Differential Evolution and Perceptron Decision Trees for Fault Detection in Power Transformers

Freitas, A. R. R. and Guimarães, F. G.

Abstract Classifying data is a key process for extracting relevant information out of a database. A relevant classification problem is classifying the condition of a transformer based on its chromatography data. It is a useful problem formulation as its solution makes it possible to repair the transformer with less expenditure given that a correct classification of the equipment status is available. In this paper, we propose a Differential Evolution algorithm that evolves Perceptron Decision Trees to classify transformers from their chromatography data. Our approach shows that it is possible to evolve classifiers to identify failure in power transformers with results comparable to the ones available in the literature.

1 Introduction

Classification problems can aid the process of decision making in many real world applications. Among those is the extraction of relevant characteristics about clients or preventing failure in equipments.

As an example of a classification problem, the composition of gases present in a transformer can indicate its condition of operation (Section 2). By obtaining the classification rules we can then infer information on classes from the data related to the problem.

Thus, we show an approach to generate classification rules with Differential Evolution (DE) algorithms. DE is a metaheuristic that iteratively searches high quality

Freitas, A. R. R.
Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG, 31270-901 Brazil, e-mail: alandefreitas@gmail.com

Guimarães, F. G.
Departamento de Engenharia Elétrica - Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG, 31270-901 Brazil, e-mail: fredericoguimaraes@ufmg.br

solutions (Section 3). Each solution used by our DE algorithm is a Perceptron Decision Tree (PDT), which are decision trees that consider all the attributes of the data in each of its nodes (Section 4).

By describing the methodology of this work, this paper introduces the following contributions:

- An algorithm based on Differential Evolution for evolving Perceptron Decision Trees (Section 5)
- An approach for representation and manipulation of PDT in the context of DE solutions (Section 5.2-5.3)
- A strategy for replacing solutions (Section 5.4), in which the worst classifiers give place to new ones
- A method for controlling the legitimacy of the evaluation (Section 5.5) that makes evaluation more legitimate when it seems to be necessary

The results of our final algorithm for different databases are compared (Section 6). Those results are also compared to works with similar databases (Section 7). We conclude that even though a more extensive study is needed, the proposed algorithm is efficient for solving the problem with chromatography database (Section 8).

2 Dissolved Gas Analysis in Power Transformers

Dissolved Gas Analysis (DGA) is a usual method for detecting faults in power transformers as it allows diagnosis without de-energizing the transformer [18].

For the experiments of this work we have used three databases of chromatography related to power transformers. In order to find potential failure in transformers, we can analyze the gases produced in the equipment oil as it is exposed to heat [13]. The problem consists in diagnosing the fault from the data describing the gases generated under that condition.

Thus, the problem of analyzing dissolved gases consists in finding rules that can identify failures in the transformers. The possible classifications of a transformer are (i) normal, (ii) electrical failure, and (iii) thermal failure.

3 Differential Evolution

Differential Evolution [12, 17] is an algorithm developed to improve the quality of a solution over many iterations. A population of candidate solutions is kept and new solutions are generated and tested from a simple combination of preexistent solutions. The best solutions are then kept. DE is an algorithm employed for real valued functions but it does not use the gradient of the function to be optimized. Thus, the problem being solved does not have to be differentiable. Many books have been published approaching theoretical aspects of DE, see for instance [8, 10].

Through the basic iterative process of DE as the one described below, it is expected that a satisfactory solution will be found:

- Generate many solutions $\mathbf{x}$ with random positions in the search space.
- Initialize the crossover probability CR and differential weight F
- Until a halting criterion is not met, the following steps are repeated:
  - For each solution $\mathbf{x}$:
    - Choose other three solutions $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ from the population that are different from $\mathbf{x}$.
    - Choose a random index R between 1 and the population size.
    - In order to generate a new solution $\mathbf{y}$, for each position i of the solution:
      - Generate a real valued number r between 0 and 1 with uniform distribution
      - If $r < CR$, $\mathbf{y}_i = \mathbf{a}_i + F(\mathbf{b}_i - \mathbf{c}_i)$, else $\mathbf{y}_i = \mathbf{x}_i$
    - If $\mathbf{y}$ is better than $\mathbf{x}$, $\mathbf{x}$ is replaced by $\mathbf{y}$
- Return the best solution found

The values F, CR and the population size must be chosen by the person using the algorithm. Given a suitable formulation of a problem of data classification, DE can then evolve classification rules for the test database.

4 Perceptron Decision Trees

A common approach for classifiers is to employ binary classification trees. In those trees, the condition of one of the attributes is tested accordingly to a threshold. The difference between the attribute value and the threshold defines the next tree node to be considered and the process occurs again. This process repeats until a leaf node is reached. In those nodes, there is information on the class that should correctly classify the data.

In this work, we propose a different approach for defining each tree node. With the equation $\mathbf{wx} + \theta = z$, where $\mathbf{w}$ as weight scalars, $\mathbf{x}$ are the data attributes and θ is a constant, we can linearly divide the search space into areas where $z > 0$ and areas where $z < 0$. With this approach, we define a PDT.

Thus, we can divide the search space considering all data attributes at each step, differently from the conventional binary tree, in which the decision is made based on one attribute only. In case we want to consider only the attribute i with the PDT, the vector $\mathbf{w}$ must be constituted of zeros, apart from the position $\mathbf{w}_i$, where its value will be 1.

Many authors have employed the similar concepts of aggregating many linear classifiers or perceptrons in a tree under different names [3, 4, 5, 6], including PDT [2].

5 Methodology

In this section we describe how DE was used for evolving PDT. The problem presents many possible parameters and a brief study of those is presented.

5.1 Data

The database employed in this work consists in chromatography data of transformers. The data has 5 attributes, representing the quantity of each of the following gases: (i) Hydrogen, (ii) Methane, (iii) Acetylene, (iv) Ethane, and (v) Ethylene. Besides the value of each of those attributes, there is also the classification of the transformer into 3 possible classes: Normal (Class 1), Electrical Failure (Class 2), and Thermal Failure (Class 3).

Three different unbalanced databases were used for the tests. A brief description of the databases is shown on Table 1. We can notice that most samples belong to class 1 and that is mostly due to the unbalancing of the database 3.

Table 1 Databases

Database	Number of samples	Class 1	Class 2	Class 3
DB 1	52	30,77%	42,31%	26,92%
DB 2	232	39,22%	26,29%	34,48%
DB 3	224	81,70%	5,80%	12,50%
Total	508	57,09%	18,90%	24,02%

The information was split into two sets. The first set, with 70% of the samples, contains the training set for the generation of the classifier while the second dataset, with 30% of the samples, contains the validation data, for testing the generalization capacity of the classifier.

5.2 Representation of a Classifier

As seen in Section 4, each PDT contains many linear classifiers, being each of those defined by a vector of weights $\mathbf{w}$ of size n and a constant θ, being n the number of attributes of the classification problem. The number of classifiers in a PDT is $2^{(h-1)}-1$, being h a variable defined as the depth of the PDT.

Besides the classifiers, each PDT contains $2^{(h-1)}$ leaf nodes, which contain a possible classification for a sample. Those nodes are defined in the discrete space.

Thus, a complete PDT can be defined by the matrices of size $n \times 2^{h-1} - 2$ for the weights, $1 \times 2^{h-1} - 1$ for the constants and $1 \times 2^{h-1}$ for the leaf nodes. This is the size of each solution employed by the DE.

Each of the $2^{h-1} - 1$ columns of the classifier matrices defines a classifier. After using the linear classifier at position i of n, the classifier at position $2i$ is used if the output of the classifier is < 0 or the classifier $2i + 1$ is used otherwise.

As the total error will be the standard of comparison between the algorithms, the objective function value of each of the solutions is defined as the number of misclassifications, including false positives and false negatives. This objective function, naturally, must be minimized by the DE.

Three initial values for h were tested. While the individual size is $O(2^{h-1} - 1)$, the evaluation cost is still $O(h)$ because only one possible path is searched through the PDT. The values h of 3, 5 and 7 were tested for evolving PDT. The PDT had representation size 3, 15 and 63, respectively. Where $h = 3$, 333 generations were executed, with $h = 5$, 200 generations, and with $h = 7$, 142 generations. That was meant to keep at least the evaluation cost similar throughout the generations. Ten executions using all the databases with $h = 5$ achieved an initial average training error of 17.65%, with $h = 3$ had 18.57% and with $h = 7$ had 20.01%.

5.3 Operators

Simple operators, as shown in Section 3 were employed for the generation of new solutions. Values of F and CR are defined as random values between 0.4 and 1 and between 0.9 and 1, respectively [17]. Those values are altered at each iteration of a generation.

As for the leaf nodes, a discrete approach must be used for the crossover of solutions [15]. The approach used is inspired in the idea that DE uses a vector of differences to alter the solution, however now this vector represents swap movements between two possible positions.

In doing so, the movement described as $\mathbf{b}_i - \mathbf{c}_i$ is only performed if $\mathbf{a}_i = \mathbf{b}_i$. If $\mathbf{a}_i = \mathbf{c}_i$, the movement $\mathbf{c}_i - \mathbf{b}_i$ is applied. In a third possible case, where $\mathbf{a}_i \neq \mathbf{b}_i$ and $\mathbf{a}_i \neq \mathbf{c}_i$, no movement is performed.

In this case, the value F is used to define if the operation should happen. The operation occurs if a randomly generated number is less than F.

5.4 Replacement of Individuals

As the replacement of individuals is made one by one, it may happen that some solutions are stagnant in bad points of the search space. In order to avoid this problem, a new individual survival operator was developed to keep always new candidates in the population.

At each generation, new random individuals are generated to replace the $x\%$ worst individuals. Moreover, the individuals with an error rate greater than $1/n_{classes}$ are also replaced.

Tests with 3 levels for x were performed. When the value of x was 0%, the response in 200 generations had an average error of 19.01%, when $x = 10\%$ the final classifier had 16.34% and when $x = 20\%$ the answer had error 18.11%. The individual replacement value was then adjusted to $x = 10\%$.

5.5 Legitimacy

In the initial generations, where all the individuals are still very random, not many comparisons are needed to perceive that some are better than the others. In most cases, with the employment of classifiers in less than 10 samples it is possible to define clearly how some PDT are better than the others. The same does not apply after many generations, when most solutions classify the samples with a low error rate.

Having this in mind, and the high cost of evaluating solutions, which may take as much as 10 seconds per generation, we propose an approach to reduce the time spent to evaluate solutions in the first generations.

The key idea is that initial solutions do not need to be evaluated with the same legitimacy as the solutions in the last generations, where a more refined analysis is necessary to distinguish good solutions. The value l defines the legitimacy utilized in the evaluation of individuals. At a generation with legitimacy l, only l samples are tested in the evaluation of each solution.

The initial value of this parameter was defined as twice as the number of attributes of each samples. After each generation, the value l is updated accordingly to the equation $l = \lceil \min(N, l * \alpha) \rceil$ where N is the number of available samples for test and α is the rate of increase of the legitimacy for each generation.

The parameter α was defined as $1 + \delta/\sigma$, where δ is the speed of legimitization, defined as 10^{-2}, and σ is the standard deviation of the objective function value of the solutions. Thus, when the solutions are still very diverse, the parameter α is smaller.

Some individuals can occasionally have an objective function value smaller than it is due in cases where the legitimization is low. For this reason, whenever the objective function value is calculated, the legitimacy value used to obtain the objective function value is also stored. Thus, if the individual is not replaced by a new one after t generations and the legitimacy value has already increased, the individual is reevaluated with the new legitimacy value l. The value of t was defined as 10.

In Figure 1, we have the relation between execution time and those legitimacy values. In this plot, the dashed line represents the legitimacy l divided by N and the continuous line represents the best objective function value found so far in a given generation. The time spent by each generation is directly linked to the legiti-

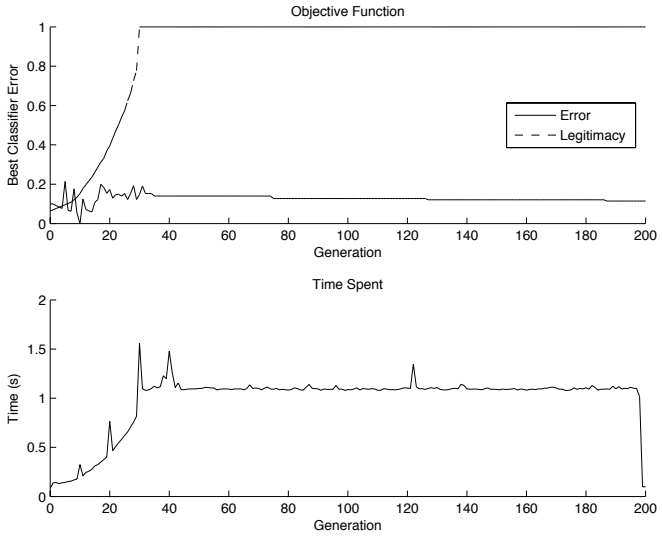

Fig. 1 Relation between evaluation legitimacy, objective function and time

macy of that generation. We can perceive the direct relation between the evaluation legitimacy and the time spent per generation by comparing the graphs.

The best objective function value known also varies much before the legitimacy value reaches its maximum value. That happens because some solutions can obtain good objective function values which are not so appropriate and this is corrected only after t generations. After full legitimacy has been reached, there is no other referential to evaluate the individuals because all the database is already being used. Therefore the best objective function value will only decrease after this.

6 Results

In computational tests, the average time for executing a DE with 200 generations for a joint database with all the samples was 17 minutes. Have the number of factor in mind, we realized 3 experiments in each of the 3 databases. That was defined as experiment 1, where the number of generations is 200.

In Table 2 we have the results of the tests with 200 generations. In the Table, we have the average time spent, the average and minimum error obtained for the training set, and the average and minimum error obtained for the validation set.

The algorithm is using the replacement operator and it does not present final convergence after 200 iterations. For this reason, in order to have a reference in relation to the capacities of the algorithm in a real situation of search for a good

Table 2 Results of tests with 200 generations

Database	Time (s)	Training Error	Minimum Error	Validation Error	Minimum Error
DB1	59	1,85%	0%	20,83%	6,25%
DB2	224	26,13%	25,31%	38,57%	34,29%
DB3	218	11,46%	8,92%	17,91%	14,93%

classifier, we performed a second experiment. In this new test, we performed only one execution of the algorithm for each database with 2000 generations. Thus, we have an idea of the capabilities of the algorithm with more generations. This test was defined as experiment 2, where the number of generations is 2000.

Table 3 shows the results of the tests. The columns represent the time spent, error in the training set and error in the validation set.

Table 3 Results of the test with 2000 generations

Database	Time (s)	Training Error	Validation Error
DB1	648	0%	6,25%
DB2	2720	17,28%	22,86%
DB3	2471	10,19%	20,90%

7 Discussion

Some works have been done to diagnose failure in transformers including Artificial Neural Networks [20], Neural Networks with expert systems [19], decision trees [7] and genetic algorithms with niches [14].

In other databases for the same problem, Pereira and Vasconcelos [14] obtain a right classification rate of 91%, comparable to many cases of this algorithm. In a profound analysis of the problem, he defines many important criteria for classification. In his work, an approach based in niches leads to different solutions that value correct classification in each of the classes.

A deeper study with benchmark databases is needed to have more general conclusions on the behavior of the proposed algorithm. Although other works have used other databases, if we assume similar complexity of the data to be classified, the results obtained in this work are satisfactory.

Castanheira [7] presents also good results for the problem with the use of neural networks and decision trees, having the last ones obtained better results. The modeling of the problem through PDT have the capacity to classify data in many

situations and it has clear advantages in relation to classification power when compared to simple decision trees. The properties of the PDT with DE still need to be deeply studied such that better results can be obtained.

Despite the classification power of PDT, the computational time spent with the operators in the case of a large tree grows exponentially and therefore it must be carefully adjusted.

8 Conclusion and Future Work

The halting criterion of 200 generations was decided in regard to the computational time used for each experiment, with the intention that many tests should be possible. However, it is likely that a greater number of generations is more suitable for the algorithm to have time to converge almost completely. When the tests were performed with 2000 generations, the validation error of the algorithm kept falling for the databases, indicating that the algorithm was not yet presenting overfitting problems. Recognizing the number of generations needed for the validation error to begin to rise is important because from this point on, new classifiers that model only noise of the training set are being generated.

The algorithm certainly is suitable for solving the problem. Nevertheless, for a comparison with the algorithms in the literature, it would be important to have an analysis with other benchmark databases.

All the parameters of the algorithm were adjusted separately, considering that there are no interaction between the factors. Of course, this adjustment has led to a condition of execution of the algorithm which is better than the initial one. However, before implementing new features in the algorithm, it would be important to adjust the parameters with factorial experiments where it is possible to better understand the interaction between the parameters. As each iteration takes much time, this experiment would have to be very well planned.

In the evolution of the classifiers, only the training error was used. In spite of the fact that we can not use the validation set throughout the evolution, a strategy of objective function that maximizes the separation margin of the data can be used to broaden the capacity of generalization of the classifiers, alike the way Support Vector Machines work [1, 9]. A boosting strategy can be applied to increase the margin by including a rise in the probability of classifying the most difficult samples [16, 11].

Another important issue in the definition of the objective function is to define the costs involved in the process. A low error rate may not simply represent good solutions for practical applications because they do not represent the specific errors of each class and the most important: the cost involved in each sort of error.

In any of the problems mentions here, however, the approach based on DE and PDT has been shown to be useful for the solution of the problem.

Acknowledgements

Acknowledgements This work has been supported by the Brazilian agencies CAPES, CNPq, and FAPEMIG; and the Marie Curie International Research Staff Exchange Scheme Fellowship within the 7th European Community Framework Programme.

References

1. Angulo, C., Gonzalez-Abril, L.: Support vector machines. Pattern Recognition p. 1pp (2012)
2. Bennett, K., Cristianini, N., Shawe-Taylor, J., Wu, D.: Enlarging the margins in perceptron decision trees. Machine Learning **41**(3), 295–313 (2000)
3. Bennett, K., Mangasarian, O.: Robust linear programming discrimination of two linearly inseparable sets. Optimization methods and software **1**(1), 23–34 (1992)
4. Bennett, K., Mangasarian, O.: Multicategory discrimination via linear programming. Optimization Methods and Software **3**(1-3), 27–39 (1994)
5. Breiman, L.: Classification and regression trees. Chapman & Hall/CRC (1984)
6. Brodley, C., Utgoff, P.: Multivariate decision trees. Machine Learning **19**(1), 45–77 (1995)
7. Castanheira, L.: Aplicação de mineração de dados no auxílio à tomada de decisão em engenharia. Master's thesis, Universidade Federal de Minas Gerais (2008)
8. Chakraborty, U.: Advances in differential evolution, vol. 143. Springer Verlag (2008)
9. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
10. Feoktistov, V.: Differential evolution: in search of solutions, vol. 5. Springer-Verlag New York Inc (2006)
11. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence **14**(771-780), 1612 (1999)
12. Mallipeddi, R., Suganthan, P., Pan, Q., Tasgetiren, M.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing **11**(2), 1679–1696 (2011)
13. Morais, D., Rolim, J.: A hybrid tool for detection of incipient faults in transformers based on the dissolved gas analysis of insulating oil. Power Delivery, IEEE Transactions on **21**(2), 673–680 (2006)
14. Pereira M. A.; Vasconcelos, J.A.: A niched genetic algorithm for classification rules discovery in real databases. In: Anais do XVIII Congresso Brasileiro de Automática (2010)
15. Prado, R.S., Silva, R.C.P., Guimarães, F.G., Neto, O.M.: Using differential evolution for combinatorial optimization: A general approach. In: SMC, pp. 11–18 (2010)
16. Schapire, R., Freund, Y.: Boosting: Foundations and algorithms (2012)
17. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4), 341–359 (1997)
18. Sun, H., Huang, Y., Huang, C.: A review of dissolved gas analysis in power transformers. Energy Procedia **14**, 1220–1225 (2012)
19. Wang, Z., Liu, Y., Griffin, P.: A combined ann and expert system tool for transformer fault diagnosis. Power Delivery, IEEE Transactions on **13**(4), 1224–1229 (1998)
20. Zhang, Y., Ding, X., Liu, Y., Griffin, P.: An artificial neural network approach to transformer fault diagnosis. Power Delivery, IEEE Transactions on **11**(4), 1836–1841 (1996)