Hybrid Parallelization of Evolutionary Model Tree Induction

Marcin Czajkowski(✉), 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. This paper illustrates a parallel implementation of evolutionary induction of model trees. An objective is to demonstrate that such evolutionary evolved trees, which are emerging alternatives to the greedy top-down solutions, can be successfully applied to large scale data. The proposed approach combines message passing (MPI) and shared memory (OpenMP) paradigms. This hybrid approach is based on a classical master-slave model in which the individuals from the population are evenly distributed to available nodes and cores. The most time consuming operations like recalculation of the regression models in the leaves as well as the fitness evaluation and genetic operators are executed in parallel on slaves. Experimental validation on artificial and real-life datasets confirms the efficiency of the proposed implementation.

Keywords: Evolutionary algorithms · Decision trees · Parallel computing · MPI · OpenMP

1 Introduction

Decision trees are one of the most known prediction techniques in data mining [13]. Their popularity can be explained by their ease of application, fast operation and effectiveness. Regression and model trees [12] may be considered as a variant of decision trees, designed to approximate real-valued functions instead of being used for classification tasks. Main difference between regression trees and model trees is that, for the latter, constant value in the terminal nodes is replaced by the regression planes.

Despite fifty years of research on the decision trees, a few open issues still remain [16]. To mitigate some of them (e.g. application of heuristics such as greedy algorithms where locally optimal decisions are made in each tree node) evolutionary algorithms (EAs) are applied to the decision tree induction [1]. The strength of such approach lies in global search for splits and predictions, and it results in higher accuracy and smaller output trees in comparison to popular top-down decision tree inducers [19]. However, one of the major drawbacks associated with the application of EAs is relatively high tree induction time, especially for large scale data. In the recent survey [1] on the evolutionary induction of decision

L. Rutkowski et al. (Eds.): ICAISC 2016, Part I, LNAI 9692, pp. 370–379, 2016.
DOI: 10.1007/978-3-319-39378-0_32

trees, authors put on the first place in the future trends the need of speeding up the evolutionary induction.

Fortunately EAs are naturally prone to parallelism and the process of artificial evolution can be implemented in various ways [5]. There are three main strategies that have been studied for the parallelization and/or distribution of the computation effort in EAs:

- master-slave paradigm [3] - parallelization of the most time consuming operations in each evolutionary loop (usually fitness recalculation);
- cellular (fine-grained) algorithm [15] - redistribution of single individuals which can communicate only with the nearest individuals (for the selection and reproduction) based on the defined neighborhood topology;
- island (coarse-grained) model [2] - grouping individuals into sub-populations that are distributed between islands and can evolve independently.

In this paper, a parallelization with a hybrid MPI+OpenMP approach (which is considered as providing better efficiency than e.g. pure MPI version [20]) is proposed to the evolutionary induction of regression and model trees. It is applied to a system called Global Model Tree (GMT) [6] that is used in many real-life applications [7]. The main objectives of this work are to accelerate the GMT system and to enable efficient evolutionary induction of decision trees on large scale data. Previously, we managed to apply similar idea for parallelizing the evolutionary induction of classification trees [8]. To the best of our knowledge, proposed solution is the first research on parallelization the evolutionary induction of regression or model trees, as there have been no such attempts in the literature.

This paper is organized as follows. The next section provides a brief background on the GMT system. Section 3 describes our approach for parallel implementation of evolutionary tree induction in detail. Section 4 presents experimental validation of the proposed solution on artificial and real-life datasets. In the last section, the paper is concluded and possible future works are sketched.

2 Global Model Tree System

The general structure of the GMT system follows a typical EA [17] framework with an unstructured population and a generational selection. Model trees are represented in their actual form as univariate trees, so each split in the internal node is based on a single attribute. If the attribute is nominal, at least one value is associated with each branch (inner disjunction). In case of a continuous-valued attribute, the typical inequality tests are applied. Initial individuals are constructed using greedy strategies [19] on random subsamples of the training data, and the tests in internal nodes are searched on random subsets of attributes. Each tree leaf contains a multivariate linear regression model that is constructed using the standard regression technique [18] with objects associated with that node. A dependent variable (y) is explained by the linear combination of multiple independent variables $\{x_1, x_2, \ldots, x_q\}$:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_q * x_q, \tag{1}$$

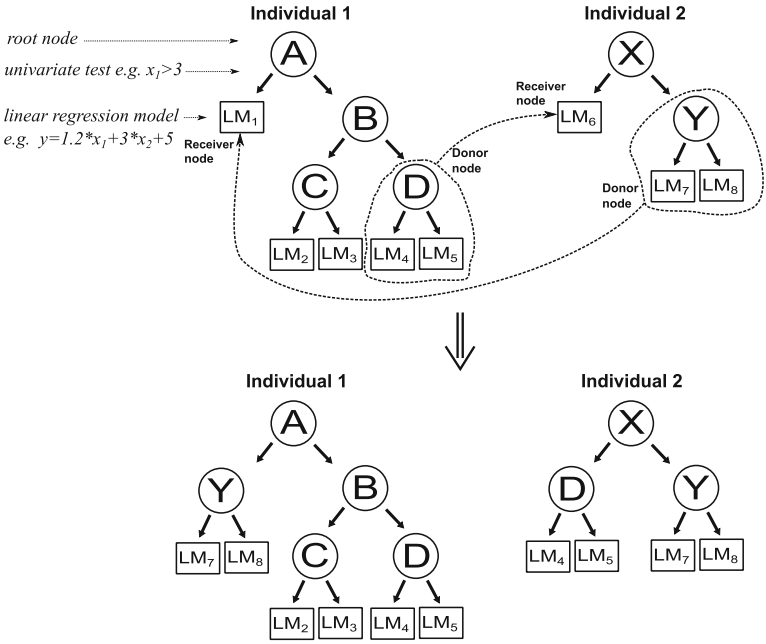

Fig. 1. Crossover between two individuals and the resulting offspring. Each individual has one donor node and one receiver node.

where q is the number of independent variables and β_i $(0 \leq i \leq q)$ are fixed coefficients that minimize the sum of the squared residuals of the model. Additionally, in every node information about training instances associated with the node is stored. This enables the algorithm to perform more efficiently local structure and tests modifications during applications of genetic operators.

Tree-based representation requires developing specialized genetic operators corresponding to classical mutation and crossover. Application of the operators can modify the tree structure, tests in internal nodes, and models in the leaves. The mutation operator makes random changes in some places of the selected individuals. The crossover operator attempts to combine elements of two existing individuals (parents) to create a new solution. The GMT system performs various specialized variants of genetic operators. An example of asymmetric crossover where the subtree of the first/second individual is replaced by a new one that was duplicated from the second/first individual is illustrated in Fig. 1. The replaced subtree starts in the node denoted as a receiver, and the duplicated subtree starts in the node denoted as a donor. It is preferred that the receiver node has a high error per instance and it is replaced by the donor node, which should have a small value of error as it is duplicated. The application of this particular variant is more likely to improve affected individuals because, with higher probability, the good nodes are duplicated and replace the weak nodes. Several variants of crossover and mutations were proposed in [6], e.g.:

- finding a new test or modification of the existing one (shift threshold on continuous attribute, re-grouping nominal attribute values) in the internal node;
- pruning the internal node and transforming it into the leaf with a new multivariate linear regression model;
- expanding the leaf into the internal node;
- replacing one of the following: subtree, branch, node, or test between two affected individuals;
- modification of the linear regression models in the leaves (add, remove, or change attributes).

Successful application of any operator results in a necessity for relocation of the instances between tree parts rooted in the modified nodes.

The selection mechanism is based on the ranking linear selection [17] with the *elitist strategy*, which copies the best individual founded so far to the next population. The fitness function measures the performance of the individuals in terms of meeting the problem objective. In the context of decision trees, a direct minimization of the prediction performance measured on the learning set often leads to the over-fitting problem and poor performance on unseen testing instances. Therefore, efficient fitness function should consider not only the predictive error but also the complexity of the tree. In *GMT*, the authors adapt the Bayesian Information Criterion (*BIC*) [21] as a fitness function. The *BIC* fitness is equal to:

$$Fit_{BIC}(T) = -2 * ln(L(T)) + ln(n) * k(T), \tag{2}$$

where $L(T)$ is the tree (T) maximum of likelihood function, $k(T)$ is the complexity term, and n equals the number of instances. The function $L(T)$ is common for regression models [9] and is defined as:

$$ln(L(T)) = -0.5n * [ln(2\pi) + ln(SS_e(T)/n) + 1]. \tag{3}$$

The term $SS_e(T)$ is the sum of squared residuals of the tree T (on the training set).

3 Parallel Implementation of the GMT System

The proposed parallelization of the model tree evolutionary inducer is based on the sequential GMT algorithm for univariate model trees. The general flowchart of our hybrid MPI+OpenMP approach is illustrated in Fig. 2. One can observe that the evolutionary induction is run in a sequential way on a master node and the most time consuming operations (evaluation of the individuals, recalculation of the regression models and genetic operators) are performed in parallel on the available nodes (slaves). This master-slave parallelization approach, where the master distributes the population among the slaves and, finally, it gathers and merges the results, does not affect the results of the induction. Information

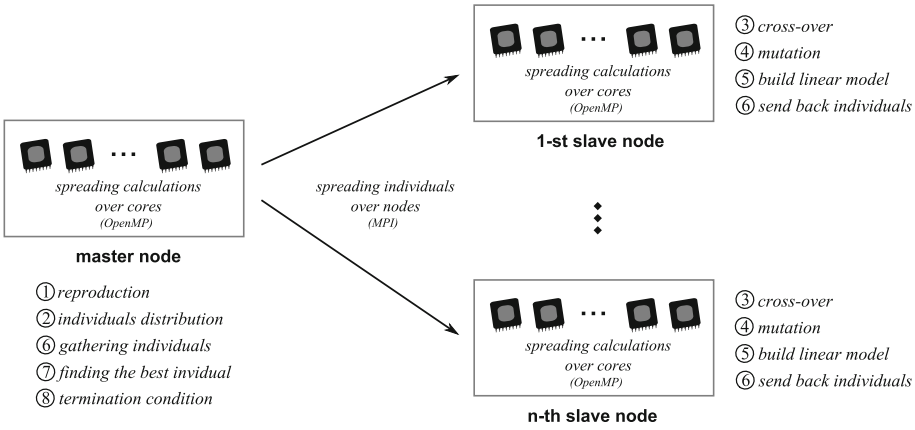

Fig. 2. Hybrid parallel approach of the evolutionary model tree induction

about the location of the training instances is stored in each node of model trees. This way the genetic operators can efficiently obtain the fitness corresponding to the individual [6]. The actual fitness calculation is embedded into the post mutation and crossover processing, when the instances in the affected parts of the tree (or trees) are relocated. This mechanism increases the memory complexity of the induction but significantly reduces its computational complexity. As a consequence, the most time consuming elements of the algorithm are genetic operators and thus are performed in parallel. In addition, each tree node contains information about the regression models. After each successive application of any genetic operator the regression models in the affected leaves are recalculated, which also takes considerable amount of time.

The first level of parallelization the GMT solution (see Fig. 2) applies distributed memory approach where the master node spreads individuals from the population over slave nodes using message-passing strategy [10]. The role of the master node is to perform selection and reproduction (steps (1) and (7)) as well as the verification of termination condition (step (8)). In each evolutionary loop, the master evenly distributes individuals among the slaves (step (2)). To avoid wasting resources, the chunk of population is left on the master which also works as a slave. Migration the individuals between nodes (steps (2) and (6)) is performed with the framework of the message-passing interface (MPI [11]) and requires: packing the tree structures into a flat message; transferring the message between nodes (sending/receiving); and unpacking the message into the corresponding tree. The packed tree structure contains information about its size and the information about each tree node:

- node type (leaf or internal node);
- type and definition of a test (only for internal node);
- definition of the multivariate linear regression model;
- additional statistics (number of instances that reaches the node, prediction error).

To avoid unnecessary unpacking-packing operations (for the trees that will not be selected into the next generation) on the master, the fitness value of the migrated individual is also transferred. However, no information about the objects redistribution in the tree is included in the package. Including such information would strongly increase the size of the package, especially on large scale datasets. Therefore, alike in our previously presented research [8], we recover the redistribution of the instances in those nodes that will be affected by the genetic operators. It is performed on slave nodes which execute the mutation and crossover operations (steps (3) and (4)). If the genetic operator is successful, then there is also a need to reallocate the instances in the sub-trees and rebuild regression models in the leaves (step (5)). Otherwise, the nodes statistics from the received package remain unchanged. It should be emphasis that the GMT system mutates the internal nodes in lower parts of the tree with higher probability. This may enhance a possible speedup of such partial nodes reconstruction as it is expected that the lower parts of the tree held fewer instances that need to be reallocated.

Second level of parallelization that applies shared (OpenMP [4]) memory approach is performed on slave nodes. All the calculations assigned to the slave node are spread over cores which run the algorithm blocks in parallel. Depending on the genetic operator type, each core processes a single individual at a time (in case of mutation) or pairs of affected individuals (in case of crossover) in parallel. All cores within the node operate independently but share the same memory resources. In contrast to the distributed memory approach, no data communication between the cores is required as the access and modification of the same memory space by one core is visible to all other cores. However, additional synchronization during write/read operations is needed in order to insure appropriate access to shared memory. Parallelization with shared memory approach is also applied on the master node for the distribution and gathering population from other nodes. In addition, those individuals that were transformed into leaves after the application of genetic operators are extended into stumps in parallel by cores at each slave node.

4 Experiments

In this section we show the performance of the proposed parallel version of the GMT system. Two sets of experiments were performed on real-life and artificial datasets using evolutionary induced regression and model trees.

4.1 Setup

All presented results were obtained with a default setting of parameters from the sequential version of the GMT system. We have tested one artificially generated dataset called *Armchair* [6] with 4 different number of instances (from 1 000 to 1 000 000) and 4 real-life datasets available in the UCI Machine Learning Repository [14]. In addition, we have compared the time performance between regression and model tree inductions, and provide some detailed time-sharing information of our MPI+OpenMP parallelization.

In the paper we focus only on the time performance of the GMT system, therefore, results for the prediction accuracy are not enclosed. However, for all tested datasets, the proposed hybrid approach achieved very good results - the same as the sequential version [6].

Experiments were performed on a cluster of sixteen SMP servers (nodes) running Ubuntu 12 and connected by an Infiniband network (20 Gb/s). Each server was equipped with 16 GB RAM, 2xXeon X5355 2.66 GHz CPUs with total number of cores equal 8. We used the Intel version 15.1 compiler, MVAPICH version 2.2 and OpenMP version 3.0. Within each node, the shared memory approach (OpenMP) was applied whereas between the nodes the message-passing interface (MPI) was used. For performance measuring we made use of the Multi-Processing Environment (MPE) library with the graphical visualization tool Jumpshot-4 [11].

4.2 Results

In the first experiment, the authors focus on the overall speedup of the proposed hybrid MPI+OpenMP approach. Table 1 presents the obtained mean speedup for different datasets. Only the best combination of nodes and cores is shown and it looked as follows for all tested datasets:

- results for 2 cores: 1 node with 2 OpenMP threads;
- results for 4 cores: 4 nodes with 1 OpenMP thread;
- results for 8, 16, 32, 64 cores: 8 nodes with 1, 2, 4, 8 OpenMP threads per node, respectively.

It should be recalled that the shared memory approach is strongly linked and limited by the available hardware (e.g. 8 cores in one node), whereas within the distributed memory approach it is usually easier to create more numerous configurations.

Results enclosed in Table 1 show that the proposed hybrid parallelization noticeable decreases the tree induction time on artificial and real-life datasets.

Table 1. Mean speedup reported for different number of cores

Dataset	Instances	Attributes	Speedup on different number of cores					
			2	4	8	16	32	64
Armchair	1 000	2	1.91	3.64	6.25	9.81	14.24	20.11
Armchair	10 000	2	1.85	3.38	6.19	9.98	14.99	22.81
Armchair	100 000	2	1.85	3.40	6.04	9.97	14.74	20.85
Armchair	1 000 000	2	1.72	3.31	5.62	9.08	13.52	17.52
Stock	950	9	1.47	3.21	3.92	8.08	11.86	19.49
Pol	15 000	48	1.74	3.23	5.47	9.06	14.95	18.33
Fried	40 768	10	1.80	2.81	5.09	8.54	13.63	23.08
Elnino	178 080	9	1.80	2.82	5.79	7.82	11.82	15.80

The speedup for 64 cores is in range from ×15.8 on the *Elnino* dataset to ×23.1 on the *Fried* dataset. With such speedup, the average tree induction time for GMT on the *Elnino* dataset (the biggest dataset from [6]) decreased from over 10 h to 40 min. The smaller speedup on the largest datasets (*Armchair* with 1 million instances and *Elnino*) may be caused by the necessity of reallocating large number of instances (after unpacking the message) in the affected node on the slaves. However, the algorithm speedup is still higher than for the evolutionary induced classification trees where the maximum speedup did not exceed ×15 [8] on artificially generated datasets. We can observe that the speedup differences between 32 and 64 cores are relatively small considering doubling the number of cores. The possible reason is the size of the population (default: 64 individuals). To achieve effective parallelization, the total number of cores should not exceed half of the population size because for some operations like crossover, each core performs calculations on two individuals.

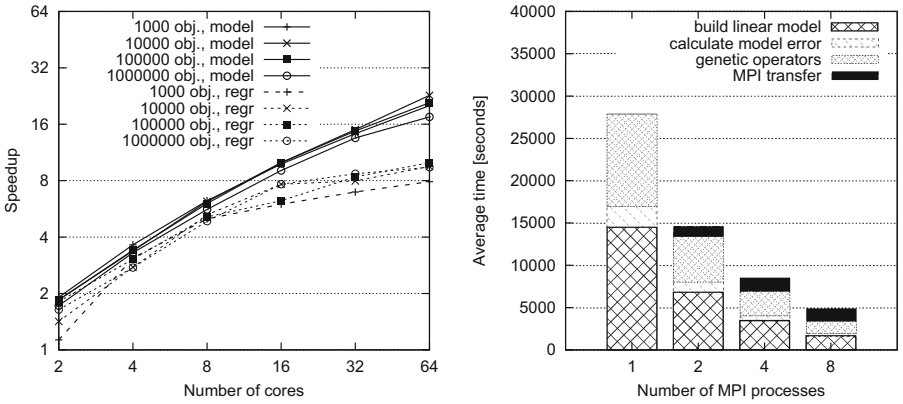

Fig. 3. Performance evaluation: (a) speedup comparison between model and regression (*regr*) tree induction on the *Armchair* dataset with different number of objects, (b) detailed time-sharing information of model tree induction (without OpenMP) with different number of slaves. Evaluation performed on the *Armchair* dataset with 100 000 instances.

Figure 3(a) illustrates the performance of the regression and model tree induction on the *Armchair* dataset with various number of instances. We can observe that with the increase of the number of cores the disproportion between the speedup for both types of tree representations is getting higher (around 2× smaller speedup for 64 cores in favor of model trees). To better understand why the proposed approach performs differently on regression and model trees see Fig. 3(b) which illustrates in details the time-sharing information for the induction of the model trees. In case of the model tree induction, more than 60 % of the evolutionary loop time is spent on building the linear models in the leaves

and calculating their errors. We can observe, that this part of the algorithm is well scalable (over 95 %), however, it only exists for the evolutionary induced model trees as the regression trees does not have linear models in the leaves. Other parallelized parts like genetic operators (which include embedded fitness recalculations) are also well scalable (around 95 %), however, due to the overhead (MPI transfer) the speedup improvement on larger number of cores is smaller. For 8 processes the MPI takes almost 30 % of the evolutionary loop time in case of the model trees and approximately 50 % for the regression trees. In addition, as some parts of the algorithm have to run sequentially, the efficiency for the higher number of cores is getting smaller, as expected from the Amdahl's law [10].

5 Conclusion and Future Works

The growing popularity of the evolutionary induced model trees can be withheld if there will be no sufficient solutions to improve their speed and their ability to analyze large scale data. In the paper, the authors propose a hybrid MPI+OpenMP parallelization to extend the GMT system. Proposed implementation takes an advantage of modern parallel machines and may provide an efficient acceleration on high-performance computing clusters as well as on low-cost commodity hardware. We see many promising directions for the future research. One of the possibilities is an additional parallelization of the models calculations in the leaves (with OpenMP) as well as to deal with a GPGPU parallelization.

Acknowledgments. This project was funded by the Polish National Science Center and allocated on the basis of decision 2013/09/N/ST6/04083 (first author) and grants W/WI/2/2014 (second author) and S/WI/2/2013 (third author) from Bialystok University of Technology.

References

1. Barros, R.C., Basgalupp, M.P., Carvalho, A.C., Freitas, A.A.: A survey of evolutionary algorithms for decision-tree induction. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**(3), 291–312 (2012)
2. Bull, L., Studley, M., Bagnall, A., Whittley, I.: Learning classifier system ensembles with rule-sharing. IEEE Trans. Evol. Comput. **11**, 496–502 (2007)
3. Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic, Norwell (2000)
4. Chapman, B., Jost, B.G., van der Pas, R., Kuck, D.J.: Using OpenMP: Portable Shared Memory Parallel Programming. MIT Press, Cambridge (2007)
5. Chitty, D.M.: Fast parallel genetic programming: multi-core CPU versus many-core GPU. Soft. Comput. **16**, 1795–1814 (2012)
6. Czajkowski, M., Kretowski, M.: Evolutionary induction of global model trees with specialized operators and memetic extensions. Inf. Sci. **288**, 153–173 (2014)
7. Czajkowski, M., Czerwonka, M., Kretowski, M.: Cost-sensitive global model trees applied to loan charge-off forecasting. Decis. Support Syst. **74**, 55–66 (2015)

8. Czajkowski, M., Jurczuk, K., Kretowski, M.: A parallel approach for evolutionary induced decision trees. MPI+OpenMP implementation. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2015. LNCS, vol. 9119, pp. 340–349. Springer, Heidelberg (2015)
9. Gagne, P., Dayton, C.M.: Best regression model using information criteria. J. Mod. Appl. Stat. Methods **1**, 479–488 (2002)
10. Grama, A., Karypis, G., Kumar, V., Gupta, A.: Introduction to Parallel Computing. Addison-Wesley, Boston (2003)
11. Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message-Passing Interface. The MIT Press, Cambridge (2014)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Data Mining, Inference, and Prediction, 2nd edn. Springer, New York (2009)
13. Kotsiantis, S.B.: Decision trees: a recent overview. Artif. Intell. Rev. **39**, 261–283 (2013)
14. Lichman, M.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (2013). http://archive.ics.uci.edu/ml
15. Llora, X.: Genetics-Based Machine Learning using Fine-grained Parallelism for Data Mining. Ph.D. Thesis. Barcelona, Ramon Llull University (2002)
16. Loh, W.: Fifty years of classification and regression trees. Int. Stat. Rev. **83**(3), 329–348 (2014)
17. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, New York (1996)
18. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. Cambridge University Press, Cambridge (1988)
19. Rokach, L., Maimon, O.Z.: Top-down induction of decision trees classifiers - a survey. IEEE Trans. SMC, Part C **35**(4), 476–487 (2005)
20. Rabenseifner, R., Hager, G., Jost, G.: Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In: Proceedings of PDP'17, pp. 427–436 (2009)
21. Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**, 461–464 (1978)