






# Relative Expression Classification Tree. A Preliminary GPU-Based Implementation

Marcin Czajkowski<sup>(✉)</sup> , Krzysztof Jurczuk , and Marek Kretowski 

Faculty of Computer Science, Bialystok University of Technology, Wiejska 45a,  
15-351 Bialystok, Poland  
{m.czajkowski,k.jurczuk,m.kretowski}@pb.edu.pl

**Abstract.** The enormous amount of omics data generated from high-throughput technologies has brought an increased need for computational tools in biological analyses. Among the algorithms particularly valuable are those that enhance understanding of human health and genetic diseases. Relative eXpression Analysis (RXA) is a powerful collection of computational methods for analyzing genomic data. It finds relationships in a small group of genes by focusing on the relative ordering of their expression values. In this paper, we propose a Relative eXpression Classification Tree (RXCT) which extends major variants of RXA solutions by finding additional hierarchical connections between sub-groups of genes. In order to meet the enormous computational demands we designed and implemented a graphic processing unit (GPU)-based parallelization. The GPU is used to perform a parallel search of the gene groups in each internal node of the decision tree in order to find locally optimal splits. Experiments carried out on 8 cancer-related gene expression datasets show that the proposed approach allows exploring much larger solution space and finding more accurate interactions between the genes. Importantly, patterns in predictive structures are kept comprehensible and may have direct applicability.

**Keywords:** Relative Expression Analysis (RXA) · Decision trees · GPU · CUDA

## 1 Introduction

Rapid growth and the popularity of high-throughput technologies cause a massive amount of gene expression datasets to become publicly accessible [19]. In the literature, we may find a good number of supervised machine learning approaches for genomic classification. Among the most popular ones, we could mention the support vector machines, neural networks or random forests. Most of currently applied methods provide ‘black box’ classification that usually involves many genes combined in a highly complex fashion and achieves high predictive performance. However, there is a strong need for ‘white box’, comprehensive decision models which may actually help in understanding and identifying casual relationships between specific genes [2, 5]. The popular ones, like the decision trees

(DTs) which have a long history in predictive modeling [10], result in insufficient accuracy [2] when applied to gene expression data. One of the problem specific alternatives is the Relative Expression Analysis (RXA) which is a powerful collection of easily interpretable computational methods. It was designed to analyze genomic data and plays an important role in biomarker discovery and gene expression data classification. It focuses on finding interactions among a small collections of genes and studies the relative ordering of their expression rather than their raw values.

In this paper, we want to merge the strength of RXA with DTs. We propose a new hybrid solution called Relative eXpression Classification Tree (RXCT) that induces DT with the splitting rules built by the RXA methodology. In order to overcome the enormous computational complexity we designed and implemented a graphic processing unit (GPU)-based parallelization of the RXA search. Finally, we added a few changes to the RXA algorithm in order to improve speed and to enable potential multi-class prediction.

This paper is organized as follows. Section 2 provides our motivation and a brief background on RXA, DTs and GPGPU parallelization. Section 3 describes in details our hybrid RXA solution and its GPU-based implementation. Next, an experimental validation is performed on real-life datasets and in the last section, the paper is concluded and possible future works are outlined.

## 2 Background

Gene expression data is very challenging for computational tools and mathematical modelling. Traditional solutions often fail due to the high ratio of features to observations as well as genes redundancy. Therefore, there is a need for new methods to be proposed to extract significant and meaningful rules from the genomic data.

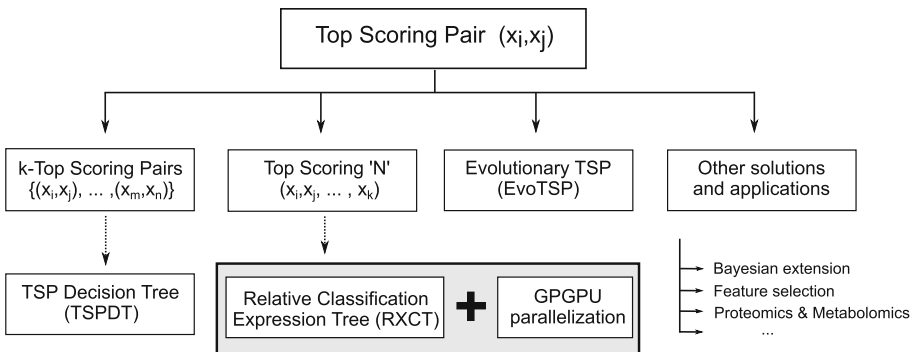


Fig. 1. The general taxonomy of the family of RXA

## 2.1 Algorithms for Relative Expression Analysis

Among many recent algorithms designed for the gene expression data classification, RXA methods are gaining popularity. The RXA taxonomy that includes the main development paths is illustrated in Fig. 1. A Top-Scoring Pair (TSP) is the first and the most popular RXA solution proposed by Donald Geman [8]. It uses a pairwise comparison of gene expression values and searches for a pair of genes with the highest rank. The  $k$ -TSP algorithm [18] is one of the first extensions of the TSP solution. It focuses on increasing the number of pairs in the prediction model and applies no more than  $k$  disjoint gene pairs with the highest score, where the parameter  $k$  is determined by the internal cross-validation. This method was later combined with a decision tree in algorithm called TSPDT [4]. In this system each non-terminal node of the tree divides instances according to a splitting rule that is based on TSP or  $k$ -TSP criteria.

Different approaches for the TSP extension focus on the relationships between more than two genes. Top Scoring Triplet (TST) [11] and Top Scoring N (TSN) [14] analyze ordering relationships between several genes, however, the general concept of TSP is retained. One of the first heuristic method applied to RXA was the evolutionary algorithm called EvoTSP [5] where the authors proposed an evolutionary search for the  $k$ -TSP and TSN-like rules. Performed experiments showed that evolutionary search is a good alternative to the traditional RXA algorithms. Finally, there are many variations of the TSP-family solutions that propose new ways of ranking the gene pairs.

## 2.2 Decision Trees

Decision trees [10] are one of the main techniques for discriminant analysis in knowledge discovery. The success of the tree-based approach can be explained by its ease of use, speed of classification and effectiveness. In addition, the hierarchical structure of the tree, where appropriate tests are applied successively from one node to the next, closely resembles the human way of making decisions.

However, there are not so many new solutions in the literature that focus on the classification of gene expression data with comprehensive DT models. Nowadays, much more interest is given in trees as sub-learners of an ensemble learning approach, such as Random Forests [3]. These solutions alleviate the problem of low accuracy by averaging or adaptive merging of multiple trees. However, when modeling is aimed at understanding basic processes, such methods are not so useful because they generate more complex and less understandable models.

## 2.3 GPGPU Parallelization

A general-purpose computation on GPUs (GPGPU) stands for the use of graphics hardware for generic problems. In the literature, we can find a few systems where GPU-based parallelization of the greedy induction of DTs was examined. One of the propositions was a CUDT [12] that parallelized the top-down induction process. In each internal node, in order to find the best locally optimal splits,

the attributes are processed in parallel. With this approach, the authors managed to reduce the induction time of a typical decision tree from 5 to 55 times when compared with the traditional CPU version. The GPU was also introduced in GDT system which parallelizes evolutionary induction of DTs [9].

In case of RXA there exists also research considering GPU parallelization. In [13] authors managed to speed up calculations of TSP and TST by two orders of magnitude. The tests for higher number of related genes were also performed [14], but only when the total number of attributes was heavily reduced by the feature selection.

## 2.4 Motivation and Contribution

In this paper we propose a new approach that combines the strength of DTs with RXA. We are motivated by the fact that single DT represents a white-box approach, and improvements to such models have considerable potential for scientific modeling of the underlying processes in a genomic research.

Proposed contribution is inspired with the existing system called TSPDT [4] which also uses RXA concept in DT nodes. The main drawback of TSPDT as well as all traditional RXA algorithms is the enormous computational complexity that equals  $O(T * k * N^Z)$ , where  $T$  is the size of DT,  $k$  is the number of top-scoring groups,  $N$  is the number of analyzed genes and  $Z$  is the size of a group of genes which ordering relationships are searched. Sequential calculation of all possible gene pairs or gene groups strongly limits the number of genes and inter-relations that can be analyzed by the algorithm. This is the reason why the TSPDT focuses only on TSP and k-TSP variants and the algorithm need to be preceded by the feature selection step.

There are several major differences between the TSPDT and proposed RXCT solutions in terms of both performance and functionality:

- with the GPU parallelization the RXCT is capable of inducing the RXA-based decision tree much faster, even on entire datasets (without feature selection step);
- RXCT extends the inter-gene relations, allows testing higher number of related genes and has additional optimizations considering strict order relations;
- RXCT in contrast to TPSDT can be applied to multi-class datasets due to the different splitting rule.

## 3 Relative eXpression Classification Tree

The proposed solution's overall structure is based on a typical top-down induced [16] binary classification tree. The greedy search starts with the root node, where the locally optimal split (test) applies RXA. Then the training instances are redirected to the newly created nodes and this process is repeated for each node until the stop condition is met. Currently, we do not apply any form of pruning

due to the small sample sizes, however it should be considered in the future to improve the generalizing power of the predictive model.

The general flowchart of our GPU-accelerated RXCT is illustrated in Fig. 2. It can be seen that the DT induction is run in a sequential manner on a CPU, and the most time-consuming operation (split search) is performed in parallel on a GPU. This way, the parallelization does not affect the behavior of the original algorithm.

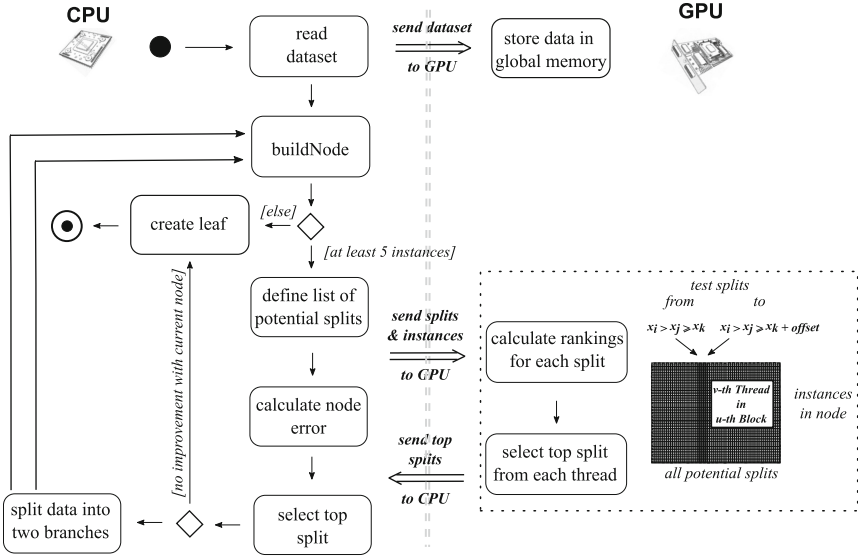


Fig. 2. General flowchart of a GPU-accelerated RXCT

### 3.1 RXCT Split Search

Each internal node contains information about a relation of two or three genes that is later used to constitute the split. The basic idea to analyze relations within a single instance is similar to TSP and TST solutions, however, there are some differences in strict ordering as well as in ranking of gene collections.

Let  $x_i$ ,  $x_j$  and  $x_k$  be the expression values of three genes  $i, j, k$  from available set of  $N$  genes. However, in contrast to existing solutions, we allow triplet reduction to a pair if  $j$  equals  $k$ . RXA can be directly applied to binary classification problems as it scores relations using probabilities of assigning instances to one of two classes. To allow application RXCT to analyze multi-class datasets, we have chosen to use Gini index [1] which is well known splitting criterion for DT. It is also slightly computationally faster than the gain ratio as there are no LOG functions.

The Gini impurity is calculated for all pairs or triplets defined by the relation:  $x_i > x_j \geq x_k$  where  $i \neq j$ . The non-strict relation between the second and third gene allows us to test both TSP (if  $j = k$ ) and TST ( $j \neq k$ ) variants. However, in contrast to RXA-based solutions we do not check all possible orderings but limit them to those that meet the assumption  $i > j \geq k$ . This limitation does not affect the results (if one of the relation is opposite then the resulting branches are swapped) but reduces the calculations twice in case of TSP and six times in case of TST. The triplet with the highest Gini impurity becomes the split and in case of a draw, a secondary ranking that bases on real-value genes expression differences is used [18].

### 3.2 GPU-Based Approach

The RXA methods like TSP and TST exhibit characteristics that make them ideal for a GPU implementation as there is no data dependence between individual scores. As it is illustrated in Fig. 2, the dataset is first copied from the CPU main memory to the GPU device memory so each thread block can access it. Typical sizes of gene expression datasets are not large and range from a few to several hundred megabytes, thus there is no problem to fit the entire set into a single GPU memory.

In each node,  $N^3$  of possible relations  $x_i > x_j \geq x_k$  need to be processed and scored. Each thread on the device is assigned an equal amount of relations (called offset) to compute (see Fig. 2). This way each thread ‘knows’ which relations of genes it should analyze and where it should store the result. However, as it was mentioned in previous section, not all relations need to be calculated (assumption  $i > j \geq k$ ). In addition, number of instances for which the Gini impurity is calculated varies in each tree node - from full set of samples in a root to a few instances in the lower parts of the tree.

Each launched kernel requires not only the relations that will be processed but also: an information about the instances that reach the internal node which runs the kernel; and an empty vector for the results. Within each thread there is no further parallelization: each thread simply loops over the instances that reach the node and calculates the scores to the assigned relations. After all thread blocks have finished, the top results from each threads are copied from the GPU device memory back to the CPU main memory where the top split is selected.

## 4 Experimental Validation

In this section, we present experimental analysis of RXCT predictive performance and confront its results with popular RXA extensions. In addition, we show the speedup achieved with proposed GPU parallelization.

## 4.1 Datasets and Setup

To make a proper comparison with the RXA algorithms, we use the same 8 cancer-related benchmark datasets (see Table 1) that are tested with the EvoTSP solution [5]. Datasets are deposited in NCBI’s Gene Expression Omnibus and summarized in Table 1. A typical 10-fold cross-validation is applied and depending on the system, different tools are used:

- evaluation of TSP, TST, and  $k$ -TSP was performed with the AUERA software [7], which is an open-source system for identification of relative expression molecular signatures;
- EvoTSP results were taken from the publication [5];
- original TSPDT and RXCT implementation are used.

Due to the performance reasons concerning other approaches, the Relief-F feature selection was applied and the number of selected genes was arbitrarily limited to the top 1000. In the experiments, we provide results for the proposed RXCT solution as well as its simplified variant called  $\text{RXCT}_{TSP}$  which uses only TSP-like splits.

**Table 1.** Details of gene expression datasets: abbreviation with name, number of genes and number of instances.

| Datasets     | Genes | Instances | Datasets     | Genes | Instances |
|--------------|-------|-----------|--------------|-------|-----------|
| (a) GDS2771  | 22215 | 192       | (e) GSE10072 | 22284 | 107       |
| (b) GSE17920 | 54676 | 130       | (f) GSE19804 | 54613 | 120       |
| (c) GSE25837 | 18631 | 93        | (g) GSE27272 | 24526 | 183       |
| (d) GSE3365  | 22284 | 127       | (h) GSE6613  | 22284 | 105       |

Experiments were performed on a workstation equipped with Intel Xeon CPU E5-2620 v3 (15 MB Cache, 2.40 GHz), 96 GB RAM and NVIDIA GeForce GTX Titan X GPU card (12 GB memory, 3 072 CUDA cores). We used a 64-bit Ubuntu Linux 16.04.6 LTS as an operating system. The sequential algorithm was implemented in C++ and compiled with the use of gcc version 5.4.0. The GPU-based parallelization part was implemented in CUDA-C [17] and compiled by nvcc CUDA 8.0 [15] (single-precision arithmetic was applied).

## 4.2 Accuracy Comparison to Popular RXA Algorithms

Table 2 summarizes classification performance for the proposed solution and its competitors. From the results we can see that the proposed RXCT solution managed to outperform in average all popular RXA classifiers. Although there are no statistical differences between TSPDT and RXCT in terms of accuracy, the size of the trees generated by RXCT is significantly smaller (Friedman test

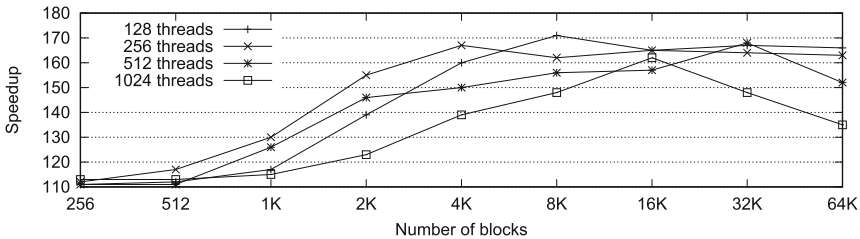
and the corresponding Dunn’s multiple comparison test are applied, p-value = 0.05 [6]).

There are two factors that may explain such good results achieved by the RXCT classifier. First of all, we use TST-like variant in each node instead of TSP split so more advanced relations are searched. Next, the additional experiments (not shown) revealed that in the case of DT using the original TSP rank to split the data, it returns worse results than when using one of the standard DT splitting criteria like Gini index or gain ratio. It also explains why the  $RXCT_{TSP}$  using a single TSP in each node can compete with TSPDT which uses more complex split ( $k$  - top pairs).

**Table 2.** Comparison of RXCT with top-scoring algorithms, including accuracy and the size of the classifier’s model. The best accuracy for each dataset is bolded.

| Dataset | TSP  | TST  | k-TSP |                   | EvoTSP      |                   | TSPDT       |                   | $RXCT_{TSP}$ |                   | RXCT         |                   |
|---------|------|------|-------|-------------------|-------------|-------------------|-------------|-------------------|--------------|-------------------|--------------|-------------------|
|         | acc. | acc. | acc.  | size <sup>1</sup> | acc.        | size <sup>2</sup> | acc.        | size <sup>3</sup> | acc.         | size <sup>4</sup> | acc.         | size <sup>5</sup> |
| (a)     | 57.2 | 61.9 | 62.9  | 10                | 65.6        | 4.0               | 60.1        | 16.4              | <b>68.2</b>  | 15.1              | 67.2         | 10.5              |
| (b)     | 88.7 | 89.4 | 90.1  | 6                 | 96.5        | 2.1               | 98.2        | 2.0               | 95.1         | 2.0               | <b>100.0</b> | 2.0               |
| (c)     | 64.9 | 63.7 | 67.2  | 10                | <b>78.1</b> | 2.8               | 72.3        | 6.8               | 74.6         | 6.8               | 77.7         | 5.0               |
| (d)     | 93.5 | 92.8 | 94.1  | 10                | <b>96.2</b> | 2.1               | 88.3        | 3.0               | 90.0         | 2.8               | 91.6         | 2.0               |
| (e)     | 56.0 | 60.5 | 58.4  | 14                | 66.9        | 3.1               | 68.1        | 5.7               | 68.6         | 6.0               | <b>73.2</b>  | 4.3               |
| (f)     | 47.3 | 50.1 | 56.2  | 18                | 66.2        | 2.7               | <b>67.2</b> | 11.9              | 60.6         | 12.6              | 60.0         | 9.2               |
| (g)     | 81.9 | 84.2 | 87.2  | 14                | 86.1        | 4.1               | 88.6        | 4.3               | 85.0         | 4.1               | <b>89.7</b>  | 3.3               |
| (h)     | 49.5 | 51.7 | 55.8  | 10                | 53.6        | 6.1               | 59.6        | 8.0               | 54.3         | 7.0               | <b>70.4</b>  | 5.4               |
| Average | 67.4 | 69.3 | 71.5  | 11.5              | 76.2        | 2.7               | 75.3        | 7.2               | 74.6         | 6.9               | <b>78.7</b>  | 5.1               |

- <sup>1</sup> Avg. number of k-TSP pairs ( $k \leq 18$ );
- <sup>2</sup> Avg. number of unique genes;
- <sup>3</sup> Avg. number of tree nodes, in each node no more than  $k$  ( $k \leq 5$ ) TSP;
- <sup>4</sup> Avg. number of tree nodes, each node with single TSP;
- <sup>5</sup> Avg. number of tree nodes, each node with single TST.



**Fig. 3.** Effect of block’s and thread’s number on the algorithm speedup.



### 4.3 Speed Improvement with GPGPU Approach

Even with the feature selection step, the number of possible relations for which the GPU needs to calculate score is very high. For example, for 1000 genes there is  $10^9$  possible split expression rules in each tree node whereas if we take the full dataset, e.g. GSE17920, this number drastically increases to  $1.63 * 10^{14}$ . The high number of possible kernel tasks requires finding optimal number of threads and blocks which will perform the calculations. Figure 3 illustrates how amount of threads impacts on the GPU-accelerated RXCT speedup in comparison to its sequential version averaged on all datasets. We see that by increasing at some point the number of blocks the speedup rises. This suggests that processing too many possible relations by each thread (high load) slows down the parallelization. Decreasing the offset value improves load balancing and thus the overall RXCT speedup.

In this section we also present the times achieved by popular RXA solutions. We perform direct time comparison between the TSPDT solution and two variants of the RXCT algorithm: original RXCT and  $RXCT_{TSP}$  version which is at some point similar to TSPDT. Table 3 shows the times and speedups of the sequential and parallel versions of RXCT for all the datasets. Alike in Sect. 4.2 the number of attributes in the datasets was limited to 1000. As expected, the algorithms which compare only two features perform much faster than the ones which analyze triplets. At the same time, the solutions that use hierarchical structures take much more time as they run multiple searches of top groups in each non-terminal node.

**Table 3.** Induction times of popular RXA solutions (in seconds). Impact of the GPGPU approach on the RXCT and  $RXCT_{TSP}$  algorithm is also included.

| Dataset | TSP  | TST  | TSPDT | seqRXCT <sub>TSP</sub> | RXCT <sub>TSP</sub> |         | seqRXCT | RXCT |         |
|---------|------|------|-------|------------------------|---------------------|---------|---------|------|---------|
|         | Time | Time | Time  | Time                   | Time                | Speedup | Time    | Time | Speedup |
| (a)     | 3.01 | 999  | 325   | 15.5                   | 0.067               | 232     | 4071    | 24.1 | 169     |
| (b)     | 1.92 | 672  | 32.3  | 7.30                   | 0.030               | 247     | 1842    | 11.9 | 155     |
| (c)     | 1.49 | 1005 | 113   | 5.13                   | 0.021               | 241     | 1276    | 8.04 | 158     |
| (d)     | 1.95 | 590  | 76.8  | 4.87                   | 0.017               | 314     | 1314    | 8.33 | 158     |
| (e)     | 1.84 | 676  | 127   | 1.83                   | 0.013               | 253     | 519     | 3.15 | 165     |
| (f)     | 2.07 | 653  | 292   | 16.9                   | 0.068               | 250     | 3444    | 23.3 | 148     |
| (g)     | 2.99 | 511  | 106   | 3.33                   | 0.024               | 276     | 857     | 5.50 | 156     |
| (h)     | 1.78 | 592  | 151   | 6.56                   | 0.031               | 250     | 1575    | 9.76 | 161     |
| Average | 2.13 | 712  | 152   | 7.68                   | 0.031               | 250     | 1862    | 21.9 | 159     |

We have also performed experiments with datasets containing all attributes. Total time required to process all 8 datasets takes over 2 weeks by TSPDT, 16h by seqRXCT<sub>TSP</sub> and only 3min by RXCT<sub>TSP</sub> ( $\sim x300$  faster than seqRXCT<sub>TSP</sub> and  $\sim x7000$  faster than TSPDT).

However, titled RXCT solution is much more computationally demanding. The total time required by RXCT equals approximately 20 days which is similar

to the time need by the TSPDT solution. For the seqRXCT version it would take over a decade to induce the same tree.

## 5 Conclusion

In this paper, we introduce a hybrid approach to analyze gene expression data which combines the problem-specific methodology with the popular white-box classifier. The Relative eXpression Classification Tree extends major variants of RXA solutions and is capable of finding interesting hierarchical patterns in sub-groups of genes. In addition, thanks to the GPU-parallelization, we managed to induce the tree in a reasonable time.

We see many promising directions for future research. First of all, there is still a lot of ways to improve the GPU parallelization of RXCT, e.g. load-balancing of tasks based on the number of instances in each node, simultaneous analysis of two branches, better GPU hierarchical memory exploitation. Then, other variants of RXA can be used in each split like  $k$ -TSP,  $k$ -TST or even both. Next, some form of tree post-pruning could be applied, not only to limit the tree size but also to decrease the number of genes used in the splits in order to reduce overfitting and promote more accurate decisions. Finally, we are currently working with biologists and bioinformaticians to better understand the decision rules generated by RXCT and preparing the algorithm to work with protein and metabolic expression databases.

**Acknowledgments.** This work was supported by the grant S/WI/2/18 from Białystok University of Technology founded by Polish Ministry of Science and Higher Education.

## References

1. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group, Belmont (1984)
2. Barros, R.C., Basgalupp, M.P., Freitas, A.A., Carvalho, A.C.: Evolutionary design of decision-tree algorithms tailored to microarray gene expression data sets. *IEEE Trans. Evol. Comput.* **18**(6), 873–892 (2014)
3. Chen, X., Wang, M., Zhang, H.: The use of classification trees in bioinformatics. *Wiley Interdisc. Rev. Data Min. Knowl.* **1**, 55–63 (2011)
4. Czajkowski, M., Kretowski, M.: Top scoring pair decision tree for gene expression data analysis. In: Arabnia, H., Tran, Q.N. (eds.) *Software Tools and Algorithms for Biological Systems*. AEMB, vol. 696, pp. 27–35. Springer, New York (2011). [https://doi.org/10.1007/978-1-4419-7046-6\\_3](https://doi.org/10.1007/978-1-4419-7046-6_3)
5. Czajkowski, M., Kretowski, M.: Evolutionary approach for relative gene expression algorithms. *Sci. World J.* **593503**, 7 (2014)
6. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
7. Earls, J.C., Eddy, J.A., Funk, C.C., Ko, Y., Magis, A.T., Price, N.D.: AUREA: an open-source software system for accurate and user-friendly identification of relative expression molecular signatures. *BMC Bioinformatics* **14**, 78 (2013). <https://doi.org/10.1186/1471-2105-14-78>

8. Geman, D., d'Avignon, C., Naiman, D.Q., Winslow, R.L.: Classifying gene expression profiles from pairwise mRNA comparisons. *Stat. Appl. Genet. Mol. Biol.* **3** (2004). Article no. 19. <https://doi.org/10.2202/1544-6115.1071>
9. Jurczuk, K., Czajkowski, M., Kretowski, M.: Evolutionary induction of a decision tree for large-scale data: a GPU-based approach. *Soft Comput.* **21**(24), 7363–7379 (2016). <https://doi.org/10.1007/s00500-016-2280-1>
10. Kotsiantis, S.B.: Decision trees: a recent overview. *Artif. Intell. Rev.* **39**(4), 261–283 (2013). <https://doi.org/10.1007/s10462-011-9272-4>
11. Lin, X., et al.: The ordering of expression among a few genes can provide simple cancer biomarkers and signal BRCA1 mutations. *BMC Bioinformatics* **10**, 256 (2009). <https://doi.org/10.1186/1471-2105-10-256>
12. Lo, W.T., Chang, Y.S., Sheu, R.K., Chiu, C.C., Yuan, S.M.: CUDT: a CUDA based decision tree algorithm. *Sci. World J.* **745640**, 12 (2014)
13. Magis, A.T., Earls, J.C., Ko, Y., Eddy, J.A., Price, N.D.: Graphics processing unit implementations of relative expression analysis algorithms enable dramatic computational speedup. *Bioinformatics* **27**(6), 872–873 (2011)
14. Magis, A.T., Price, N.D.: The top-scoring ‘N’ algorithm: a generalized relative expression classification method from small numbers of biomolecules. *BMC Bioinformatics* **13**, 227 (2012). <https://doi.org/10.1186/1471-2105-13-227>
15. NVIDIA Developer Zone - CUDA Toolkit Documentation. <https://docs.nvidia.com/cuda/cuda-c-programming-guide/>
16. Rokach, L., Maimon, O.Z.: Top-down induction of decision trees classifiers - a survey. *IEEE Trans. SMC Part C* **35**(4), 476–487 (2005)
17. Storti, D., Yurtoglu, M.: *CUDA for Engineers: An Introduction to High-Performance Parallel Computing*. Addison-Wesley, New York (2016)
18. Tan, A.C., Naiman, D.Q.: Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics* **21**, 3896–3904 (2005)
19. Taminau, J., et al.: Unlocking the potential of publicly available microarray data using inSilicoDb and inSilicoMerging R/Bioconductor packages. *BMC Bioinformatics* **13**, 335 (2012). <https://doi.org/10.1186/1471-2105-13-335>