DATA MINING VIA DISTRIBUTED GENETIC PROGRAMMING AGENTS

Gabriel K. Kronberger[(a)], Stephan M. Winkler[(b)], Michael Affenzeller[(c)], Stefan Wagner[(d)]

[(a) (b) (c) (d)]Heuristic and Evolutionary Algorithms Laboratory
School of Informatics, Communications and Media
Upper Austria University of Applied Sciences
Softwarepark 11, 4232 Hagenberg, Austria

[(a)]gkronber@heuristiclab.com, [(b)]swinkler@heuristiclab.com, [(c)]maffenze@heuristiclab.com, [(c)]swagner@heuristiclab.com

ABSTRACT
Genetic programming is a powerful search method which can be applied to the typical data mining task of finding hidden relations in datasets. We describe the architecture of a distributed data mining system in which genetic programming agents create a large amount of structurally different models which are stored in a model database. A search engine for models that is connected to this database allows interactive exploration and analysis of models, and composition of simple models to hierarchical models. The search engine is the crucial component of the system in the sense that it supports knowledge discovery and paves the way for the goal of finding interesting hidden causal relations.

Keywords: distributed data mining, genetic programming, knowledge discovery

1. INTRODUCTION

1.1. Data Mining and Genetic Programming

Hand et al. give the following definition of data mining: *"Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner."* (Hand, Mannila and Smyth 2001). Data mining is just one step in the more comprehensive process of knowledge discovery. Other equally important steps include the preparation of data for the mining process and subsequent interpretation of generated models; cf. (Fayyad, Piatetsky-Shapiro and Smyth 1996). The goal of the whole process is to gain new knowledge about the observed system which can be utilized consequently to improve aspects of the system, for instance to gain a competitive advantage.

Genetic programming is an optimization technique that works by imitating aspects of natural evolution to generate a solution that maximizes or minimizes a fitness function. A population of solution candidates evolves through many generations towards a solution using three evolutionary operators: selection, recombination and mutation. Genetic programming is based on genetic algorithms the main difference is the representation of solution candidates, whereas genetic algorithms are intended to find an array of characters or integers representing the solution of a given problem, the goal of GP is to produce a computer program solving the problem at hand.

Genetic programming has been used successfully for data mining tasks using different forms of solution representations. One approach is symbolic regression to build formulas that describe the behavior of systems from measured data, see for example (Koza 1992; Keijzer and Babovic 1999; Langdon and Poli 2002; del Re, Langthaler, Furtmüller, Winkler and Affenzeller 2005; Winkler, Affenzeller and Wagner 2007a; Winkler, Affenzeller and Wagner 2007b). Other approaches use GP to discover predictive IF-THEN-rules typically with prior discretization of variables (Freitas 1999; Wong and Leung 2000) or to evolve decision trees (Fu 1999; Ryan and Rayward-Smith 1998; Papagelis and Kalles 2001).

In this paper we describe a system to support and improve the knowledge discovery process based on distributed data mining agents which concurrently run genetic programming processes. In section 2 we describe the main problems of GP-based data mining and describe potential benefits of a distributed data mining system. In section 3 we describe the architecture of this system and its major components while sections 4 and 5 conclude this paper with ideas to further improve the knowledge discovery process by integrating a priori knowledge and user feedback.

2. MOTIVATION

A frustrating aspect of GP is that it takes a long time until the result of a run is available. Especially for non-trivial datasets it is usually necessary to analyze the result of a previous run before a new run can be started for instance to counteract over or under fitting or to exclude dominant input variables. This is an even bigger problem for domain experts who do not fully understand the internal details of GP and thus often have problems to configure the algorithm correctly. Usually a few iterations are necessary until a configuration for the algorithm is found that works for data mining task at hand. However, even when such a

configuration has been found there is another aspect that causes friction in the knowledge discovery process.

Multiple genetic programming runs with the same settings and the same input data typically result in a diverse set of structurally different models with similar predictive accuracy. This behavior is caused by the fact that GP is a stochastic method which searches for a model that fits the target variable as well as possible. GP specifically doesn't search for the simplest or most compact model, on the contrary internal dynamics of the evolutionary process cause the models to become overly complex. This effect is known in the GP community as "bloat"; cf. (Langdon and Poli 2002). Various strategies to combat bloat have been discussed in GP literature, one recent addition is (Dignum and Poli 2008), giving a full overview would go beyond the scope of this paper. For the task at hand the effect can be alleviated through simplification of the resulting models. However the basic problem remains because there are often implicit dependencies between input variables and there is usually an infinite number of ways to express the same function.

The result is that it is difficult to extract knowledge out of the generated models because the relevant information about the underlying structure in the data is blurred by the large amount of possible mathematical representations of that structure. The knowledge gained from these experiments is often limited to the insight into which variables play an important role in models for the target variable. While this insight can be valuable in itself it can also be reached with statistical methods with a lot less effort. One important feature of GP that distinguishes it from many other optimization methods is that it is able to automatically optimize the model structure while at the same time optimizing the model parameters. This characteristic cannot be utilized to its full extent when the analyzed models are all structurally different and thus difficult to analyze which thwarts the effort spent to build the model structure.

One approach to improve the discovery of more detailed knowledge is to run many independent GP processes to generate a large number of models for each possible target variable with different complexities and to extend the data mining process to search for implicit dependencies between input variables. In combination with an interactive user-interface to filter and analyze the generated models and to compose simple models to hierarchic models the user gains new knowledge step by step while investigating the set of models.

Evolutionary algorithms especially genetic programming are often slower than other more specialized data mining algorithms while reaching comparable predictive accuracy. However it's easy to parallelize evolutionary algorithms. In the proposed data mining system this is even simpler since the independent genetic programming processes can be executed concurrently. Depending on the complexity and extent of the dataset it can still take hours or even a few days to generate enough models to start interpreting the generated models, however as Freitas states: *"Data mining is typically an off-line task, and in general the time spent with running a data mining algorithm is a small fraction (less than 20%) of the total time spent with the entire knowledge discovery process. [...] Hence, in many applications, even if a data mining algorithm is run for several hours or several days, this can be considered an acceptable processing time, at least in the sense that it is not the bottleneck of the knowledge discovery process."* (Freitas 2002). A benefit of the system is that the user can already start to analyze preliminary results while the background GP processes are still refining models.

The design goals of the system can be summarized in the following four points:

- Find all potentially interesting (non-linear) relations of variables of a dataset.
- Store models of different complexity and accuracy.
- Provide functionality to explore, analyze and compose such models.
- Record all steps in the mining process which led to a given result.

These goals are closely related to the goals given by Blumenstock, Schweiggert and Müller 2007 in that they result out of similar considerations regarding the focus and breadth of the search process and the transparency and ease of use of the system.

In the following section we describe the architecture of the proposed system and its components.

3. DISTRIBUTED DATA MINING ARCHITECTURE

The system is made up of three parts: a central model database, a mesh of distributed genetic programming agents and a component to import new datasets into the system and to navigate, explore and search models stored in the central database.

3.1. Layout of the Generic Model Database

Figure 1 shows an entity-relationship diagram of the generic data model of the model database. The two most important entities in the data model are the dataset and the model. It is often the case that a dataset once imported is preprocessed for instance to scale all variable values to a predefined range. For sake of transparency each dataset is linked back to its source. Later all processing steps can be retraced with this relation. Additionally the person who imported or manipulated the dataset is linked to each dataset. Each model is linked to the process that generated it to make the origin of that model transparent. Each process is also linked back to a person who is the controller of that process. By adding the algorithm that each process executes in the data model it is possible to repeat each experiment at a later point in time.

The layout of the data model is kept very generic on purpose to make it easy to add new data mining algorithms with different model representations to the

system. Algorithms could be different genetic programming variants or even other (non-evolutionary) data mining algorithms like C4.5, kNN, CART or SVM.

The main design goal of the model database is transparency. It must be possible to reproduce every single result that is stored in the database. For this reason the algorithm implementation and the parameter settings are also stored in the database.

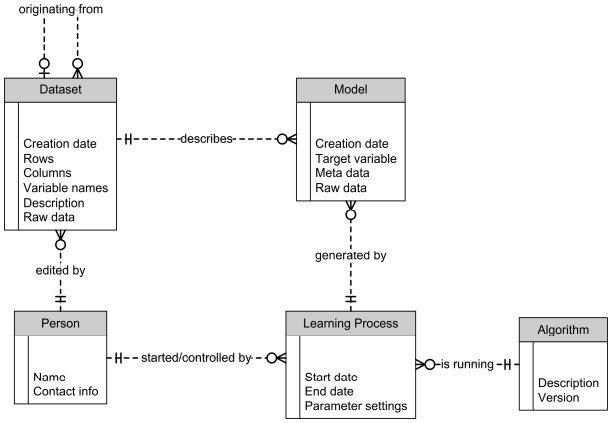

Figure 1: Entity-relationship diagram showing the generic data model of the model database.

3.2. Distributed Genetic Programming Agents

Figure 2 shows the typical cycle of data mining with genetic programming. The user supplies the dataset and configures the parameters of the algorithm. The most important parameters are the set of allowed input variables the target variable and the set of functions that should be used to compose the models. After a few hours the result of the algorithm in the form of a formula is available. Through the analysis of this result the user gains new knowledge and starts a new GP run with different settings (for instance removing an input variable).

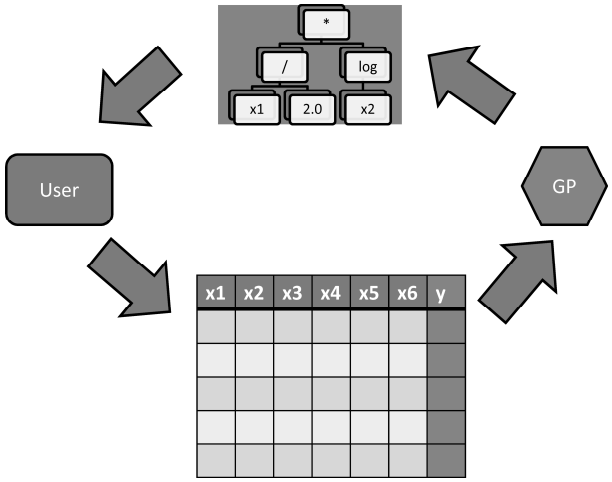

Figure 2: The usual way of GP-based data mining has a cycle with a long feedback loop.

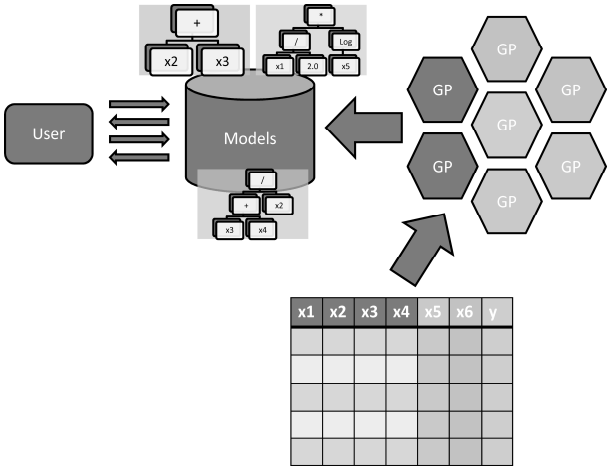

Figure 3: In the proposed system distributed GP agents continuously analyze the dataset and store new models in the database. The user interactively explores and analyzes available models.

Figure 3 shows how this process could be improved by using parallelism to run different GP processes at the same time. Each of the distributed genetic programming agents has different settings for the target variable, maximally allowed model complexity and the set of allowed input variables. Controller agents create new GP jobs and coordinate the running GP agents.

It would be interesting to have more intelligent controller agents which try to predict which models are more interesting for the user and guide the GP agents to search especially for such models; this remains a topic for future research.

3.3. Interactive Model Exploration and Analysis

The mesh of distributed genetic programming agents generates a flood of potentially interesting models. However only a few of these models will actually be interesting for the user and it is usually impossible to automatically recognize the relevant models and throw out the rest. So an user-interface for the interactive exploration and analysis of all available models is essential. It is the most crucial component for the knowledge discovery process because the facts the user is searching for are likely hidden and can only be uncovered when it is possible to arbitrarily filter and sort the available models and drill down to uncover alternative representations of a model.

The quality of any model can be inspected visually through line charts of the estimated and the original value of the target variable and through scatter plots showing the correlations of estimated vs. original values. The relative complexity of models can be visualized through different colors.

We plan to implement a kind of search engine for models which allows filtering and sorting available models for a given dataset by at least the following attributes:

- Target variable
- Input variables
- Accuracy
- Complexity

Search queries can be freely combined and negated for instance to create the search request *"Find models for variable X which do not contain variable Y and Z sorted by accuracy"* or *"Find models for variable Z using variable A and B with a maximal tree-depth of four"*. Additionally to the basic filtering and sorting functionality it is interesting to explicitly search for similar models to a given model. This can be implemented by searching for models with the same target variable and the same input variables. Searching for structural similarity could also be useful but remains an open topic.

To make the search process transparent to the user it will be possible to display which algorithm in combination with which settings produced a given model. Going a step further it's also interesting to show the internal state of the algorithm when it produced the given model. This information is especially helpful to refine and improve the distributed data mining system itself.

4. FUTURE WORK

The first step towards an interactive data mining environment for practical application is the implementation and roll-out of the system described in the previous sections. We plan to combine the distributed data mining system with HeuristicLab (available at http://www.heuristiclab.com) (Wagner, Affenzeller 2005), a modern framework for prototyping and analyzing optimization techniques for which both generic concepts of evolutionary algorithms and many functions to evaluate and analyze algorithms are available. HeuristicLab makes it very easy to create customized algorithms from predefined components. It is a close to ideal environment for the data mining system because as Hand et al. stated: *"When faced with a data mining application, a data miner should think about which* components *fit the specifics of his or her problem, rather than which specific "off-the-shelf" algorithm to choose. In an ideal world, the data miners would have available a software environment within which they could* compose *components (from a library of model structures, score functions, search methods, etc.) to synthesize an algorithm customized for their specific applications. Unfortunately this remains a ideal state of affairs rather than the practical norm; current data analysis software packages often provide only a list of algorithms, rather than a component-based toolbox for algorithm synthesis."* (Hand, Mannila and Smyth 2001).

Once the basic infrastructure is implemented and running there are a few more possible research directions additionally to the open topics mentioned above.

One interesting aspect is to enhance the cooperation between the distributed GP agents by reusing models from the model database generated by other agents. These models could be integrated into other models as virtual variables replacing the actual training data with predictive models for those variables. This scheme enhances reuse of learned concepts and knowledge and enables the creation and composition of successively higher-level models. This idea is related to automatically defined functions (Koza 1992) and similar to adaptive representation through learning (Rosca and Ballard 1996).

A possible direction for future research is extending the user-interface to automatically learn and recognize which models are interesting for the user. One possible approach is to track user search actions and using a form of online data mining to analyze and predict the interestingness of a given model based on previous user actions.

5. SUMMARY

In this paper we describe the architecture of a distributed data mining system utilizing genetic programming agents to compose a diverse set of predictive models for a given dataset. The system is made up of three parts: a central model database, a mesh of distributed genetic programming agents and a component to import new datasets into the system and to navigate, explore and search models stored in the central database.

In the proposed system distributed genetic programming agents continuously process available datasets and create models of different complexity for different target variables and for different sets of input variables. These agents store potentially interesting models in a database together with meta-data to filter and sort models. Because of the the high volume and diversity of the model database an interactive user-interface to navigate, filter and analyze the models is necessary. This interface is the crucial component in the knowledge discovery process in the sense that it allows interactive exploration of results and paves the way for the goal of finding interesting hidden causal relations.

ACKNOWLEDGMENTS

The research presented in this paper is funded by the Upper Austrian University of Applied Sciences in the scope of the project "Cooperative Evolutionary Data Mining Agents".

REFERENCES

Blumenstock, A., Schweiggert F., Muller M., 2007. Rule cubes for causal investigations. *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, pp. 53 – 62. October 28-31 2007, Omaha, Nebraska, USA.

Del Re, L., Langthaler, P., Furtmüller, C., Winkler, S., Affenzeller, M., 2005. NOx Virtual Sensor Based on Structure Identification and Global Optimization. *Proceedings of the SAE World*

Congress 2005, paper number 2005-01-0050. April 11-14 2005, Detroit, MI, USA.

Dignum, S., Poli, R., 2008. Operator equalisation and bloat free GP. *Genetic Programming, 11th European Conference, EuroGP 2008*, pp. 110 – 121. March 26-28 2008, Naples, Italy.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., 1996. From data mining to knowledge discovery: an overview. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds. *Advances in knowledge discovery and data mining*: pp. 1 – 34. Menlo Park, CA, USA:AAAI

Freitas, A. A., 1999. A genetic algorithm for generalized rule induction. In Roy, R., Furuhashi, T., Chawdhry, P. K., eds. *Advances in Soft Computing: Engineering Design and Manufacturing*. Berlin:Springer-Verlag, pp. 340 – 353.

Freitas, A. A., 2002. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Fu, Z., 1999. An innovative GA-based decision tree classifier in large scale data mining. *Principles of Data Mining and Knowledge Discovery*, pp. 348 – 353.

Hand, D. J., Mannila, H., Smyth, P., 2001. *Principles of Data Mining*. The MIT Press

Keijzer, M., Babovic, V., 1999. Dimensionally aware genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference 1990*, pp. 1069 – 1076. July 13-17, 2005, Orlando, FL, USA.

Koza, J., 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: The MIT Press.

Langdon, W. B., Poli, R., 2002. *Foundations of Genetic Programming*. Berlin, Germany:Springer-Verlag.

Papagelis, A., Kalles, D., 2001. Breeding decision trees using evolutionary techniques. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*, pp. 393 – 400. June 28 – July 1 2001, Williamstown, MA, USA.

Rosca, J. P., Ballard, D. H., 1996. Discovery of subroutines in genetic programming. In Angeline, P. J., Kinnear, K. E., eds. *Advances in Genetic Programming 2*, pp. 177 – 202. Cambridge, MA, USA: The MIT Press.

Ryan, M. D., Rayward-Smith, V. J., 1998. The evolution of decision trees. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., Riolo, R., eds. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 350 – 358, University of Wisconsin, Wisconsin, USA:Morgan Kaufmann.

Wagner, S., Affenzeller, M., 2005. HeuristicLab: A generic and extensible optimization environment. *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms*, pp. 538 – 541. March 21-23 2005, Coimbra, Portugal.

Winkler, S., Affenzeller, M., Wagner, S., 2007a. Advanced Genetic Programming Based Machine Learning. *Journal of Mathematical Modelling and Algorithms*, Vol. 6: pp. 455 – 480.

Winkler, S., Affenzeller, M., Wagner, S., 2007b. Selection Pressure Driven Sliding Window Genetic Programming. *Computer Aided Systems Theory – EUROCAST 2007, Lecture Notes in Computer Science*, Vol. 4739: pp. 788 – 795.

Wong, M. L., Leung, K. S. 2000. *Data Mining Using Grammar Based Genetic Programming and Applications*, Norwell, MA, USA: Kluwer Academic Publishers

AUTHORS BIOGRAPHY

GABRIEL K. KRONBERGER received his MSc. in computer science in 2005 from Johannes Kepler University Linz, Austria. His research interests include parallel evolutionary algorithms, genetic programming, machine learning and data mining. Currently he is a research associate at the Research Center Hagenberg of the Upper Austrian University of Applied Sciences.

STEPHAN M. WINKLER received his MSc in computer science in 2004 and his PhD in engineering sciences in 2008, both from JKU Linz, Austria. His research interests include genetic programming, nonlinear model identification and machine learning. Currently he is research associate at the Research Center Hagenberg of the Upper Austrian University of Applied Sciences, working on the research program L284-N04 "GP-Based Techniques for the Design of Virtual Sensors", a research project funded by the Austrian Science Fund (FWF).

MICHAEL AFFENZELLER has published several papers and journal articles dealing with theoretical aspects of evolutionary computation and genetic algorithms. In 1997 he received his MSc in mathematics and in 2001 his PhD in engineering sciences, both from JKU Linz, Austria. He is professor at the Upper Austria University of Applied Sciences (Campus Hagenberg) and associate professor at the Institute of Formal Models and Verification at JKU Linz since his habilitation in 2004.

STEFAN WAGNER also received his MSc in computer science in 2004 from Johannes Kepler University Linz, Austria. He currently holds the position of an associate professor at the Upper Austrian University of Applied Sciences (Campus Hagenberg). His research interests include evolutionary computation and heuristic optimization, theory and application of genetic algorithms, machine learning and software development.