Dynamic Genotype Reduction for Narrowing the Feature Selection Search Space

Sašo Karakatič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Koroška cesta 46, 2000 Maribor
Email: saso.karakatic@um.si

Iztok Fister Jr.
Faculty of Electrical Engineering and Computer Science
University of Maribor
Koroška cesta 46, 2000 Maribor

Dušan Fister
Faculty of Economics and Business
University of Maribor
Razlagova ulica 14, 2000 Maribor

***Abstract*—Large search space optimization problems can pose a challenge to any optimization approach, as there are many chances for the algorithm to become stuck in the local optima. In this paper, we present a novel dynamic representation of genotype, where the search areas with low potential are discarded from the search, and areas of the more significant potential for global optimum are kept in focus. The validity of the proposed method is tested on a large scale feature selection problem, using an extensive dataset with a large number of features compared to instances. An arrhythmia dataset was specifically chosen for a case study, and a self-adaptive differential evolution algorithm with the dynamic genotype reduction was implemented. The results of the experiments are promising for the future since the proposed method achieves better results than standard feature selection with stochastic population-based nature-inspired algorithms.**

I. Introduction

Feature Selection (FS) is a computational process intended to reduce/diminish the dimensionality of data. FS is a standard step of Data Mining (DM) method (often machine learning), which is a part of a Knowledge Discovery in Databases (KDD) procedure. FS aims to select a subset of data features, which performs better at the data mining problem, e.g., classification, either because: (1) to achieve faster completion of the algorithm, or (2) to eliminate data redundancy. Several variants of FS exist, among most common are: filter-based, wrapper-based and embedded-based. Filter-based perform FS separately from DM, while wrapper-based combine FS and DM. Embedded-based variants combine characteristics of both mentioned variants. In this paper, the second variant is discussed, which means that the evaluation of FS is intrinsically connected to the DM method, e.g., artificial neural network (ANN), random forests (RF), k-nearest neighbors, or others.

Evolutionary and swarm-based algorithms in the FS domain are gaining their popularity in recent years, and literature on this subject is also arising. Karakatič [1] has proposed an EvoPreprocess FS framework that bases on meta-heuristic and population-based nature-inspired algorithms (NIA) from NiaPy [2]. The author has shown that the framework can deal with high-dimensional and imbalanced datasets. Fister et al. [3] have proposed an FS using the so-called threshold mechanism, where a population-based NIA handles the attendance of features in a test subset. Authors in [4] show FS using the k-nearest neighbor algorithm and [5] show the FS on a large, complex dataset. Authors in paper [6] research the stability of filter- and wrapper-based FS procedures.

We define the FS problem as a wrapper-based, NIA optimized dynamically changing process that reduces/shortens (eliminates) irrelevant features simultaneously with the evolution. Whether irrelevant or redundant features exist in the dataset, it is more effective to eliminate them instantly than to keep them in a potential search space for forthcoming trial solutions. Since the features that are eliminated cannot be restored into potential search space anymore, such a solution demands the implementation of dynamically reducing gene representation.

Our approach thus differs from the mentioned literature twofold. First, we implement the dynamically reducing gene representation to eliminate redundant, low-variance, inexpensive or irrelevant features completely during the online execution of the optimization algorithm, and second, we attain the feature selection search space to narrow accordingly, keeping the more expensive features for further space search and at the same time decreasing the difficulty of the problem. Both of the novelties in the paper are hypothesized to increase classification results and reduce computation times.

The remainder of this paper is structured as follows. Section II outlines the fundamentals of methods used in this paper, while Section III presents the proposed method. Section IV is devoted to the experiments and results. Paper is summarised with the conclusions in Section V.

II. Fundamentals

Wide datasets, i.e., many features and a low number of instances, are commonly referred to have a negative effect in extracting patterns, e.g., in classification, as there are many options to consider (many features) with a low amount of variance (low observation count). In this case, extracting patterns from data can be extremely burdensome. Here, a compelling expression named “Curse of Dimensionality” was introduced by Bellman [7] to describe the problem of distinguishing the noise from useful data.

A. Arrhythmia dataset

Wide dataset problem is prevalent in the dataset used in this paper, called Arrhythmia, an example of a multivariate dataset intended to detect the presence of cardiac arrhythmia disease. In this dataset, 279 features, with values ranging from negative to positive integer and floating-point values, are supplied. The 452 instances of the dataset are classified into 16 classes, with the majority falling into the "normal" class and the rest incorporating any cardiac disorders. Some essential characteristics of the dataset are shown in Tab. I; interested readers are invited to read original website[1] for a detailed explanation of dataset.

TABLE I: Arrhythmia dataset details.

No.	Feature	No.	Feature
1.	Age	*Of channels**	
2.	Gender	28.-39.	DII
3.	Height	40.-51.	DIII
4.	Weight	52.-63.	AVR
5.	QRS duration	64.-75.	AVL
6.	P-R interval	76.-87.	AVF
7.	Q-T interval	88.-99.	V1
8.	T interval	100.-111.	V2
9.	P interval	112.-123.	V3
Vector angles		124.-135.	V4
10.	QRS	136.-147.	V5
11.	T	136.-147.	V5
12.	P	148.-159.	V6
13.	QRST	*Of channel DI*	
14.	J	160.	JJ wave
15.	Heart rate	161.	Q wave
Of channel DI		162.	R wave
16.	Q wave average width	163.	S wave
17.	R wave average width	164.	R' wave
18.	S wave average width	165.	S' wave
19.	R' wave average width	166.	P wave
20.	S' wave average width	167.	T wave
21.	No. of intrinsic deflections	168.	QRSA
22.	Existence of ragged R wave	169.	QRSTA
23.	Existence of diphasic derivation of R wave	*Of channels***	
		170.-179.	DII
24.	Existence of ragged P wave	180.-189.	DIII
25.	Existence of diphasic derivation of P wave	190. 199.	AVR
		200.-209.	AVL
26.	Existence of ragged T wave	210. 219.	AVF
27.	Existence of diphasic derivation of T wave	220.-279.	V1, V2, V3 V4, V5, V6

Note: "*" means similar as for 16.-27. features and "**" means similar as 160.-169. features.
Source: UCI Machine Learning Repository.

B. Self-Adaptive Differential Evolution

Differential evolution (DE) [8] belongs to a class of evolutionary algorithms and is considered a powerful and efficient method for solving global optimization tasks. DE is a population-based NIA that is composed of differential mutation, crossover, and selection operators. Although DE is among the most popular NIAs, its efficiency of searching through search space depends heavily on suitable parameter control. Finding a suitable combination of control parameter settings is a very long-lasting process. Fortunately, some hybridized DE variants with a self-adaptation of control parameters during the optimization run were proposed, e.g., one of the most efficient variants is jDE that was proposed in 2006 by Brest et al. [9]. Here, each individual is extended with scale factor F and crossover rate CR that undergo the variation operators, mathematically represented as follows:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, ..., x_{i,M}^{(t)}, F_i^{(t)}, CR_i^{(t)}). \tag{1}$$

In jDE, both parameters are modified according to the following equations:

$$F_i^{(t+1)} = \begin{cases} F_l + \text{rand}_1 \cdot (F_u - F_l) & \text{if } \text{rand}_2 < \tau_1, \\ F_i^{(t)} & \text{otherwise}, \end{cases} \tag{2}$$

and

$$CR_i^{(t+1)} = \begin{cases} \text{rand}_3 & \text{if } \text{rand}_4 < \tau_2, \\ CR_i^{(t)} & \text{otherwise}, \end{cases} \tag{3}$$

where $\text{rand}_{i=1...4} \in [0,1]$ are randomly generated values drawn from uniform distribution in interval $[0,1]$, τ_1 and τ_2 are learning rates, F_l and F_u lower and upper bound for parameter F, respectively.

III. Proposed Approach

In literature, the FS procedure is commonly applied to remove any redundant features and thus mitigate the curse of dimensionality problem before the pattern extraction process. However, large search space problem can also pose significant difficulties to FS algorithms, i.e. optimization algorithms, since increasing the search space may also incorporate numerous local optima. With the arise of local optima, chances of becoming stuck arise as well and chances the optimization algorithm to continue the search towards the global optimum diminish. NIA wrapper-based FSs [10]–[12] commonly suffer from a large search space problem, where the larger the feature set, the wider the search space.

To cope with the mentioned problem sufficiently, we propose a dynamic genotype reduction, using which we achieve dynamic search space narrowing. Here, the optimization algorithm's genotype representation is automatically reduced periodically. We expect that this search space narrowing might (1) potentially improve the optimization process, or make it less difficult and (2) subsequently shorten the computation time needed for the search toward the global optimum. The proposed method exploits the standard wrapper-based FS procedure, optimized with NIA optimization methods from EvoPreprocess, with periodic dynamic genotype reductions.

A. Genotype representation

Typically, the NIA representation of features in the wrapper-based FS algorithm consists of a fixed-length array genotype, where each gene covers precisely one feature of the dataset (number of genes equal to the number of features). Genotypes can be either (1) boolean values, where 1 represents the attendance of i-th feature in a subset and 0 represents the absence, or (2) real (floating-point) decimal values, usually limited within the interval $(0,1)$, which are transformed into

[1] https://archive.ics.uci.edu/ml/datasets/Arrhythmia

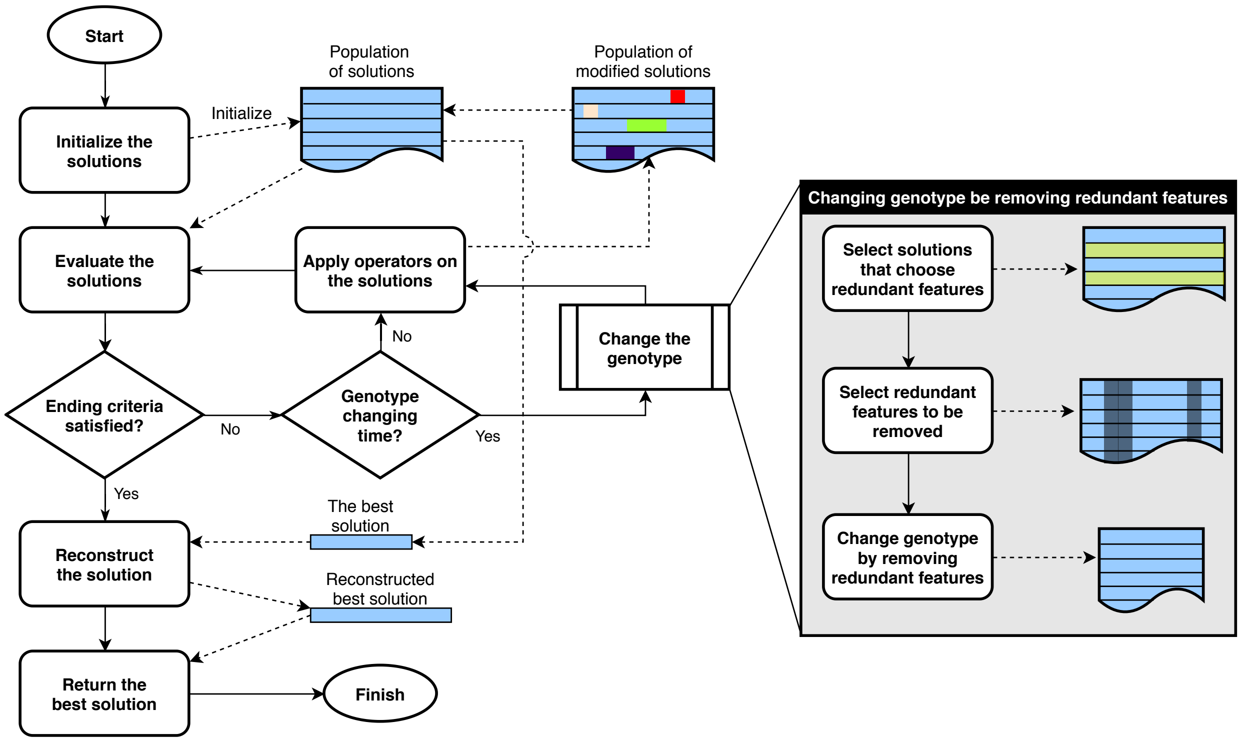
Start
Initialize the solutions
Initialize
Population of solutions
Population of modified solutions
Evaluate the solutions
Apply operators on the solutions
No
Ending criteria satisfied?
No
Genotype changing time?
Yes
Change the genotype
Yes
The best solution
Reconstruct the solution
Reconstructed best solution
Return the best solution
Finish
Changing genotype be removing redundant features
Select solutions that choose redundant features
Select redundant features to be removed
Change genotype by removing redundant features

Fig. 1: The optimization process for feature selection with changing genotype. Source: Authors.

the boolean form using a so-called mapping function, e.g., fixed thresholding at threshold T, to effectively designate the attendance or absence of the feature in a subset. The proposed methodology in this paper follows the second option, i.e., real-value representation of features and subsequent mapping.

With each reduction cut, the most irrelevant parts of the genotype are removed, and the optimization continues on the remaining solutions. Irrelevant genes are removed, but the sequence and the values of the remaining genes stay unchanged. A brief graphical demonstration is provided on the right side of Fig. 1 as follows: First, random trial solutions are initialized, bearing in mind that the problem's dimension follows the number of features in a dataset. Next, the optimization process using dynamic genotype reduction is started, where the jDE operators are applied to dynamically change the genotype, and the trial solutions are evaluated against the fitness function f accordingly. After each optimization generation, a simple test to verify whether (1) the dynamic genotype change needs to be executed, or (2) ending criteria are reached, is performed. It is worth mentioning that the genotype changing is only carried out when the number of generations equals the pre-defined genotype changing margins $M = \{m_1, m_2, ..., m_n\}$. Here, n represents the number of genotype cuts. After the complete optimization process, the best solution is finally reconstructed and returned to the user.

B. Dynamic genotype reductions

The dynamic genotype reductions are executed as follows: first, the W best performing solutions are selected, which are in the next step exploited to vote the F worst features (both the W best performing solutions and F least performing solutions are control parameters). The voting rule here is set as: whether the feature is present in a particular solution, it gets a vote from that solutions. In this way, W best solutions give votes to those features that are present in those solutions. Features with the least votes then represent the features that are least common in best performing feature sets and are considered as the features that make the FS worse off. These are removed from the further optimization process instantly. By removing the less relevant features the genotype of the optimization process reduces and further optimization process continues with the narrowed search-space (Fig. 1).

C. F_1-score statistical evaluation

Every solution of selected features is evaluated with the built classifier of the CART decision tree on the training set and evaluated on the validation set with F_1-score. To prevent the data leakage, the data is split into the training and the validation set. Hence, classification instances used to evaluate the solutions are never seen during the selection and classifier training. F_1-score is a classification measure that consolidates both statistical precision and recall measures. It is calculated as

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}, \tag{4}$$

where $precision = TP/(TP+FP)$ and $recall = TP/(TP+FN)$. Here, TP stands for true positives, FP for false positives and FN for false negatives.

IV. Experiments and Results

The purpose of experimental work was to determine whether the proposed method is beneficial for a practical problem. Control parameters during the experiments were set as designated in Tab. II. Genotype cuts were done at $M = [512, 256, 126, 64]$, total $GEN = 960$ generations; $W = 50$ best performing solutions vote for the removal of 10% of features ($R = 10$ %); the arrhythmia detection dataset was split into $k = 5$ folds to perform cross-validation. To prevent data leakage, each fold was further split into training and validation sets. $N = 10$ consecutive optimization runs were executed, and the results of those runs were averaged. The fitness function for evaluating the solution was $f = 1 - F_1$, so the minimization of f was pursued. All classification tasks were implemented using the CART decision tree. Table III shows the experimental results, with the accuracy and F_1-score classification metric on every fold, for three different methods: (1) CART without FS ("CART"), (2) CART with wrapper-based EvoPreprocess jDE-FS ("EvoFS") and (3) CART with dynamic genotype reduction jDE-FS ("DynFS").

TABLE II: Control parameters. Source: Authors.

Parameter	**Symbol**	**Value**
Genotype changing margins	M	512, 256, 126, 64
Total no. of generations	GEN	960
No. of best performing solutions	W	50
Removal rate	R	10 %
Initial crossover rate	CR	0.8
Initial scaling factor	F	1.0
No. of classification folds	k	5
Training-validation split		50%-50%

TABLE III: Classification results. Highest values are in bold. Source: Authors.

	CART		**EvoFS**		**DynFS**	
Fold	**Acc**	$\boldsymbol{F_1}$	**Acc**	$\boldsymbol{F_1}$	**Acc**	$\boldsymbol{F_1}$
1	96.70	66.67	**98.90**	**90.91**	**98.90**	**90.91**
2	**95.60**	50.00	**95.60**	**60.00**	**95.60**	50.00
3	94.44	44.44	**96.67**	**66.67**	95.56	50.00
4	**97.78**	**80.00**	93.33	57.14	95.56	60.00
5	93.33	50.00	**95.56**	33.33	**95.56**	**60.00**
Mean	95.57	58.22	96.01	61.61	**96.23**	**62.18**
Std. dev.	0.016	0.132	0.018	0.185	**0.013**	**0.150**
Avg. rank	2.2	2.2	**1.4**	1.8	**1.4**	**1.6**

As the results show, the DynFS achieved the best results in three out of five folds accuracy-wise and in two out of five folds F_1-score-wise. Overall, it also achieved the best means in accuracy and F_1-score, with the smallest standard deviation (which means that the results are more robust). Also, the average ranks are the best for both accuracy and F_1-score.

A more in-depth look into the results shows the following phenomenon: in all fivefold, the EvoFS consistently chooses a single feature – the *J vector angle* as most important, while *existence of the diphasic derivation of T wave in channel DI* was the next most frequently chosen feature. On the other hand, the DynFS extended the consistent choices of features from the previously mentioned two, with *R wave average width in V4 channel* and *S' wave in DII channel*.

V. Conclusion

The paper presents the dynamic genotype reduction mechanism for the wrapper-based feature selection optimization problem. The results show that the periodic reductions of the genotype result in superiority, compared to similar optimization techniques, without the genotype reductions implemented. The idea of shrinking the search space seems as very promising to be exploited during the optimization process, to skip the least beneficial search space parts.

Although there are many practical questions with the proposed approach, which should be addressed in the future – namely, the reduction strategy, the reduction frequency (changing margins) and reduction capacities (how much of the search space should be removed from the optimization process), the DynFS might become a serious candidate for preprocessing large-scale datasets.

Acknowledgment

This research was funded by the Slovenian Research Agency (research cores funding Nos. P2-0057 and P5-0027).

References

[1] S. Karakatič, "EvoPreprocess–Data preprocessing framework with nature-inspired optimization algorithms," *Mathematics*, vol. 8, no. 6, p. 900, 2020.

[2] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," *Journal of Open Source Software*, vol. 3, no. 23, p. 613, 2018.

[3] D. Fister, I. Fister, T. Jagrič, I. Fister, and J. Brest, "A novel self-adaptive differential evolution for feature selection using threshold mechanism," in *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018*. IEEE, 2019, pp. 17–24.

[4] A. Wang, N. An, G. Chen, L. Li, and G. Alterovitz, "Accelerating wrapper-based feature selection with K-nearest-neighbor," *Knowledge-Based Systems*, vol. 83, pp. 81–91, 2015.

[5] J. Leng, C. Valli, and L. Armstrong, "A Wrapper-Based Feature Selection for Analysis of Large Data Sets," *Proceedings of 2010 3rd International Conference on Computer and Electrical Engineerings (ICCEE 2010)*, no. 2010, pp. 167–170, 2010.

[6] R. Wald, T. M. Khoshgoftaar, and A. Napolitano, "Stability of filter- and wrapper-based feature subset selection," in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*. IEEE, 2013, pp. 374–380.

[7] R. Bellman, "Curse of dimensionality," *Adaptive control processes: a guided tour. Princeton, NJ*, vol. 3, p. 2, 1961.

[8] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[9] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, dec 2006.

[10] L. Brezočnik, I. Fister, and V. Podgorelec, "Swarm intelligence algorithms for feature selection: A review," *Applied Sciences (Switzerland)*, vol. 8, no. 9, p. 1521, 2018.

[11] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A Survey on Evolutionary Computation Approaches to Feature Selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[12] Y. Xue, B. Xue, and M. Zl, "Self-Adaptive particle swarm optimization for large-scale feature selection in classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 5, pp. 1–27, 2019.