceptional prediction abilities, but merely to be reactive to the features intrinsic relationships. CART [17] decision tree implementation in MatLab was used.

Different datasets were used both for binary, Table I, and multi-class classification, Table II. All were selected from UCI repository [18], except for AldoA, which is a confidential dataset. These have a wide range of features, instances, types of data and not all are balanced.

TABLE I: Datasets for binary classification.

Name	Features	Instances	% Positives
AldoA	74	1434	89.1
Mushroom	112	8124	48.2
Musk	166	476	43.5
Phishing	68	11055	55.7
Sonar	60	208	46.6
Spectf	44	267	79.4

TABLE II: Datasets for multi-class classification.

Name	Classes	Features	Instances	Unbalaced
DNA	3	180	3186	Yes
Pendigits	10	16	10992	No
Satimage	6	36	6435	Yes
Vehicle	4	18	846	No
Vowel	11	10	990	No

Pre-processing of these datasets included normalizing features between $[-1;1]$, eliminating instances with missing values, expanding categorical variables into several binary

variables, and dividing the data into training and testing sets. This data division was chosen in detriment of cross-validation for its simplicity, considering the amount of subset evaluations to be performed during an evolutionary algorithm. The ratio used was 75% and 25% for training and testing, respectively, keeping the training set balanced, i.e., having an equal number of instances corresponding to each label.

III. Metrics in Wrapper Feature Selection

As mentioned in the introductory section, it's advantageous to present several classifier's performance metrics to the decision makers so they are able to prioritize *a posteriori* the ones they prefer, both for binary and multi-class classification.

Nine metrics were selected for study in binary classification, detailed in Table III. For the multi-class classification problem, the hamming loss function, as defined in Equation 1 was used, where x_P is the predicted labels vector, x_T is the target, and D and L are the number of samples and labels, respectively. Additionally, dividing the problem into several binary problems and calculating each sub-problem's accuracy, as described in [14] was also used. However, considering most binary sub-problems derived from the division of the multi-class problem are very unbalanced, recall for each class was also used.

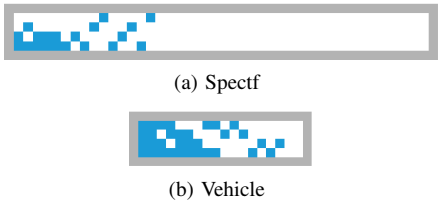
TABLE III: Binary classifier performance metrics.

Name	Formula
Accuracy	$\frac{TP+TB}{TP+FP+FN+TN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
Specificity	$\frac{TN}{TN+FP}$
F1-Score	$\frac{2TP}{2TP+FP+FN}$
Kappa	$1-\frac{1-p_o}{1-p_e}$
NPV	$\frac{TN}{TN+FN}$
Matthews	$\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
Markdness	$Precision + NPV - 1$

$$\text{HammingLoss}(x_P, x_T) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{xor(x_i, y_i)}{|L|}, \qquad (1)$$

However, intuition would suggest that removing unwanted features from a dataset improves a classifier and therefore all its performance metrics. Figure 1 is a binary map for Spectf and Vehicle datasets, where each column represents a feature and a line a solution. Blue squares indicate selected features. The solutions presented are the ones that maximize the metrics mentioned above, of a universe of 30000 randomly generated solutions, after eliminating duplicate solutions.

It's evident in Figure 1 that different metrics prefer totally distinct feature subsets, further justifying the need to present several wrapper's performance metrics to the decision maker. Additionally, these results encourage the application of multi/many-objective optimization in feature selection.



(a) Spectf

(b) Vehicle

Fig. 1: Best solutions' subsets for Spectf and Vehicle datasets.

There's a need to study the relationship between metrics, not only for presentation purposes but also because it's advantageous for multi-objective algorithms since redundancy might decrease an algorithm's performance [19]. The relationships are pairwise, and can be of three types: independent, if objectives are autonomous in their optimization, i.e., optimizing one does not affect the other on; harmonious if objectives are improved or deteriorated simultaneously; and conflictual if enhancing one objective the other is imperatively damaged as a side-effect.

Independent objectives can be optimized separately and harmonious objectives are redundant and therefore only one of them must be optimized. Only conflictual objectives truly justify the usage of multi-objective algorithms, since they result in a trade-off curve.

To identify these relationships, a quantitative analysis using parallel coordinates plot was used. The reasoning is that conflictual objectives have lots of crossings in a parallel coordinate plot, while independent or harmonious have very few. In Figure 2 four different solutions are displayed in a parallel coordinate plot. While objectives 2 and 3 have exactly the same solution ordering and therefore are harmonious, solution 1 and 5 only discrepancy is the ordering of the blue and grey solutions. These should also be, to a certain degree, considered conflictual.

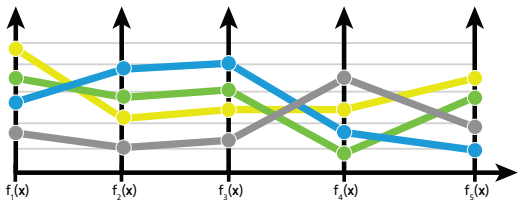


Fig. 2: Parallel coordinates plot.

For the end of quantifying the degree of conflict based on the ordering of solutions, Kendall's correlation was used, as suggested. Benchmark problems such as DTLZ were tested to get a threshold value above which two objectives are considered harmonic. The attained value is $K_c = 0.3$. Thirty thousand randomly generated solutions for each dataset were used, which in most of them represents a very small part of the search space. The pairwise Kendall's correlation was calculated and the averaged results for all binary datasets using the metrics in Table III are in Figure 3.

The results are dataset dependent, but from the averaged

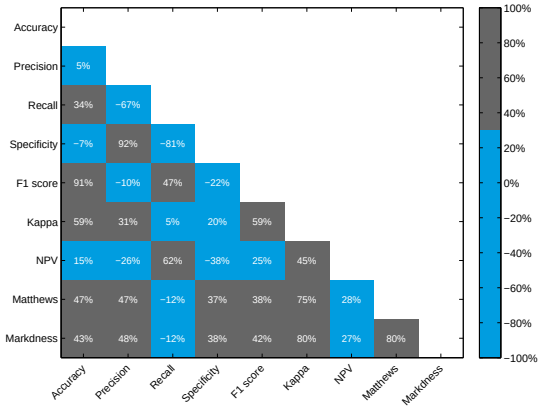


Fig. 3: Kendall correlation averaged.

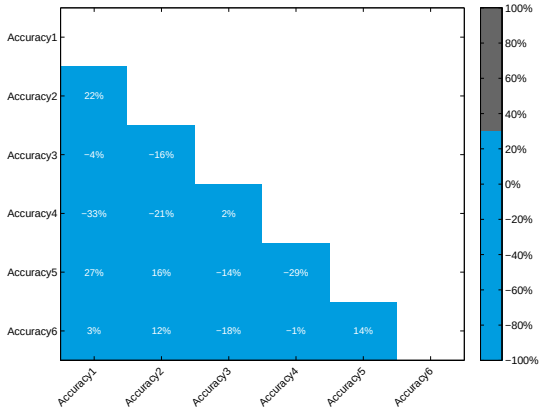


Fig. 4: Kendall correlation averaged.

TABLE IV: Metrics used in binary classification.

Nr. of Metrics	Perfomance Metrics
2	Accuracy
3	Recall, Specificity
6	Accuracy , Recall, Specificity, Kappa, NPV
10	All

heatmap some conclusion can be made. Foremost, the most conflictual objectives are recall and specificity. Further analysis can be made not by finding the most conflictual but the most redundant. For example, markedness and Matthews coefficient are made redundant by most other metrics. F1-score is highly correlated to accuracy, and therefore is eliminated, precision is also made redundant by specificity. NPV is not eliminated, despite its high correlation value, because it is conflictual in some datasets. Table IV summarizes the combinations of metrics chosen for binary classification to analyse in multi-objective optimization.

The same analysis was done for multi-class classification using accuracy for each class, with results showing a high degree of conflict between objectives. Figure 4 shows this for the Satimage dataset, but same results were obtained for the remaining multi-class datasets and also using recall for each class.

Both results encourage the use of multi-objective algorithms in feature selection, despite some objective redundancy in binary classification.

IV. Search Procedure

Multi-objective problem formulation is defined in Equation 2, with contraints g_n and h_k. The first difference to single-objective optimization is the existence of two spaces: design space where the variables $[x_1, x_2, ..., x_N]$ are changed, and the objective space which yields the result $[f_1, f_2, ..., f_M]$ [20].

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & F(x) = \{f_1(x), f_2(x), ..., f_m(x)\} \\ \text{subject to} \quad & g_n(x) \leq 0, n = 1, 2, ..., J, \\ & h_k(x) = 0, k = 1, 2, ..., K \end{aligned} \tag{2}$$

Moreover, the global optimal concept of single-objective optimization is replaced by Pareto optimality. In a bi-objective problem, if a solution is better in one objective but worse in the other one, no solution is worse nor better. Instead, these solutions are non-dominated and together form a trade-off. Formally, a solution A strictly dominates solution B, when there's at least one objective in which A is better while being no worse in all others [21]. Mathematically $A \preceq B$, if these two conditions are met:

1) $\forall i \in \{1, 2, ..., k\} : f_i(A) \leqslant f_i(B)$
2) $\exists j \in \{1, 2, ..., k\} : f_j(A) < f_j(B)$

This is defined as weak Pareto dominance, while strong Pareto dominance ($A \prec B$) requires all objectives to be better. The Pareto front is defined as the ideal trade-off curve.

Considering several objectives in an optimization procedure is a relatively established field. The most simple methods do so *a priori*, that require preference information before the algorithm's start and usually combines the objectives or creates several sub-problems to apply single objective optimization. These include weighted sum and lexicographic ordering. Interactive methods require the decision maker to input preference during the searching procedure. *A posteriori* methods only require information after the searching procedure, finding several solutions that are then filtered according to user's preference. This is obviously the ideal kind of methods, for not limiting the search space nor having user preference dependency.

A posteriori methods are heavily based on single-objective optimization ones, differing in the selection mechanism, which can be either based on Pareto dominance, decomposition or indicator. Of the three, Pareto dominance based are undoubtedly the most well known and used. However, these are well known for their scalability issues. Increasing the number of objectives escalates the percentage of non-dominated solutions in the population, weakening the selection pressure to the Pareto front. Pareto based approaches convergence heavily depends on the dominated part of the population, and therefore have their performance severely deteriorated [22]. Figure 5 illustrates the increasing ratio of non-dominated solutions in a population as the number of objectives increases, and also

shows that increasing the population size is an effective as a remedy, although computationally expensive.

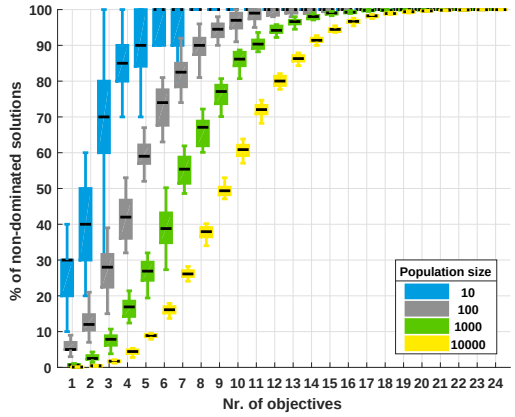


Fig. 5: Ratio of non dominated solutions depending on the number of objectives and population size, using numerous random generated vectors.

This phenomena results in denominating problems with more than 4/5 objectives many-objective problems [23].

V. Methods

Considering the previous analysis of wrapper performance metrics relationships, the combinations considered promising range from 2 objectives up to 10 in binary classification, and in multi-class classification these depend on dataset's number of classes, ranging from 4 objectives up to 12. Therefore both multi and many-objective problems are being considered. Keeping this in mind and the current surveys in this fields [23]–[25], five different methods were chosen.

1) **NSGA-II** [26]
2) **NSGA-III** [27]
3) **MOEA/D** [28]
4) **HypE** [29]
5) **PICEA-g** [30]

NSGA-II is the most well-known algorithm for multi-objective optimization, for its simplicity and for lacking extra parameters. It's a Pareto based method and therefore suffers from the scalability issues mentioned above. As a response to those scalability issues, the same authors created NSGA-III, a decomposition based algorithm which has shown great results for many-objective problems. However, not always does the NSGA-III perform better than its ancestor, particularly in the many-objective knapsack problem [31], which is discrete and binary-coded, and therefore somewhat similar to feature selection. MOEA/D, another decomposition based algorithm that creates several sub-problems, has also become a benchmark problem for multi-objective optimization, with [32] showing it frequently outperforms NSGA-II. Additionally, the work in [30] compared several algorithms (NSGA-III was still unborn) and found that PICEA-g and HypE had better performance, the first being also decomposition based, in which there's simultaneous evolution of a solutions and a goals population, and the second is indicator based.

All these algorithms involve a population and therefore a single run is capable of providing a non-dominated front to the decision maker. PlatEmo [33] was used for the algorithm's implementation in MatLab environment.

A. Algorithms Performance Indexes

A multi-objective optimization algorithm should be evaluated according not only to convergence, but also on its ability to provide diverse solutions. Consider solutions A and B for the bi-objective problem illustrated in Figure 6. Solution B is well converged but its solutions are poorly spread. For that reason three different indexes were used.

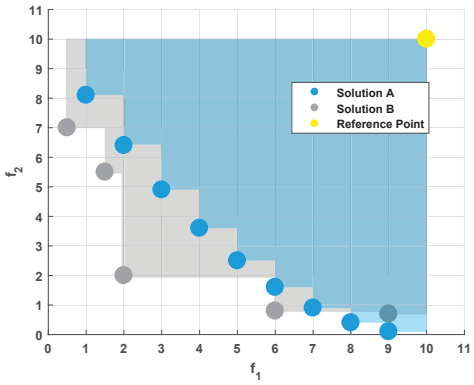


Fig. 6: Two example solutions for a bi-objective problem.

Set Coverage: Set coverage is a pairwise comparison between populations convergence. $C(A,B)$, as defined in Equation 3, computes the ratio of population in B dominated by any solution in A. This index is not symmetrical, therefore both $C(A,B)$ and $C(B,A)$ should be calculated.

$$C(A,B) = \frac{|\{b \in B; \exists a \in A : a \succeq b\}|}{|B|} \tag{3}$$

Spacing: Spacing [34], S, is a measure of standard deviation between solution's distance, as described in Equation 4, where $d_i(A) = min_{k \in A \wedge k \neq i} \sum_{m=1}^{M} \left| f_m^i - f_m^k \right|$ and $\overline{d(A)} = \sum_{i=1}^{|A|} \frac{d_i}{|A|}$. When the solutions are uniformly spaced, this metric is small, since the distance vectors are similar. Therefore, unlike set coverage, the intention is to minimize it.

$$S(A) = \sqrt{\frac{1}{|A|} \sum_{i=1}^{|A|} \left(d_i - \overline{d}\right)^2} \tag{4}$$

Hypervolume: Finally, the well known hypervolume metric can measure both convergence and diversity. It calculates the area/volume created by the non dominated front in the objective space, given an reference point. High value of hypervolume is synonym of a well-spread and/or converged solution. Hypervolume computation is simple in a bi-objective problem, but with increasing number of objectives become highly demanding.

Table V presents the three metrics values of solutions A and B of Figure 6.

TABLE V: Indices results to the solutions presented in Figure 6

Solution	Hypervolume	Spacing	Set Coverage
A	350.50	0.51	$1/5 = 0.20$
B	362.15	0.61	$7/9 \approx 0.78$

B. Benchmark Comparison

To validate the algorithms benchmarks problems were used. Of DTLZ test suit [35], DTLZ4 and DTLZ7 were chosen, defying diversity and convergence, respectively. Additionally, the multi-objective knapsack problem was also used, due to its similarity to feature selection, being both discrete and binary-coded. For each benchmark problem $M \in \{2, 4, 10\}$ were used. The algorithms are stochastic and therefore a variance analysis is done using box plots. DTLZ4 and DTLZ7 had the expected performance. NSGA-II performed the best when dealing with $M = 2$ but increasing the number of objectives the best algorithms tend to be HypE, NSGAIII, and PICEA-g. Moreover, HypE showed high performance across all objectives, similarly to PICEA-g.

The behaviour for the multi-objective knapsack problem was different, and due to its similarities to feature selection problem, namely for being discrete and binary-coded, only these results are shown here. Figure 7 shows the hypervolume performance, after performing the min-max normalization described in Equation 5 in relation to each objective.

$$x_i' = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{5}$$

It's visible that for $M = 2$, MOEA/D performs very well, closely followed by NSGA-II and HypE. Increasing the number of objectives, HypE becomes the top-performer. NSGA-III is inferior across all range of objectives, likely explained by its decomposition mechanism based on normalisation not being suited for discrete problems.

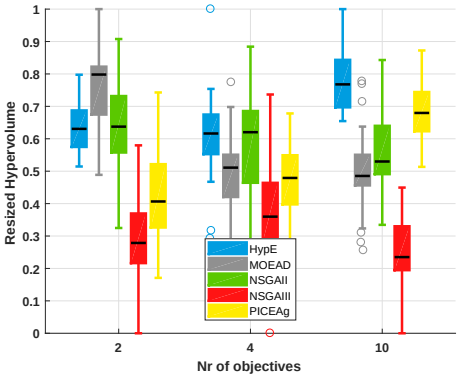


Fig. 7: Hypervolume comparison for MOKP.

Spacing analysis, in Figure 8, shows that MOEA/D performs the best in terms of solution's uniformity, except when $M = 10$, where HypE outperforms it. Nevertheless, the results are very similar for all algorithms, except NSGA-II which has very poor uniformity in the highest dimensional example.

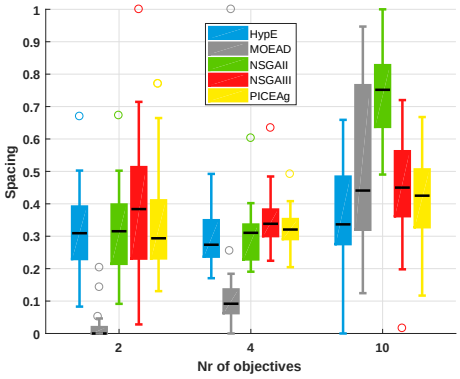


Fig. 8: Spacing comparison for MOKP.

The results of set coverage, being pairwise, are of harder visualization. Keeping in mind that each bar represents the domination of that bar's respective algorithm, it's visible that the results are similar to the hypervolume's. for $M \in [2, 4]$ MOEA/D is the top-performer. However, when $M = 10$, HypE performs slightly better. This coincides with the results in [36].

C. Encoding for Feature Selection

Initial Population, Size and Stopping Criteria

The initial population is a fundamental parameter in an evolutionary algorithm. In order to motivate genetic operators to reach solutions with a low number of features in the first generations, the method used consisted, for each individual, in generating a uniformly random number $K =\in [1; N_{feat}]$ and then uniformly choosing K features to be selected. This ensures a uniformly spread amount of subset sizes.

Population size was set to 100, to both allow diverse populations and also keep the final population and computational time manageable. Number of generations was set as the stopping criteria.

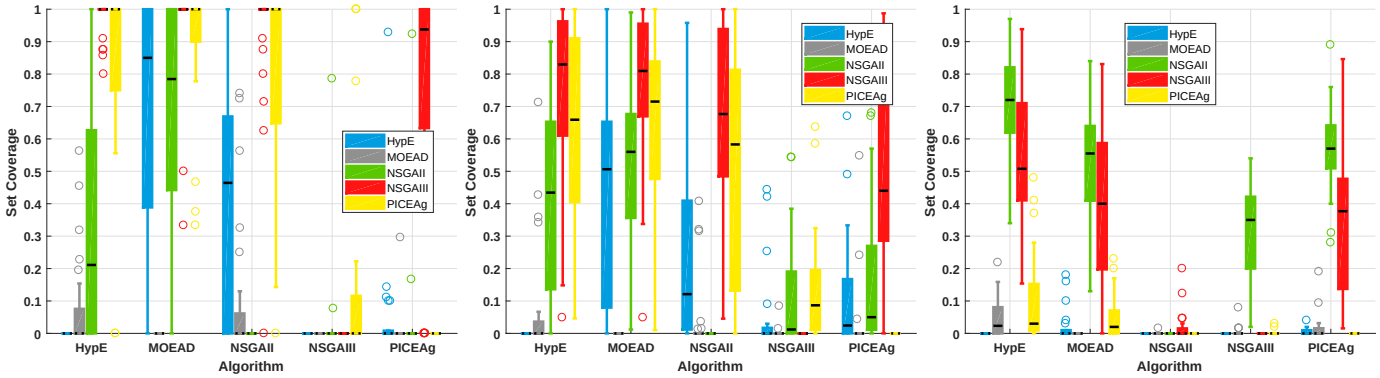
Genetic Operators

All chosen algorithms are genetic based, meaning they use the typical selection, crossover and mutation operators. Only NSGA-II and HypE use non-random selection to choose the parents, and both use binary tournament. The first with rank level and crowding distance as second criteria, and HypE with hypervolume-based fitness value of each solution. So they have no extra parameters.

For the remaining genetic operators, two-point crossover was used and bitwise mutation, with a mutation probability of $p_m = \frac{1}{L}$, was selected.

Constraint Handling

Only two simple constraints exist in feature selection: 1- each solution is a binary vector; 2- at least one feature must be included. The first is not infringed using the genetic operators mentioned, and the second is handled by simply assigning zero performance and maximum number of features.



(a) MOKP 2 objectives (b) MOKP 4 objectives (c) MOKP 10 objectives

Fig. 9: Set coverage behaviour of the algorithms.

Algorithms Parameters

Only NSGA-II and NSGA-III are parameter-free, except for the usual genetic operators. MOEA/D, HypE and PICEA-g all have external parameters. MOEA/D's parameter is the number of neighbours, i.e., the number of close sub-problems sharing information. HypE is characterized by the number of samples used to estimate the hypervolume metric, and PICEA-g uses the goal's population size. All these were tested using time, hypervolume and spacing analysis, for the MOKP.

As the number of neighbours increases in MOEA/D, so did the computational time in a tiny amount, and the spacing metric decreased. Hypervolume seemed to remain independent of the number of neighbours. The decomposition function for MOEA/D being used is Tchebycheff's [28].

In both HypE and PICEA-g the main change of the number of samples and of goals, respectively, is the computational time, which increases exponentially. Hypervolume slightly increased, and no significant differences in spacing were found.

Considering all this and the suggestions in the literature, the parameters listed in Table VI were selected.

TABLE VI: Parameters values

Algorithm	Parameter Name	Value
HypE	Nr. of Samples	1000
MOEA/D	Nr. of Neighbours	10
NSGA II	-	-
NSGA III	-	-
PICEA-g	Nr. of Goals	5000

VI. Results

Considering that HypE performed consistently very well in the multi-objective knapsack problem across all objectives, it was chosen to test if it's beneficial to use several wrapper's performance metrics in feature selection, i.e., if the algorithm using more metrics reaches better solutions both in terms of convergence and diversity.

A. Binary Classification

The four sets of wrapper performance metrics listed in Table IV were tested in all datasets using 100 generations composed of 100 individuals, repeated 20 times. The results are similar in all datasets, and Figure 11 illustrates Spectf dataset's performance in terms of hypervolume, spacing and set coverage. It's visible that using only accuracy yields poor performance in all metrics. Moreover, despite $M = 6$ presenting slightly better hypervolume, both spacing and set coverage analysis show that the population resulting from using HypE with $M = 10$ is more uniformly spread and has overall better solutions than $M = 6$.

To visualize these results, the populations resulting from using mRMR [37], $M = 2$ and $M = 10$ are compared in Figure 10, with evident better performance for $M = 10$, reaching "peak solutions", i.e., solutions that have very high performance in one objective while being only average in others.

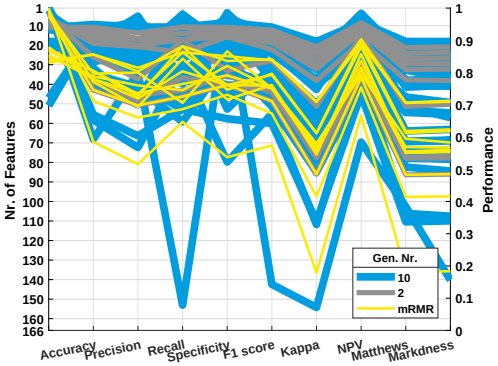
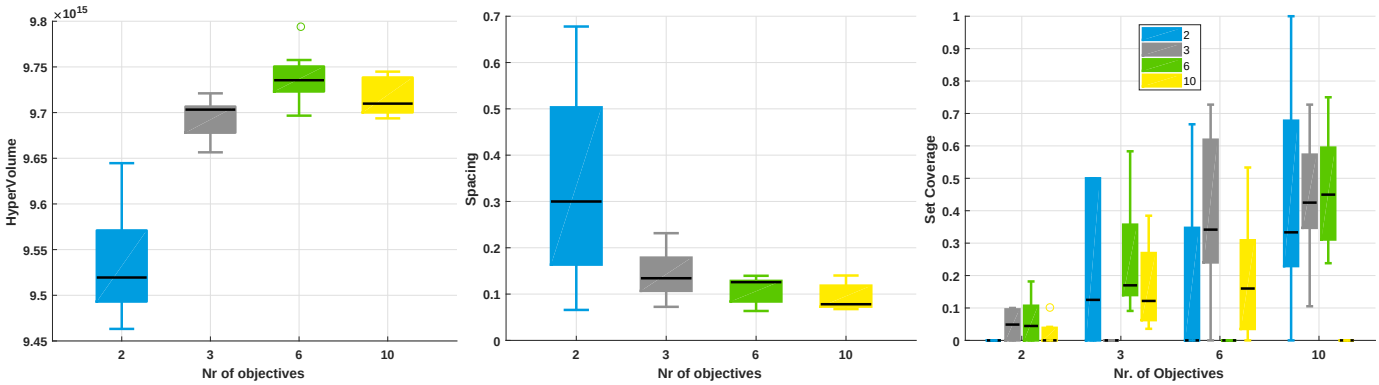


Fig. 10: Musk dataset binary classification feature selection with three different approaches.

Further study showed that NSGA-II is the algorithm that performs the best in binary classification feature selection using $M = 10$, closely followed by HypE. NSGA-II's performance might be explained by the high level of redundancy



(a) Hypervolume (b) Spacing (c) Set Coverage

Fig. 11: HypE performance in binary classification using Spectf dataset.

between objectives, as shown in Figure 3, which lowers to value of true conflictual objectives.

B. Multi-class Classification

The same analysis was done in multi-class classification datasets, using HypE for 100 generations, each composed of 100 individuals in 20 repetitions. The results are also similar across datasets and are illustrated by Vowel in Figure 13. The results suggest there's no major advantage in dividing the problem into binary subproblems considering spacing and set coverage metrics, but hypervolume shows improvement. This result suggests that the population resulting from using accuracy in each class has a wider presence in the search space.

Similarly to what was done before, Figure 12 compares mRMR, $M = 2$ and accuracy to each class in multi-class classification. It's visible that using $M = 2$ and accuracy for each class have similar performance, but the latter yields a more diverse set of solutions.

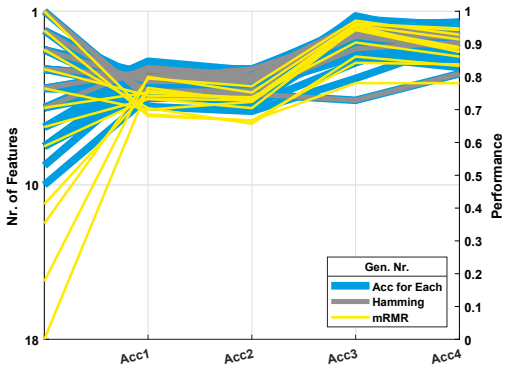


Fig. 12: Vehicle dataset multi-class classification feature selection with three different approaches.

Further study suggested HypE to be more adequate for multi-class feature selection, with PICEA-g as a second choice. NSGA-II was able to find a diverse set of solutions but with very poor convergence.

VII. Decision Interface

Despite the advantages in using several objectives to guide the feature selection search procedure into better solutions, this produces a high number of different non-dominated solutions, making the selection process similar to "finding a needle in a haystack". For that reason, and entering the multi-criteria decision making field, a decision interface was created in MatLab, which allows the user to set minimum values for each objective, observe both the search space and the design space simultaneously, and aids in the selection by finding the solution which shows higher weighted sum when the weight vector is specified by the user.

VIII. Conclusion

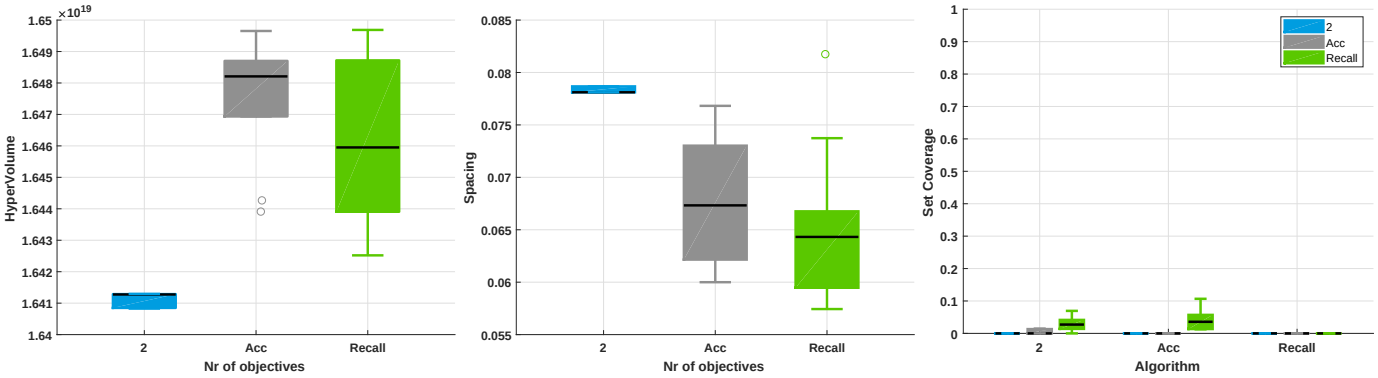
This work's investigation is a contribution to the necessarily growing field of feature selection, in an age where data acquisition and storage is done effortlessly.

In addition to a deep study of the relationships between wrapper's performance metrics, where conflictual set of metrics were selected, this study has shown great advantages to the arduous feature selection task when using many objectives, not only reaching better solutions, i.e., well-converged, but also yielding a wider set of solutions, capable of describing all search space and providing many different solutions to offer to the decision maker.

The result of that process is a big number of trade-off solutions for the decision makers to select. To help in this process, a decision interface was created to support the decision makers find the solutions that are most suited for their needs/priorities.

With this work, research is surely motivated. Not only could these results be tested and verified using other classifiers such as k-NN or SVM, but also other, more complex datasets. Additionally, other metrics could be used, along with other multi-objective algorithms, in order to find the perfect combination. Regression problems can be discussed, although possibly not as many metrics are non-redundant.

Adding flexibility to the decision interface, such as fuzzy decision making, could also make it an indispensable tool for any feature selection process.



(a) Hypervolume (b) Spacing (c) Set Coverage

Fig. 13: Multi-class classification HypE performance with different set of wrapper metrics in Vowel dataset.

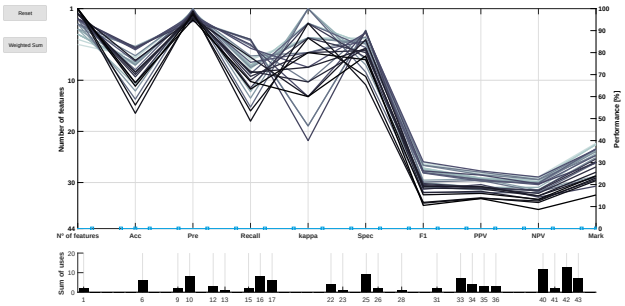


Fig. 14: Created decision interface.

REFERENCES

[1] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, vol. 10, no. 5, pp. 335–347, 1989.

[2] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research (JMLR)*, vol. 3, no. 3, pp. 1157–1182, 2003.

[3] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, no. 1-2, pp. 237–260, 1998.

[4] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[5] K. Kira and L. Rendell, "The feature selection problem: Traditional methods and a new algorithm," *Aaai*, pp. 129 – 134, 1992.

[6] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, vol. 10, no. 5, pp. 335 – 347, 1989.

[7] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition," *Proc. Int. Conf. on Pattern Recognition*, vol. 1, pp. 568–571, 2002.

[8] G. Pappa, A. Freitas, and C. A. Kaestner, "A Multiobjective Genetic Algorithm for Attribute Selection," *Proc. of the Fourth International Conference on Recent Advances in Soft Computing*, pp. 116–121, 2002.

[9] S. M. Vieira, M. C. Sousa, and T. A. Runkler, "Multi-Criteria Ant Feature Selection Using Fuzzy Classifiers," *Swarm Intelligence for Multi-objective Problems*, pp. 19–36, 2009.

[10] Y. Zhang, D.-w. Gong, and J. Cheng, "Multi-objective Particle Swarm Optimization Approach for Cost-based Feature Selection in Classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. PP, no. c, pp. 1–13, 2015.

[11] Y. Zhang, D.-w. Gong, X.-y. Sun, and Y.-n. Guo, "OPEN A PSO-based multi-objective multi- label feature selection method in classification," *Scientific Reports*, no. August 2016, pp. 1–12, 2017.

[12] J. Garcia-Nieto, E. Alba, L. Jourdan, and E. Talbi, "Sensitivity and specificity based multiobjective approach for feature selection: Application to cancer diagnosis," *Information Processing Letters*, vol. 109, no. 16, pp. 887–896, 2009.

[13] M. Pal and S. Bandyopadhyay, "Many-objective Feature Selection for Motor Imagery EEG Signals using Differential Evolution and Support Vector Machine," 2016.

[14] A. Khan and A. R. Baig, "Multi-Objective Feature Subset Selection using Non-dominated Sorting Genetic Algorithm," *Journal of Applied Research and Technology*, vol. 13, no. 1, pp. 145–159, 2015.

[15] F. Jimenez, E. Marzano, G. Sanchez, G. Sciavicco, and N. Vitacolonna, "Attribute selection via multi-objective evolutionary computation applied to multi-skill contact center data classification," *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, vol. 07821, pp. 488–495, 2016.

[16] B. Huang, B. Buckley, and T. M. Kechadi, "Multi-objective feature selection by using NSGA-II for customer churn prediction in telecommunications," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3638–3646, 2010.

[17] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, "Classification and regression trees," vol. 1, no. February, p. 368, 1984.

[18] M. Lichman, "UCI Machine Learning Repository," 2013.

[19] R. C. Purshouse and P. J. Fleming, "Conflict, Harmony, and Independence: Relationships in EvolutionaryMulti-Criterion Optimisation," vol. 4403, pp. 388–402, 2007.

[20] C. A. Coello Coello, G. B. Lamont, and D. a. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2007.

[21] K. Deb, "Multi-Objective Optimization Using Evolutionary Algorithms," p. 497, 2001.

[22] H. Ishibuchi, N. Tsukamoto, Y. Hitotsuyanagi, and Y. Nojima, "Effectiveness of scalability improvement attempts on the performance of NSGA-II for many-objective problems," *Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08*, p. 649, 2008.

[23] S. Chand and M. Wagner, "Evolutionary many-objective optimization: A quick-start guide," *Surveys in Operations Research and Management Science*, vol. 20, no. 2, pp. 35–42, 2015.

[24] C. Von Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707–756, 2014.

[25] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

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

[27] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting

Approach, Part I: Solving Problems with Box Constraints," *Ieeexplore.Ieee.Org*, vol. 18, no. c, pp. 1–1, 2013.
[28] Q. Zhang, S. Member, and H. Li, "MOEA / D : A Multiobjective Evolutionary Algorithm Based on Decomposition," vol. 11, no. 6, pp. 712–731, 2007.
[29] J. Bader and E. Zitzler, "HypE: an algorithm for fast hypervolume-based many-objective optimization." *Evolutionary computation*, vol. 19, no. 1, pp. 45–76, 2011.
[30] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-Inspired Coevolutionary Algorithms for Many-Objective Optimization," vol. 17, no. 4, pp. 474–494, 2013.
[31] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of NSGA-II and NSGA-III on various many-objective test problems," *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, pp. 3045–3052, 2016.
[32] H. Li and Q. Zhang, "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.
[33] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization," pp. 1–20, 2017.
[34] J. R. J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," 1995.
[35] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, A. Abraham, L. Jain, and R. Goldberg, "Scalable test problems for evolutionary multiobjective optimization," *Evolutionary Multiobjective*, no. 1990, pp. 1–27, 2001.
[36] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Evolutionary Many-Objective Optimization by NSGA-II and MOEA / D with Large Populations," *Optimization*, vol. 1, no. October, pp. 1758–1763, 2009.
[37] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.