Accelerated Evolutionary Induction of Heterogeneous Decision Trees for Gene Expression-Based Classification

Marcin Czajkowski
Bialystok University of Technology
Poland
m.czajkowski@pb.edu.pl

Krzysztof Jurczuk
Bialystok University of Technology,
Poland
k.jurczuk@pb.edu.pl

Marek Kretowski
Bialystok University of Technology,
Poland
m.kretowski@pb.edu.pl

ABSTRACT

Decision trees (DTs) are popular techniques in the field of eXplainable Artificial Intelligence. Despite their effectiveness in solving various classification problems, they are not compatible with modern biological data generated with high-throughput technologies. This work aims to combine evolutionary induced DT with a recently developed concept designed directly for gene expression data, called Relative eXpression Analysis (RXA). We propose a new solution, termed Evolutionary Heterogeneous Decision Tree (EvoHDTree), which uses both classical univariate and bivariate tests that focus on the relative ordering and weight relationships between the genes in the splitting nodes. The search for the decision tree structure, node representation, and splits is performed globally by the evolutionary algorithm.

To meet the huge computational demands, we enriched our solution with more than a dozen specialized variants of recombination operators, GPU-computed local search components, OpenMP parallelization, and built-in gene ranking to improve evolutionary convergence. Experiments performed on cancer-related gene expression-based data show that the proposed solution finds accurate and much simpler interactions between genes. Importantly, the patterns discovered by EvoHDTree are easy to understand and to some extent supported by biological evidence in the literature.

CCS CONCEPTS

• **Computing methodologies** → *Classification and regression trees*; Supervised learning by classification; • **Applied computing** → *Bioinformatics.*

KEYWORDS

evolutionary data mining, decision trees, relative expression analysis, gene expression data

ACM Reference Format:
Marcin Czajkowski, Krzysztof Jurczuk, and Marek Kretowski. 2021. Accelerated Evolutionary Induction of Heterogeneous Decision Trees for Gene Expression-Based Classification. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459376

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459376

1 INTRODUCTION

The rapid growth and popularity of high-throughput technology [22] have led to increasing demand for new solutions in molecular biology analysis. Most of them fall under the discipline called data mining, an umbrella term that encompasses a wide range of tools and techniques for extracting hidden knowledge from large quantities of data. Working with biomedical data can be challenging due to its enormous dimensionality, natural diversity, experimental noise, and other perturbations. Moreover, the large size of the data as well as the computational complexity of the algorithms is often a bottleneck in processing and analyzing real genomic data. Finally, currently developed machine learning algorithms for biomedical data tend to focus almost exclusively on prediction accuracy and propose complex predictive models. Such an approach hinders the process of uncovering new biological understanding and is often an obstacle for mature applications [2].

In this research, we focus on the more underappreciated but much-needed computational methods for biomedical analysis in the field of eXplainable Artificial Intelligence (XAI) [26]. These solutions can perform predictions efficiently, but more importantly, they provide insight that may help identify relationships between specific features and improve biomarker discovery. After all, this is the ultimate goal of data-driven biology. To address this problem, we extended the Decision Tree (DT), which is a highly popular white-box approach [16], to unlock its potential in modern biological data analysis. Therefore, we designed an evolutionary algorithm that combines DTs with a powerful collection of new computational methods with easily interpretable models for classifying gene expression data, called Relative eXpression Analysis (RXA) [10], to take the best of both worlds. Although we consider only genomic data, we believe that our approach can also be applied to other types of highly multidimensional biological data, including metabolomics and proteomics.

DTs are known for their ease of use, speed of classification, and efficiency [16]. However, currently traditional single-tree approaches are not really applicable for genomic data classification. Existing research has shown [17] that DT algorithms often induce classifiers with weaker prediction so there is more interest in trees as sub-learners of an ensemble learning such as Random Forests. These solutions mitigate the problem of low accuracy by averaging or adaptively combining multiple trees. However, when modeling to understanding underlying processes, such methods are not as useful because they generate more complex and less understandable models.

In contrast, RXA represents a family of new methods designed directly for gene expression data [10]. In general, they focus on

finding interactions among a small collection of genes by studying e.g. the relative ordering of their expressions rather than their raw values. Classification algorithms based on this methodology are known to be accurate, robust to methodological and technical factors, study-specific biases as well as normalization and standardization procedures. However, one of the most important limitations of the RXA concept is enormous computational complexity, which forces the use of heavy feature selection and other simplifications or restrictions to the algorithm implementation.

Our paper makes several important contributions to the literature. First, we propose a new hybrid method called Evolutionary Heterogeneous Decision Tree (EvoHDTree), which combines the power of DT and unified variants of the RXA-family algorithms based on ordering and weight comparisons. Its novelty lies in its flexible tree node representation, which involves both classical univariate and bivariate tests inspired by the RXA concept. Such a self-adaptive EvoHDTree can not only explore a much larger solution space but also may reduce over- and under-fitting that is common for high-dimensional biological data [7]. Second, by incorporating our knowledge of decision tree induction and RXA methodology and designing more than a dozen specialized variants of recombination operators, we improved evolutionary exploration and exploitation.

The search for the overall decision tree structure, node representations, and splits is performed globally by the evolutionary algorithm. To overcome the enormous computational complexity resulting from the use of RXA, we implemented OpenMP and GPU-based parallelization. Furthermore, more discriminative genes based on embedded ranking are preferred. Finally, experiments performed on cancer-related gene expression-based data indicate that the proposed heterogeneous tree representation finds accurate and more precise,gene-gene interactions that are not limited by the representation. We believe that the combination of these two XAI approaches has the potential to provide far-reaching benefits in the field of computational molecular biology.

This paper is organized as follows. Section 2 presents related work along with RXA, DTs, and topic-related parallelization. Section 3 describes in details our hybrid EvoHDTree solution and Section 4 possible acceleration. This is followed by experimental validation on real-life datasets, and in the last section, we summarize the work and present possible future work.

2 RELATED WORK

A brief taxonomy of the family of RXA algorithms and DTs inducers are shown in Figure 1. The proposed EvoHDTree system applies to all the boxes highlighted in gray.

2.1 Decision trees

DTs have a knowledge representation structure consisting of nodes and branches, where: each internal node is associated with a test on one or more attributes; each branch represents a test result; and each leaf (terminal node) is designed by a class label. Typically, a tree induction algorithm partitions the feature space using axis-parallel hyperplanes according to the given goodness of split. We call such trees *univariate* because the tests in the internal nodes consist of a single feature. Trees with multivariate tests (usually called *oblique*) are based mainly on linear combinations of multiple dependent attributes and divide the feature space by a non-orthogonal hyperplane.

Since it is known that the induction of an optimal DT is an NP-complete problem, [14], practical learning algorithms must be heuristically enhanced. There are two main types of tree induction: greedy search [23] and evolutionary approach [1]. In order to mitigate some of its negative effects of locally optimal solutions generated by the greedy top-down inducers, a wide range of metaheuristics have been proposed [1]. An evolutionary induced DT for gene expression data is presented in [2] where the authors propose HEAD-DT inducer. This solution was later contrasted with an Evolutionary Multi-Test Tree [7], which globally searched for the sets of similar univariate tests within each node of the tree.

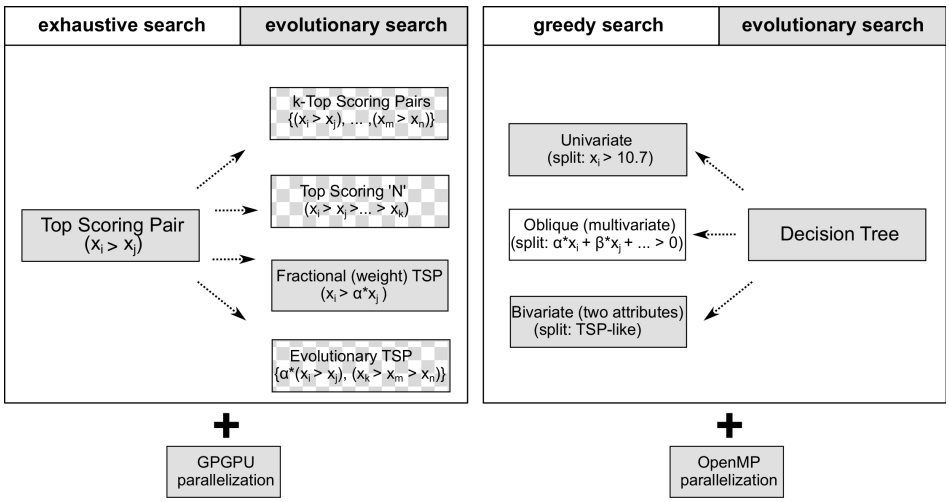

Figure 1: The brief taxonomy of the RXA and DT family of algorithms. Gray boxes refer to the proposed EvoHDTree system.

2.2 RXA-based classifiers

The basic concept of Relative Expression Analysis (RXA) focuses on studying the relative ordering of expression among a small number of transcripts. A pioneering study in 2004 proposed the Top Scoring Pair (TSP) method, [11], which is a straightforward prediction rule. It's based on the RXA concept that uses building blocks of rank-altered gene pairs in case and control comparison. Such gene pairs can be viewed as *biological switches* that can be directly related to regulatory $motifs$ or other properties of transcriptional networks. The discriminating power of each pair of genes i, j was measured by the absolute difference between the probabilities P_{ij} of the event $(x_i > x_j)$ in two classes. One of the first extensions of the TSP solution is the k-TSP algorithm [25]. By using no more than k top scoring disjoint gene pairs (k determined by internal cross-validation) and simple majority voting, it was able to significantly improve classification accuracy. An alternative Top Scoring N (TSN) [19] approach focuses on exploring relationships between more than two genes. A more general form of gene-gene interaction called $weighting\ relationship$ has been proposed in [6]. The authors introduced an additional component w, which is the ratio of the genes relationships in a pair: $(x_i > w \times x_j)$ among instances.

To the best of our knowledge, the only research combining raw data values with RXA was proposed in [13]. The authors proposed an ensemble greedy classifier called VH-k-TSP to evaluate feature pairs based on vertical and horizontal relationships. Experimental evaluation was performed on gene expression and metabolomic data. The application of EA to RXA was initially proposed in the EvoTSP solution [5]. Recently, it has been extended to use DT with node splits based on a single Weight TSP pair in the REDT [6] algorithm. The results indicate that evolutionary search is a good alternative to traditional RXA algorithms. Thus, our motivation is to combine all these three concepts: single-attribute tests from VH-k-TSP with ordering and weight gene-pair relationships to see if such self-adapting heterogeneous tree can improve the discovered patterns and relations within the data.

2.3 Parallelization

Fortunately, EAs are naturally amenable to parallelism, and the artificial evolution can be implemented in a variety of ways [12]. Parallelization can also be performed by different paradigms: OpenMP, which uses a shared address space and works well on multicore chips; Message Passing Interface (MPI) for computer clusters; general -purpose computation on graphic processing units (GPGPU). In the context of parallel DT induction, most of the research focuses on top-down induced trees, such as CUDT [18], to find the best locally optimal tests through parallel attribute processing. The second type of DT systems involve parallelization of ensembles of trees, such as random forests where a single CUDA thread builds one tree in a forest.

As for evolutionary induced DTs, we found only a few works that deal with parallel extensions of the Global Decision Tree (GDT) data mining system [17]. The best results were obtained for GPU-accelerated solutions, which were able to induce trees two orders of magnitude faster compared to the original CPU-based version. As for RXA, GPU-accelerated parallelization appears to be highly effective [19].

3 EVOHDTREE

This paper presents an accelerated Evolutionary Heterogeneous Decision Tree (EvoHDTree) along with a new, richer language for representing decision trees. The proposed solution extends the Global Decision Tree (GDT) system [17] which can be seen as an underlying framework for inducing different types of trees.

3.1 Representation

Since the number of EvoHDTree nodes and their representation is not known in advance, we use a tree-encoding schema where the individuals are in their actual form as potential tree solutions.

The overall structure of EvoHDTree is not much different from a standard binary classification tree, e.g., C4.5 [21] as it is illustrated in Figure 2. The main change is that each internal node of the tree can contain a different split representation composed of:

- univariate test: a typical inequality test as in C4.5 [21] consisting of one continuous attribute and a threshold value (e.g. $x_i > 10$);
- TSP-like test: an adaptation of the TSP algorithm [5] that uses a pair of genes and their ordering relation to split instances (e.g. $x_i > x_j$);
- Weight TSP-like test (see Algorithm 1): an adaptation of the relative weight comparison algorithm [6] that uses a pair of TSPs and a weight w (e.g. $x_i > w \times x_j$).

To more effectively perform local modifications of the tree structure and tests when applying genetic operators, additional information about training instances and related statistics are stored. Additionally, EvoHDTree keeps an embedded ranking of genes passed as an input of the algorithm. It contains the gene name and its cost C, which corresponds to the discrimination power ranking generated manually or by some feature ranking algorithm. This cost is later used in the fitness function and in the mutation operator so the top-genes from the ranking are more likely to be considered. The gene rank (cost) ranges from 0.5 to 1, while 0.5 corresponds to the highest-ranked gene and 1 to the worst-ranked gene.

3.2 Initialization

Initial individuals are created by using a top-down algorithm with random subsamples of the original training data. This helps to maintain a balance between exploration and exploitation. At each EvoHDTree internal node, the algorithm divides the training instances that reach this particular point by one of the three tests (equal probability of selecting univariate, TSP-like or Weight TSP-like test) built on a random attribute (or attributes). The reason why we add the suffix *like* to TSP and Weight TSP is that the original ranking that is based on classification error can only be used in a

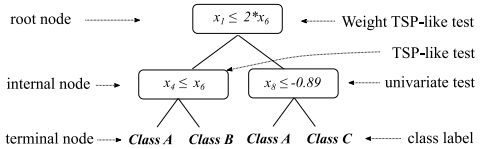

Figure 2: Example representation of EvoHDTree.

input : instances that reach the node (matrix $N \times M$)
input : r smoothness factor of the weight (w) relation
output : Top weight TSP-like test based on Gini impurity

```
for i ← 1 to M do                    // i,j are the attributes
    for j ← 1 to M; i! = j do
        for u ← 1 to N do
            w = round(x_iu/x_ju, r)          // where x_ju!=0
            B_L ⟵ ∅;                          // left branch
            B_R ⟵ ∅;                         // right branch
            for v ← 1 to N do
                if x_iv < w * x_jv then
                    B_L ⟵ X_v;
                else
                    B_R ⟵ X_v;
                end
            end
            /* R_1 primary & R_2 secondary ranking  */
            R_1 = GiniImpurity(B_L) + GiniImpurity(B_R);
            R_2 = abs(w − x_iu/x_ju);
            /* store i',j',w',R'_1,R'_2 for min(R_1&R_2) */
        end
    end
end
return i', j', w', R'_1, R'_2
```

Algorithm 1: Weight TSP-like test search

binary classification problem. Therefore, all test variants at non-terminal nodes are scored according to the Gini Impurity [6], which is a well-known partitioning criterion for DT.

Algorithm 1 describes how a Weight TSP-like test is generated. Of the three tests, this is the most advanced and computationally intensive test because its overall complexity is $O((M * N)^2)$, where M is the number of attributes and N is the number of instances. The remaining tests are simplified versions of *Algorithm* 1 because, for (i) the TSP-like test, the weight of w always equals to 1, so iteration through u is not required; (ii) the univariate test involves a single attribute, so only two nested loops are required. The secondary ranking R_2 is the tie-breaker and is equal to the absolute difference between the smoothed weight value and the actual value.

3.3 Genetic operators

Using the solution representation as the solution itself often requires the design of specialized genetic operators, which is the case in the proposed EvoHDTree heterogeneous tree-encoding schema. To preserve genetic diversity, two specialized genetic meta-operators corresponding to classical mutation and crossover are proposed. The operators can have a three-level effect on individuals as they can impact the structure of the decision tree, the representation of nodes, and the split test itself.

The mutation operator starts by randomly selecting a node type (non-terminal node or leaf). A ranked list of nodes of the selected type is then created and a mechanism analogous to ranking selection is used to decide which node will be affected by the mutation.

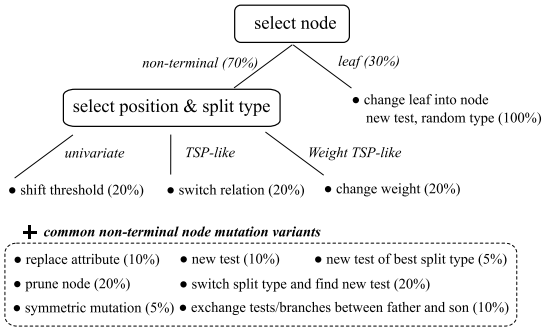

Figure 3: Mutations variants for EvoHDTree along with their relative probabilities of occurrence.

Two aspects are considered to determine which node is more likely to be selected:

- the position (location) of the tree node. It is preferable to modify nodes in the lower parts of the tree more frequently since such a change has only a local impact. It is clear that modifying the root or internal nodes in the upper parts of the tree affects the entire DT;
- tree node quality, since nodes with higher error per instance should be mutated with higher probability.

A list of possible mutation variants is shown in Figure 3. Some of the variants are unique to the split type (representation), e.g., shifting the threshold value of a univariate test; changing the relation of a TSP-like test; or modifying the fractional relationship in a Weight TSP-like test. Others affect the split representation or modify two nodes simultaneously. There are also variants of operators that act as a pruning procedure (internal or straight to the leaf). Finally, there are variants that look for new tests, some even check all three split representations and choose the best one.

The crossover procedure starts with the random selection of two individuals to interact with and the selection of positions (nodes) in both individuals. Three variants of the crossover are used on randomly selected nodes and involve the simple exchange of tests, branches, and subtrees (see Figure 4). The two variants shown in Figure 5 require an additional mechanism to decide which node will be affected - analogous to the mutation variants. The probability of using each variant is equal. The algorithm ranks all tree nodes in both individuals according to their classification accuracy and the probability of selection is proportional to the rank in a linear manner. Nodes, for example, with a low classification error, are more likely to be donors, while weak nodes (with high classification error) are more likely to be replaced by donors (and become a recipient). We also allow recipient nodes to be replaced by a subtree that starts at the donor node of the best individual. Note that only one individual is affected in such recombination.

3.4 Fitness and selection

Decision trees are prone to overfitting, especially when the tree is particularly large [7]. In typical greedy induction, this problem is partially mitigated by the post-pruning procedure. In the case of evolutionary induced DT, this problem can be controlled by a

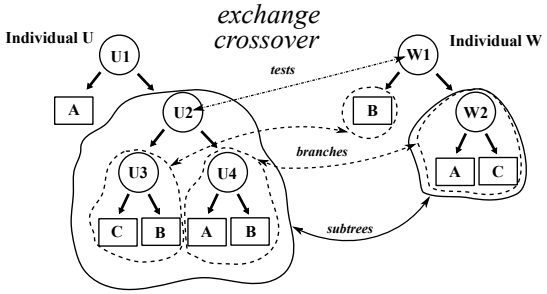

Figure 4: EvoHDTree exchange crossover variants: swap tests, branches and subtrees.

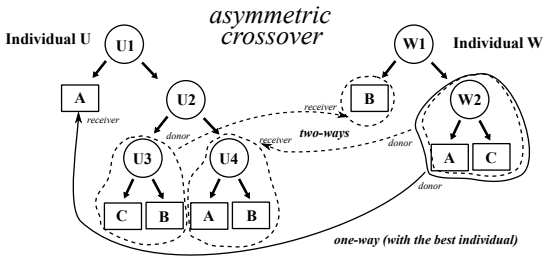

Figure 5: EvoHDTree asymmetric crossover variants: two-way exchange and one-way with the best individual.

multi-objective fitness function to maximize accuracy and minimize the complexity of the output tree. In this paper, a modified weight fitness function proposed by the GDT system is used:

$$Fitness(T) = Accuracy(T) - \alpha * Complexity(T), \quad (1)$$

where $Accuracy(T)$ represents the classification quality of the tree T estimated on the learning set, $Complexity(T)$ which reflects the total cost of attributes that constitute tests at non-terminal nodes along with the number of instances that reach that node. Let HT denotes the number of internal nodes of the tree T and H_i be an i-th internal node with $|X_i|$ instances, then:

$$Complexity(T) = \sum_{i=1}^{HT} (\frac{|X|}{|X_i|} * C(H_i)), \quad (2)$$

where $C(H_i)$ is the cost of the attributes that make up node H_i. The reason why the total cost of a node increases when the number of instances in a node decreases ($\frac{|X|}{|X_i|}$) is to avoid the overfitting in the lower parts of the hierarchy, as this will further reduce the induction of overgrown trees.

The selection mechanism is based on the linear ranking selection [20]. Additionally, at each iteration, the one individual with the highest fitness in the current population is copied to the next population (elitist strategy). The evolution ends when the fitness of the best individual in the population does not improve within a fixed number of generations or when the maximum number of generations is reached.

4 ACCELERATING EVOHDTREE

Applying the typical parallel data-decomposition technique in the context of biomedical data, where the number of instances is low, may not be efficient [15]. Therefore the EvoHDTree is accelerated using a hybrid approach with a shared address space paradigm (OpenMP) and GPU-based parallelization (see Figure 6). Individuals from the population are spread over the CPU cores using OpenMP threads. For the assigned pool of individuals, each OpenMP thread is responsible for the crossover, mutation, and evaluation algorithm blocks.

Parallelization on the GPU is applied differently. When the mutation operator updates or computes a new splitting note, the local search for the test is parallelized. Each thread on the device is assigned an equal number of relations (called offset) to compute so that it 'knows' which relations of genes it should analyze and where it should store the result. Finding a univariate split is less computationally demanding than finding an RXA gene pair. Therefore in TSP and Weight TSP-like tests, the first attribute is pre-selected by the CPU and, along with the offset and indexes of the instances that reached the mutated node, is sent to the GPU. After all the threads of the block have been completed, the best results from each thread are copied from GPU device memory back to the CPU main memory. Simplified ranking linear selection is used to select the best test that will constitute the split in the mutated node.

5 EXPERIMENTS

In this section, we experimentally validate the proposed EvoHDTree approach and confront its results with popular counterparts.

5.1 Setup

All experiments were performed on cancer-related gene expression-based datasets deposited in NCBI's Gene Expression Omnibus [4] and summarized in Table 1. Typical 10-fold stratified cross-validation was used (although the datasets are well balanced) and the average accuracy score with standard deviation from 20 runs is presented.

For performance reasons regarding other approaches, Relief-F feature selection was used and the number of selected genes was arbitrarily limited to the top 1000. Experiments were conducted on a workstation equipped with an Intel Core i5-8400 CPU, 32 GB RAM, and NVIDIA GeForce GTX 1080 GPU card (8 GB memory, 2 560 CUDA cores). The sequential algorithm was implemented in C++ and the GPU-based parallelization part was implemented in CUDA-C (compiled by nvcc CUDA 10; single-precision arithmetic used)

Table 1: Summary of gene expression datasets: abbreviation with name, number of genes and number of samples.

Datasets	Genes	Samples	Datasets	Genes	Samples
GDS2771	22215	192	GSE10072	22284	107
GSE17920	54676	130	GSE19804	54613	120
GSE25837	18631	93	GSE27272	24526	183
GSE3365	22284	127	GSE6613	22284	105

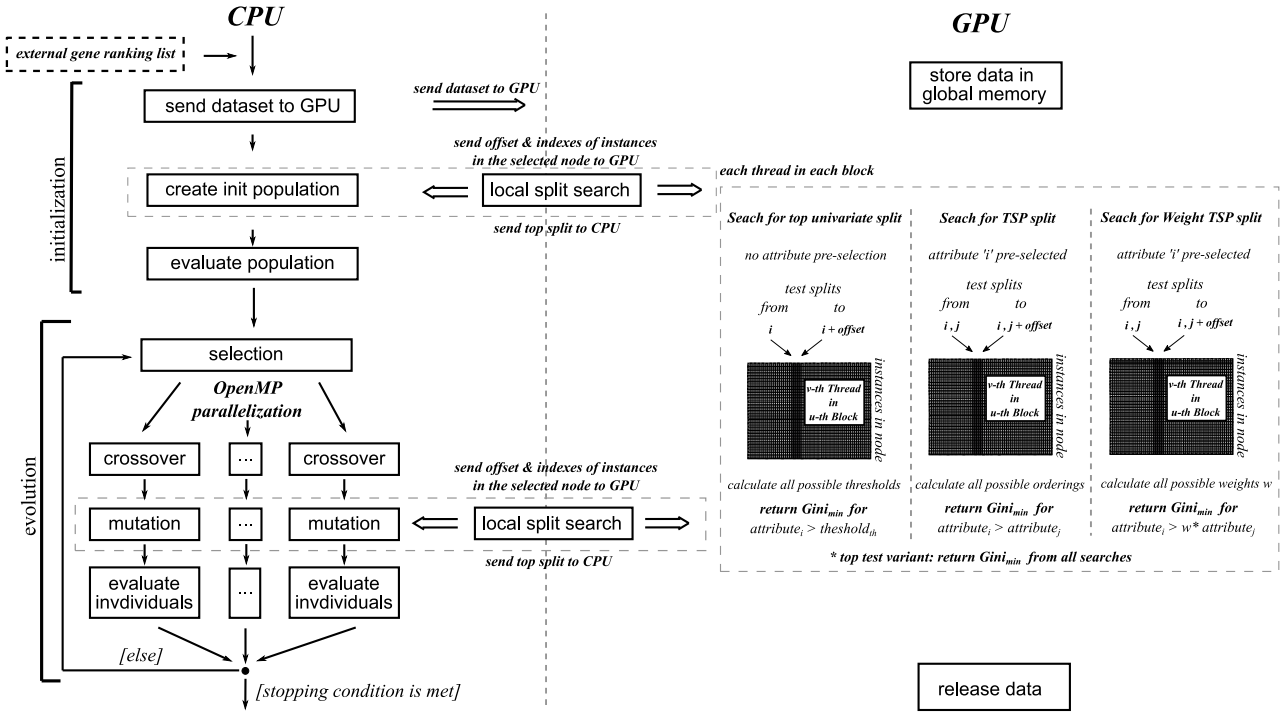

Figure 6: General flowchart of accelerated EvoHDTree approach.

To make a proper comparison, we contrasted the proposed EvoHDTree solution with popular RXA algorithms:

- TSP, TSN, and k-TSP: tests performed using AUERA software [9];
- EvoTSP and REDT results were taken from the publication because we use the same datasets [6] and preprocessing procedure.

To check the impact of the proposed heterogeneous representation, we tested different EvoHDTree configurations:

(1) *EU* - EvoHDTree with tree nodes using only univariate tests;
(2) *ET* - EvoHDTree with tree nodes using only TSP tests
(3) *EW* - EvoHDTree with tree nodes using Weight TSP tests (similar to REDT)

.

Since EvoHDTree is a typical generational EA, parameters such as population size, the maximum number of generations, elitism rate, crossover and mutation probability must be selected before evolution. Table 2 gives a brief summary of the main parameters that were used, however further research is required to tune these settings.

In addition to the typical evolutionary parameters, the EvoHDTree algorithm requires a α setting to control the complexity term. We used $\alpha = 0.1$, which is supported by our previous studies.

5.2 Results

The experimental accuracy results are quite predictable (see Figure 7). All evolutionary approaches outperform the sequential RXA solutions due to the more complex representation - vertical as in *EvoTSP* and horizontal (decision tree) otherwise. It can also be observed that the largest differences between the tested EvoHDTree variants occur when only univariate splits are used. The lower accuracy for the *EU* tree was expected since without RXA nodes the tree is a typical globally induced DT. The improvement in accuracy by inducing a tree with heterogeneous representation is small but consistent and noticeable.

Analysis of the results using the Friedman test showed that there are statistically significant differences between the algorithms (significance level is equal to 0.05) in terms of accuracy. According to Dunn's multiple comparison test EvoHDTree together with REDT managed to significantly outperform all traditional RXA solutions: TSP, TSN, and on some datasets EvoTSP. We believe this is a good result, especially considering that Dunn's test is the most conservative option (less likely to find a significant difference) among all multiple comparison procedures [8].

However, the real improvement is evident when we analyze Table 3. We see that the proposed heterogeneous tree with univariate

Table 2: EvoHDTree default parameters

Basic EA parameters	
Population size:	64 individuals
Elitism rate:	1% of the population
Max generations:	1000
Mutation rate:	90% assigned to the individual
Crossover rate:	10% assigned to the individual

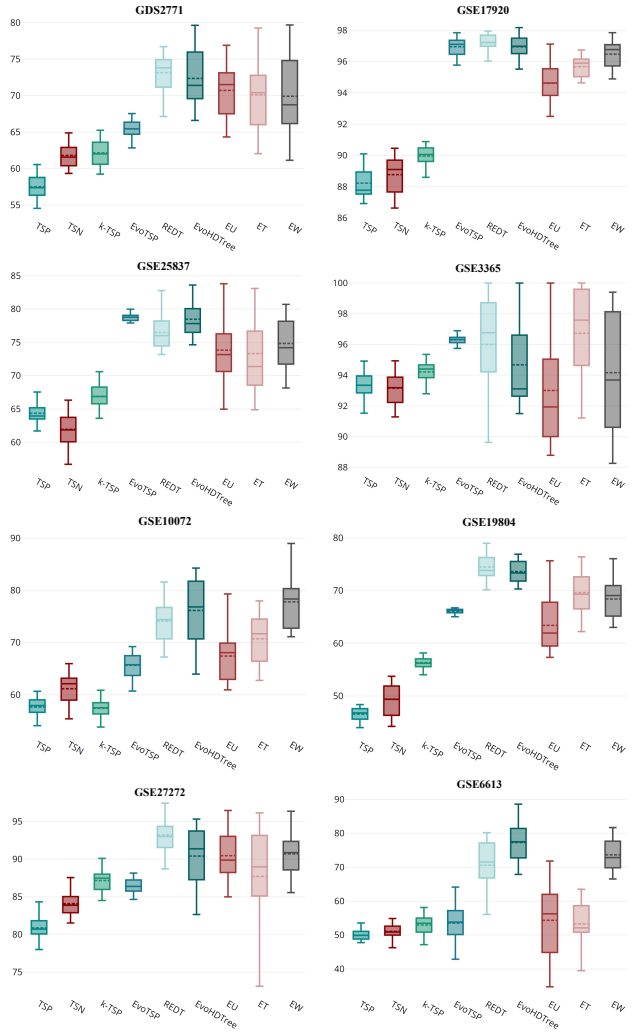

Figure 7: Boxplots of accuracy scores; the dotted line is the mean and the straight line is the median.

and RXA splitting nodes (denoted in the table as EH) induces significantly smaller trees than the competing REDT and other tree variants (* the same statistical assumptions as for testing accuracy). The vast majority of the induced EvoHDTrees contained two or three types of node representation, which may suggest that the founded rules and patterns are more precise.

Speeding up the proposed solution was also one of our priorities. The experimental analysis showed that the most time-consuming part of the algorithm is the local search for tests in internal node splits, which corresponds to more than 97% of evolutionary loop time (the remaining 2.5% is CPU calculation and 0.5% is data transfer and memory allocation related to GPU parallelization). This is not surprising, because:

- the most time-consuming CPU task for evolutionary induced DT (without local search components) is usually the fitness calculation. Since the number of instances in the dataset is small, this operation is relatively fast compared to the calculation of local search components performed on GPU. The proposed OpenMP parallelization, which used 6 cores, managed to speed up the GPU calculations by about 2.5×. However, for assigned population size such parallelization has a small impact on the overall algorithm execution time;
- memory allocation and data transfer time are also almost imperceptible, as only instance indexes together with results are sent between CPU and GPU during the evolutionary loop.

Table 3: Number of genes involved in the tree model, *proposed EvoHDTree is denoted here as EH.

Datasets	k-TSP	EvoTSP	REDT	EH*	EU	ET	EW
GDS2771	10	4.0	8.2	4.5	3.1	6.6	6.1
GSE17920	6	2.1	2.2	1.2	1.0	2.1	2.1
GSE25837	10	2.8	7.3	3.9	3.0	6.2	5.0
GSE3365	10	2.1	2.8	1.1	1.0	2.0	2.1
GSE10072	14	3.1	6.0	5.0	2.9	6.0	6.1
GSE198040	18	2.7	7.9	4.5	3.7	7.0	6.2
GSE27272	14	4.1	3.9	2.9	2.2	4.3	4.0
GSE6613	10	6.1	8.4	6.8	4.4	9.8	8.9
Average	11.5	3.4	5.8	3.7	2.7	5.5	5.0

Table 4: Average time in seconds of a single test search operator calculated on the CPU/GPU (blocks x threads). Tests performed in the root node (with all 192 instances) on dataset GDS2771 with 1000 and 22215 attributes.

	$GDS2771_{1000}$		$GDS2771_{full}$	
Test	CPU	GPU_{16x32}	GPU_{16x32}	GPU_{32x256}
top uni.	12.5	0.033	0.93	0.088
TSP	0.075	0	0.006	0
Weight TSP	14.6	0.037	1.02	0.091
Best	28.88	0.069	1.937	0.177

The most computationally intensive task is to compute the tests in the split. The number of base arithmetic operations is directly related to the test type and the number of instances that reach the node. Table 4 shows how GPU acceleration improves each type of split on the GDS2771 dataset. To simplify the presentation of the calculations, we show the computation times only for the root node (so the number of instances is constant and equal to 192). Since the decomposition strategy on GPU is implemented through parallelization of the attributes, the number of blocks × threads must not exceed their total value. As the potential of parallelization on GPU has not been fully exploited ($GDS2771_{1000}$ has only 1000 attributes), we also include results for the full dataset with 22215 attributes (only on GPU, since CPU takes too much time). On average, the speedup at the root node for $GDS2771_{1000}$ is approximately 400× when using the GPU (16 blocks × 32 threads). The actual speedup for the entire DT is twice smaller because there are fewer instances in the lower parts of the tree, which affects GPU performance. However, for a larger dataset (higher number of instances and attributes), the speedup may be even higher because more threads can be used, as is the case for $GDS2771_{full}$ dataset.

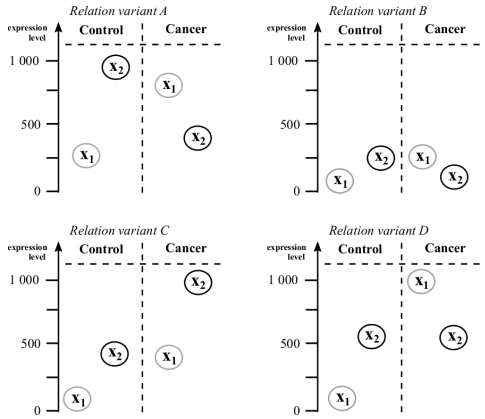

Figure 8: Four possible relations between two genes X_1 and X_2 in control and cancer sample.

6 DISCUSSION

The detected gene-gene interactions strongly depend on the type and capabilities of the algorithm used in the analysis. When the classifier representation is inadequate, the true relation is found either partially or with additional uninformative genes that hinder the understanding and interpretation of the output model. In most cases, such a flaw becomes apparent through the increased size of the generated model rather than lower accuracy. It is well demonstrated in other problems, such as when comparing trees induced by the greedy techniques with ones induced by EA [17].

Suppose hypothetically that the two genes x_1 and x_2 have constant expression values among instances from the same classes. Figure 8 shows four simple scenarios (A), (B), (C), (D) of possible relationships between these two genes in the control and cancer class:

- Variant A: the ordering relationship between x_1 and x_2 in control and cancer class is reversed, and the weight change w is large. All RXA solutions will choose this pair. The pair is significant;
- Variant B: the ordering relation is reversed, but the weight change w between classes is low. The TSP-like solutions will select this pair, but the Weight TSP algorithm may ignore this pair. The pair is insignificant due to the small change in expression level;
- Variant C: the ordering relation is not reversed, but the weight change w is large. The TSP-like solutions will not select this pair but the Weight TSP algorithm may. The pair is significant because of the large change in expression level;
- Variant D: the ordering relation is reversed and the weight change w is large. All RXA-like solutions will choose this pair. The pair is irrelevant because only gene x_1 is relevant and should be selected.

Among all available RXA solutions, only EvoHDTree managed to properly selects (A) and (C) variants and ignored (A) and (D). The importance of destructive impact on the use of non-informative genes in RXA classification models is exhaustively explored in [27]. Finally, it should also be noted that with hierarchical tree structure

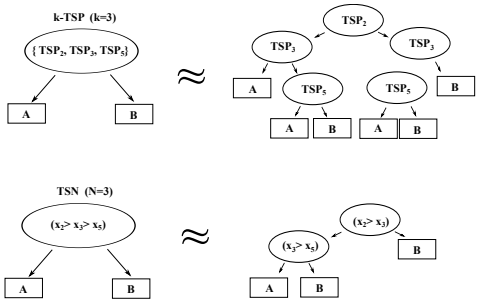

Figure 9: An example of k-TSP and TSN splits and its hierarchical TSP equivalent.

EvoHDTree can theoretically mimic more complex RXA tests like k-TSP and TSN as illustrated in Figure 9. Inducing such a tree is possible, but it is rather unlikely in a real-world scenario.

Due to the paper length we only mention that by decoding gene names with GPL96 platform, we examined the rules generated by the EvoHDTree and their biological relevance for the GDS2771 dataset available in NCBI's GenBank [3]. We found that on average, 30% of the genes used in our model were directly related to lung cancer, and other 30-40% were discussed in several papers in the medical literature. For example, the most popular gene used in EvoHDTree models for GDS2771 dataset was the HBA1 gene (attribute id: 211699). The literature suggests [24] that it has an impact on the survival of lung cancer patients with diabetes mellitus. However, further work is planned with biologists to better understand the gene-gene relationships generated by EvoHDTree.

7 CONCLUSION

This paper introduces a heterogeneous DT with flexible node representation that to some extent can self-adapt to the currently analyzed data. The EvoHDTree algorithm searches globally for the best DT structure, node representation, and splits using a specialized EA. EvoHDTree is able to seek various patterns based on gene-gene interactions along with their raw values. With a parallel implementation including OpenMP and GPU, we can efficiently explore much larger datasets, which is especially important for computationally intensive RXA. The performed experiments show that the proposed solution generates significantly smaller and thus simpler rules with at least similar (if not better) accuracy. We see many promising directions for future research. In particular, we want to adapt EvoHDTree to work with proteomic, metabolomic, and integrated multi-omics data.

8 ACKNOWLEDGMENTS

This project was funded by the Polish National Science Centre and allocated on the basis of decision 2019/33/B/ST6/02386 (first author). The second and third author were supported by the grant WZ/WI-IIT/3/2020 from BUT founded by Polish Ministry of Science and Higher Education

REFERENCES

[1] Rodrigo Coelho Barros, Márcio Porto Basgalupp, André C.P.L.F. De Carvalho, and Alex A. Freitas. 2012. A survey of evolutionary algorithms for decision-tree

induction. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 42, 3 (2012), 291–312. https://doi.org/10.1109/TSMCC.2011.2157494 arXiv:1202.1112

[2] Rodrigo C. Barros, Márcio P. Basgalupp, Aléx A. Freitas, and Andre C.P.L.F. De Carvalho. 2014. Evolutionary design of decision-tree algorithms tailored to microarray gene expression data sets. *IEEE Transactions on Evolutionary Computation* (2014). https://doi.org/10.1109/TEVC.2013.2291813

[3] Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, James Ostell, Kim D. Pruitt, and Eric W. Sayers. 2018. GenBank. *Nucleic Acids Research* (2018). https://doi.org/10.1093/nar/gkx1094 arXiv:1611.06654

[4] Emily Clough and Tanya Barrett. 2016. The Gene Expression Omnibus database. In *Methods in Molecular Biology*. https://doi.org/10.1007/978-1-4939-3578-9_5

[5] Marcin Czajkowski, Marek Grześ, and Marek Kretowski. 2014. Multi-test decision tree and its application to microarray data classification. *Artificial Intelligence in Medicine* 61, 1 (2014), 35–44. https://doi.org/10.1016/j.artmed.2014.01.005

[6] Marcin Czajkowski, Krzysztof Jurczuk, and Marek Kretowski. 2020. Generic Relative Relations in Hierarchical Gene Expression Data Classification. In *Parallel Problem Solving from Nature – PPSN XVI*, Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann (Eds.). Springer International Publishing, Cham, 372–384.

[7] Marcin Czajkowski and Marek Kretowski. 2019. Decision tree underfitting in mining of gene expression data. An evolutionary multi-test tree approach. *Expert Systems with Applications* 137 (2019), 392–404. https://doi.org/10.1016/j.eswa.2019.07.019

[8] Janez Demsar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research 7* (2006). https://doi.org/10.1016/j.jecp.2010.03.005 arXiv:arXiv:1011.1669v3

[9] John C. Earls, James A. Eddy, Cory C. Funk, Younhee Ko, Andrew T. Magis, and Nathan D. Price. 2013. AUREA: An open-source software system for accurate and user-friendly identification of relative expression molecular signatures. *BMC Bioinformatics* (2013). https://doi.org/10.1186/1471-2105-14-78

[10] James A. Eddy, Jaeyun Sung, Donald Geman, and Nathan D. Price. 2010. Relative expression analysis for molecular cancer diagnosis and prognosis. https://doi.org/10.1177/153303461000900204

[11] Donald Geman, Christian D'Avignon, Daniel Q. Naiman, and Raimond L. Winslow. 2004. Classifying Gene Expression Profiles from Pairwise mRNA Comparisons. *Statistical Applications in Genetics and Molecular Biology* (2004). https://doi.org/10.2202/1544-6115.1071 arXiv:NIHMS150003

[12] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. 2015. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing* 34 (2015), 286–300. https://doi.org/10.1016/j.asoc.2015.04.061

[13] Xin Huang, Xiaohui Lin, Lina Zhou, and Benzhe Su. 2018. Analyzing omics data by pair-wise feature evaluation with horizontal and vertical comparisons. *Journal of Pharmaceutical and Biomedical Analysis* (2018). https://doi.org/10.1016/j.jpba.2018.04.052

[14] Laurent Hyafil and Ronald L. Rivest. 1976. Constructing optimal binary decision trees is NP-complete. *Inform. Process. Lett.* 5, 1 (1976), 15–17. https://doi.org/10.1016/0020-0190(76)90095-8

[15] Krzysztof Jurczuk, Marcin Czajkowski, and Marek Kretowski. 2021. Multi-GPU approach to global induction of classification trees for large-scale data mining. *Applied Intelligence* 137 (2021), 392–404. https://doi.org/10.1007/s10489-020-01952-5

[16] S. B. Kotsiantis. 2013. Decision trees: A recent overview. *Artificial Intelligence Review* 39, 4 (2013), 261–283. https://doi.org/10.1007/s10462-011-9272-4

[17] Marek Kretowski. 2019. *Parallel Computations for Evolutionary Induction*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-21851-5_8

[18] Win Tsung Lo, Yue Shan Chang, Ruey Kai Sheu, Chun Chieh Chiu, and Shyan Ming Yuan. 2014. CUDT: A CUDA based decision tree algorithm. *Scientific World Journal* (2014). https://doi.org/10.1155/2014/745640

[19] Andrew T. Magis and Nathan D. Price. 2012. The top-scoring 'N' algorithm: a generalized relative expression classification method from small numbers of biomolecules. *BMC Bioinformatics* (2012). https://doi.org/10.1186/1471-2105-13-227

[20] Zbigniew Michalewicz. 1996. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. https://doi.org/10.2307/2669583

[21] J. R. Quinlan. 1992. Learning With Continuous Classes. World Scientific, 343–348.

[22] Nimrod Rappoport and Ron Shamir. 2018. Multi-omic and multi-view clustering algorithms: review and cancer benchmark. *Nucleic Acids Research* 46, 20 (10 2018), 10546–10562. https://doi.org/10.1093/nar/gky889

[23] L Rokach and O Maimon. 2005. Top-down Induction of Decision Trees Classifiers - a Survey. *Trans. Sys. Man Cyber Part C* 35, 4 (2005), 476–487.

[24] Katie E. Rollins, Krishna K. Varadhan, Ketan Dhatariya, and Dileep N. Lobo. 2016. Systematic review of the impact of HbA1c on outcomes following surgery in patients with diabetes mellitus. https://doi.org/10.1016/j.clnu.2015.03.007

[25] Aik Choon Tan, Daniel Q. Naiman, Lei Xu, Raimond L. Winslow, and Donald Geman. 2005. Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics* (2005). https://doi.org/10.1093/bioinformatics/bti631

[26] Giulia Vilone and Luca Longo. 2020. Explainable Artificial Intelligence: a Systematic Review. arXiv:cs.AI/2006.00093

[27] Kaimin Wu, Xiaofei Nan, Yumei Chai, Liming Wang, and Kun Li. 2016. DTSP-V: A trend-based Top Scoring Pairs method for classification of time series gene expression data. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 1787–1794. https://doi.org/10.1109/BIBM.2016.7822790