Using machine learning techniques to interpret results from discrete event simulation

Dunja Mladenić[1], Ivan Bratko[1,2], Ray J. Paul[3], Marko Grobelnik[1]

[1]J. Stefan Institute, Dept. of Computer Sc., Jamova 39, Ljubljana, Slovenia

[2]Faculty of Electr. Eng. and Computer Sc., Tržaška 25, Ljubljana University

[3]Brunel University, Dept. of Computer Sc., Uxbridge, England

e-mail: dunja.mladenic@ijs.si

Abstract

This paper describes an approach to the interpretation of discrete event simulation results using machine learning techniques. The results of two simulators were processed as machine learning problems. In the experiments, the simulation results were interpreted by two attribute-based learning programs RETIS and ASSISTANT, and by an Inductive Logic Programming system MARKUS. Some interesting regularities were thereby automatically discovered.

1 Introduction

Discrete simulation involves several stages: design and implementation of a model for the system under study, performing experiments with the simulator, analysis and presentation of experimental results. Analysis and presentation of discrete event simulation results is usually done intuitively. Deriving useful conclusions from the simulation results is a demanding task. In this paper we propose to automate this task by means of machine learning techniques.

It should be emphasised that in this application of Machine Learning tools the prediction accuracy is not the main goal. After all, if the user is interested in prediction, she or he can always use the simulator for that. The real goal here is to find interpretation of simulation data and discover regularities, thus helping the user to develop intuitive understanding of the domain. Discrete event simulation produces example situations that can be used as examples for machine learning. In the presented research, different machine learning systems were used in searching for a simple and intuitive interpretation of simulation results. First, attribute-value learning system RETIS (Karalič 1991, Karalič 1992) was chosen because it can handle continuous attributes and class. Interpretation obtained by this system was intuitive but obviously expressed in a complicated way. To enable a more powerful knowledge representation, Inductive Logic Programming (ILP) system MARKUS (Grobelnik 1992) was used. For this purpose, the problems were reformulated to have Boolean, instead of continuous class. The same reformulation was used in experiments with the attribute-value learning algorithm ASSISTANT (Cestnik, Kononenko and Bratko 1987). Additional to compact knowledge representation, MARKUS also highlighted some attribute combinations that could be useful in attribute-value learning. These attribute combinations were used as new attributes in further experiments with RETIS and ASSISTANT, which are more appropriate for noisy domains than MARKUS. In fact, MARKUS's inability to handle noise proved to be very limiting in this application.

Section 2 describes discrete event simulation domains used in our experiments with machine learning. Section 3 describes three machine learning algorithms used to interpret the results of discrete event simulation. Results are discussed in Section 4.

2 Domains description

In our experiments we used two discrete event simulators, the supermarket and the pub. They were both generated by *The Extended Lancaster Simulation Environment eLSE* (Paul and Balmer 1991) and used with the run-in time parameter set to 50 minutes and simulation duration set to 1000 minutes. The interaction of objects for each example simulator is described by *Activity Cycle Diagrams* (Fig. 1). These diagrams use two types of objects: circles to denote queues and rectangles to denote activities.

Each discrete event simulator has parameters that can be changed by the user (e.g. the number of cashiers in the supermarket simulation) and some statistical indicators calculated by the simulator (e.g. barmaid utilisation in the pub simulation). The user defined parameters can be viewed as input to the simulator (attributes) while the statistical parameters are the outputs (classes). Usually, the result of machine learning describes the relation between a single *class* and the *attributes*. Therefore, for each simulator and for each output value, a machine learning domain was constructed and experiments with it performed.

2.1 The supermarket domain

In the supermarket domain (Fig. 1, left hand graph), customers arrive and spend some time shopping. After shopping, they have to wait to be served by a cashier. A customer who on arrival sees a queue of more than `QWSize` people waiting to be served, leaves without entering the supermarket.

In the described process, there are some attributes whose influence we want to investigate:

- number of cashiers in the supermarket, `Cashiers` (varied from 1 to 5)
- maximum queue size tolerated by a customer, `QWSize` (varied from 1 to 10)
- time between two arrivals, `Arrival` (exponentially distributed with a mean of 1 to 10 minutes)

We considered as interesting the following statistical outputs, used as classes in learning:

- customer "utilisation" (ratio between the number of customers that enter the shop and the number of customers that arrive)
- cashier utilisation (fraction of cashier's time spent actually serving customers)
- length of the customers wait queue

From the above parameters, three learning problems can be formulated, each concerned with the relation between the attributes and one of the statistical outputs (e.g. influence of the number of cashiers, customers tolerance and arrival rate on customer utilisation). Each combination of attribute values corresponds to a run of simulator producing a machine learning example. In this way, domains for all possible attribute values were generated, having $5 \times 10 \times 5 = 500$ examples.

2.2 Pub simulator domain

In the pub simulator (Fig. 1, right hand graph), customers arrive and wait for the barmaid to pour them a glass of drink. After drinking, they leave or return to the queue for another drink. The barmaids just pour beer and wash glasses. A glass can be either clean (and empty), full or dirty.

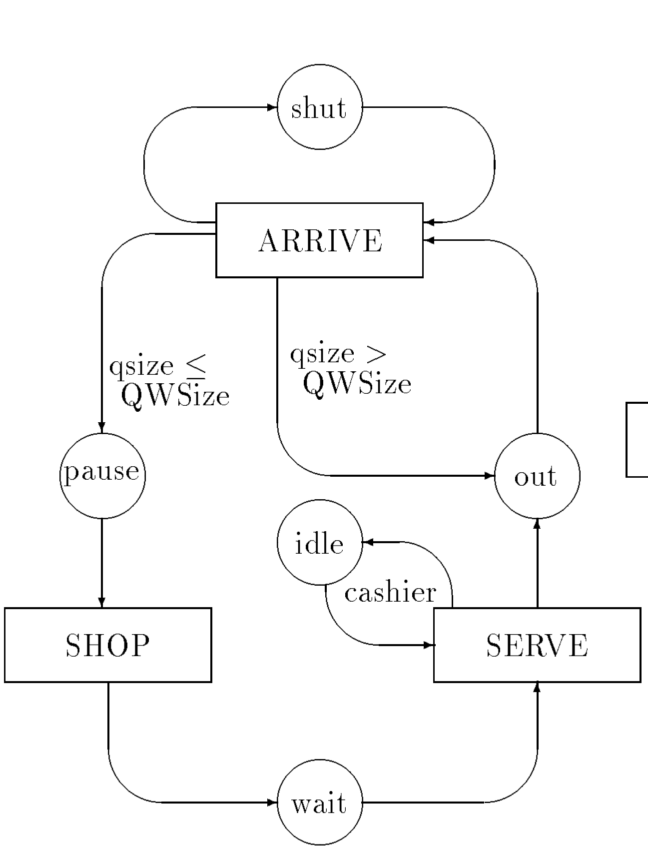

Figure 1: Activity Cycle Diagram for the supermarket (left) and the pub (right) simulation

In the described process, there are the following attributes whose influence on simulation we want to investigate:

- number of barmaids in the pub, `Barmaids` (varied from 1 to 5)
- number of glasses in the pub, `Glasses` (varied: 50, 100, 150)
- time between two arrivals, `Arrival` (exponentially distributed with a mean of 5 to 17 minutes)

- drinking time, `Drink` (uniformly distributed between X and $X+3$ minutes; X is varied from 3 to 7)

We considered as interesting the following class values calculated by the simulator:

- barmaid utilisation
- length of the customers wait queue
- length of the clean glasses wait queue

From the above parameters and values, three learning problems can be formulated, each investigating the relation between the above attributes and one of the three class values (e.g. influence of number of barmaids, number of glasses, arrival rate and drinking time on barmaid utilisation). Combining all possible attribute values, domains were generated having $5 \times 3 \times 13 \times 5 = 975$ examples.

3 Machine learning algorithms used

This section describes three machine learning algorithms used to interpret the results of discrete event simulation. The first two algorithms are attribute-value learning algorithms ASSISTANT (Cestnik, Kononenko and Bratko 1987) and RETIS (Karalič 1991, Karalič 1992) while the third one is an inductive logic programming system MARKUS (Grobelnik 1992).

ASSISTANT is an attribute-value learning algorithm for the construction of binary classification trees. It belongs to the Top Down Induction of Decision Trees (Quinlan 1986) family of algorithms. RETIS is an attribute-value learning algorithm for the construction of regression trees. Regression trees are similar to binary classification trees constructed by the ASSISTANT algorithm. The most important difference is that, while classification trees are used to classify objects into *discrete* classes, regression trees are used when the class is *continuous*. A regression tree actually implements a function $y(x_1, x_2, \ldots, x_n)$ of n continuous or discrete attributes $x_1, x_2, \ldots, x_n$. Both algorithms incorporate several noise handling mechanisms that are used in our experiments.

MARKUS is an Inductive Logic Programming (ILP) system designed for inferring Prolog programs from examples and counterexamples of there behaviour. The MARKUS algorithm has its basis in Shapiro's MODEL INFERENCE SYSTEM (Shapiro 1983), but extends it in several ways. A Prolog program induced by MARKUS uses background-knowledge predicates defined by the user. Used predicates can be very simple such as the "greater-than" (>) predicate or complex procedures written in Prolog.

4 Experiments and results

Experiments were performed on six learning domains (three for the shop and three for the pub simulator). Used attribute-value learning systems, MAGNUS ASSISTANT (Mladenić 1990) and RETIS (Karalič 1991, Karalič 1992) perform *tree pruning* as noise handling mechanism. They were therefore executed with various settings of the parameters for pruning and some smaller trees were chosen for the analysis.

Four resulting regression and one classification tree are analysed in the following Sections (4.1, 4.2), two regression trees for each simulator.

4.1 Supermarket simulator

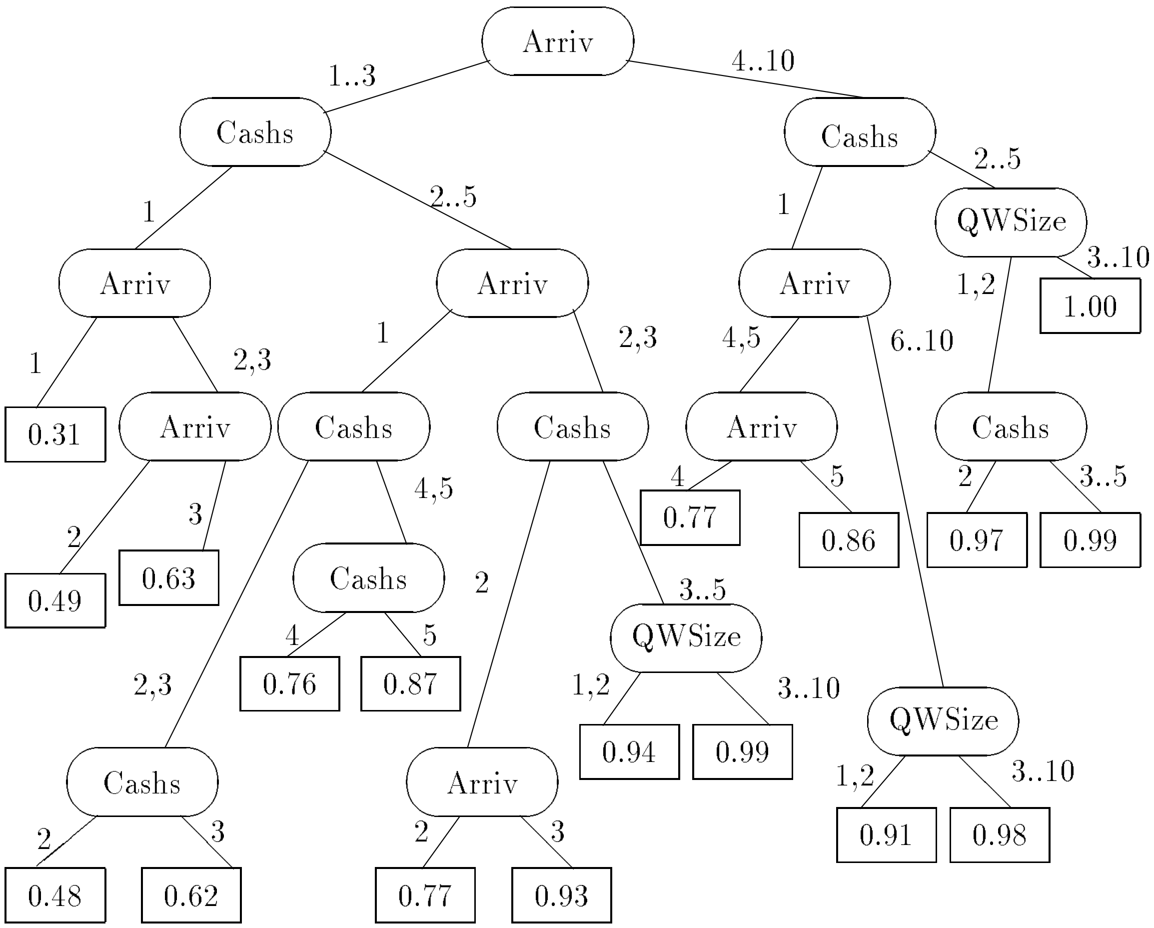

Figure 2: Regression tree for customer utilisation generated by Retis (postpruning with m=2).

As an example of regression trees generated by RETIS, let us consider the tree for customer utilisation (Fig. 2). The three left-most leaves correspond to `Cashiers` = 1. Customer utilisation here monotonically increases between 0.31 and 0.63. Customer utilisation is under these conditions independent of customer tolerance (`QWSize`). This can easily be explained as follows: for a small number of cashiers and a high arrival rate of customers (attribute `Arrival` small), the waiting queue in the steady state of the system is always long and occasionally drops below `QWSize`. The actual absolute length of the queue therefore doesn't matter and customer utilisation does not depend on `QWSize`. It can be clearly seen in Fig. 2 that customer utilisation does not depend on `QWSize` in the whole left-hand side of the tree. `QWSize` begins to affect the results for the combinations `Arrival` ≥ 6 or `Arrival` ≥ 2 and `Cashiers` ≥ 3. At lower arrival rates and/or higher number of cashiers, `QWSize` becomes important.

We analysed the simulation results with MARKUS with respect to the question: what combinations of the supermarket attributes will result in customer utilisation below or equal to 90%. For this experiment, we used the available 500 simulation results as follows. To

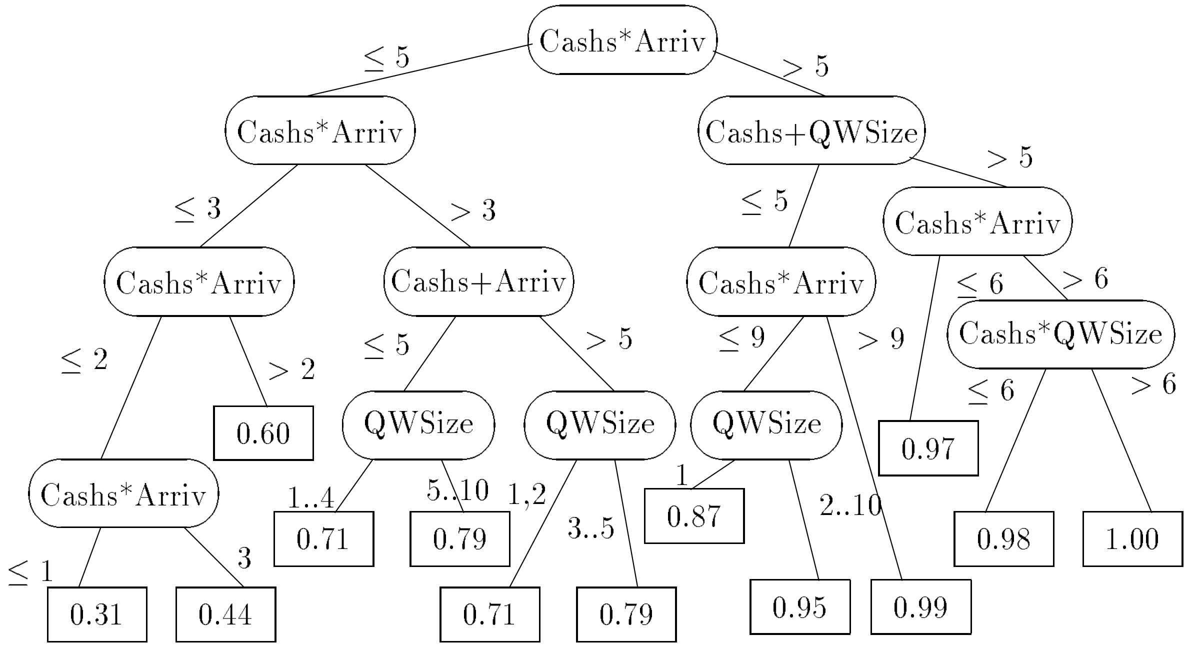

Figure 3: Regression tree for customer utilisation (added attributes) generated by Retis (postpruning with m=2).

avoid problems with small statistical fluctuations, we disregarded the results that were close (within ±5%) of the target threshold 90%. Accordingly, we considered a simulation result as an example of low performance if `CustomerUtil` ≤ 85% and a counter example if `CustomerUtil` > 95%. This resulted in 90 positive and 377 negative examples. We defined the following predicates as background knowledge for MARKUS:

```
mult( A, B, C ) :- C is A * B.
add( A, B, C ) :- C is A + B.
gtr1( X ) :- X > 1.
gtr2( X ) :- X > 2.
⋮
gtr9( X ) :- X > 9.
```

The target predicate for learning was: `shop( Cashs, QWSize, Arriv)`, which is true for customer utilisation ≤ 90%. The following two Prolog clauses were induced by MARKUS:

shop(Cashs, QWSize, Arriv) :- mult(Arriv, Cashs, X), not gtr5(X).
shop(Cashs, QWSize, Arriv) :- mult(Arriv, Cashs, X), not gtr1(QWSize), not gtr6(X).

These clauses can be read as:

if `Cashs*Arriv` ≤ 5 or `Cashs*Arriv` ≤ 6 and `QWSize` = 1
then `CustomerUtil` ≤ 90%.

MARKUS nicely discovered the law that holds in situations with high input of customers and low number of cashiers. In the extreme case the flow of customers can be viewed as a continuous liquid flow with the waiting queue never empty. It can be shown that under these conditions, customer utilisation is proportional to the product `Cashiers*Arrival`. At lower input rates or higher number of cashiers, the parameter `QWSize` begins to affect the performance which can also be seen in the rule above.

After seeing MARKUS results that highlighted multiplication of attributes, we constructed new attributes (in similar way as the ILP system LINUS (Lavrač, Džeroski and Grobelnik 1990) does) with four basic algebraic operations ($*$, $/$, $+$, $-$) applied on pairs of original attributes. Experiments with RETIS were repeated on such data sets. Additional experiments on the problems reformulated to have Boolean, instead of continuous class, were performed by the ASSISTANT algorithm. Utilisation problems were reformulated into learning when utilisation is below or equal to 90%. Mean length of customers wait queue was reformulated into problem of queue length beeing below or equal to 1 and mean length of clean glasses wait queue into problem of queue length beeing below or equal to 25. Both learning systems confirmed the results obtained by MARKUS, moreover in all three domains for the supermarket simulator the same combined attribute (`Cashiers*Arrival`) is recognised as the "best" one.

Adding attributes in the data, results in smaller regression tree (tree in Fig. 3 is smaller than the tree in Fig. 2), under the same pruning conditions. It can be clearly seen in Fig. 3 that the left-most part of the tree (`Cashiers*Arrival` ≤ 3) is similar to the left-most part of the tree in Fig. 2 (generated without added attributes). Customer utilisation here monotonically increases between 0.31 and 0.60 and is independent of the customers tolerance (`QWSize`). Classification tree generated by the ASSISTANT algorithm on the data with added attributes, has the same attribute (`Cashiers*Arrival`) in the root as the regression tree in Fig. 3. It is also interesting to compare Prolog clauses induced by MARKUS with trees constructed by RETIS and ASSISTANT. The same attribute with the same threshold 5 (`Cashiers*Arrival` ≤ 5) appears as the "best" one. The role of parameter `QWSize` is also very similar. It begins to effect the performance when `Cashiers*Arrival` is high (> 3 for RETIS, > 5 for MARKUS and ASSISTANT).

4.2 Pub simulator

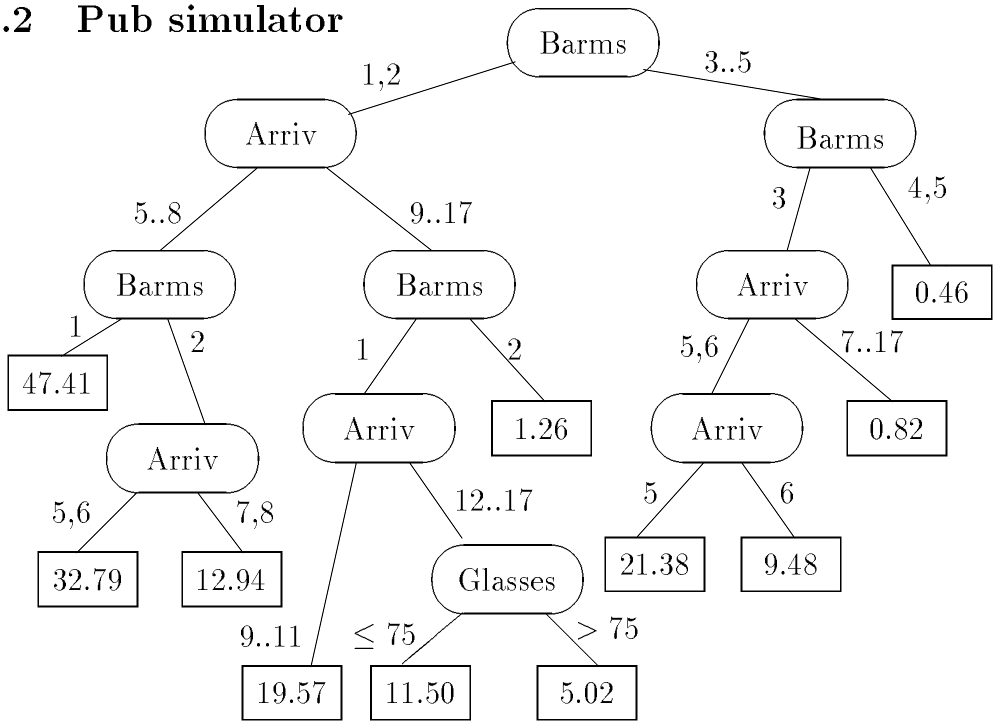

Figure 4: Regression tree for length of the customer wait queue generated by Retis (prepruning with m=2).

A regression tree was generated by RETIS for the customers wait queue length in the pub, as a function of the pub attributes: number of barmaids, number of glasses, customer arrival rate and drinking time. In that tree (Fig. 4), it is interesting to note that the glasses become an important resource in the case of one barmaid and low arrival rate (`Arrival` between 12 and 17). On average, an increase of glasses above 75 reduces the wait length from 11.50 to 5.02. Customers wait queue length drops below 1 only in the case of `Barmaids` > 4 or three barmaids and low arrival rate (`Arrival` between 7 and 17).

Trees in Figures 5 and 6 are generated for the customers wait length in the pub, as a function of four pub attributes (as above) and additional attributes constructed from the four original attributes with two algebraic operations ($*$, $+$). Like in regression tree generated without additional attributes, glasses again become an important resource (Fig. 5) in the case of low arrival rate (`Arrival` between 10 and 15). It is interesting that here glasses are important also in the right-most leaves of the tree (`Barmaids*Arrival` > 15). In these leaves customers wait length ≤ 1 for the number of glasses > 75 or `Barmaids*Arrival` > 19. Classification tree generated by ASSISTANT (Fig. 6) pointed out similar regularity: customers wait length ≤ 1 for the combinations `Barmaids*Arrival` > 25 or `Barmaids*Arrival` > 20 and `Glasses*Arrival` > 500. It is interesting that in ASSISTANT tree glasses are important for $20 <$`Barmaids*Arrival` ≤ 25. In other situations answer to the question about customer queue length ≤ 1 is very clearly determined by the arrival rate and number of barmaids.

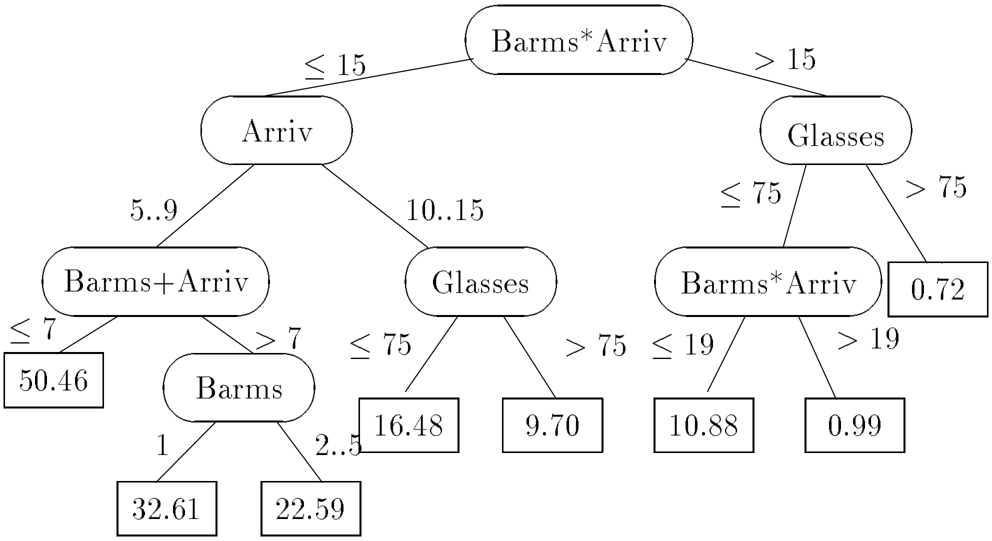

Figure 5: Regression tree for length of the customer wait queue (added attributes) generated by Retis (prepruning with m=2).

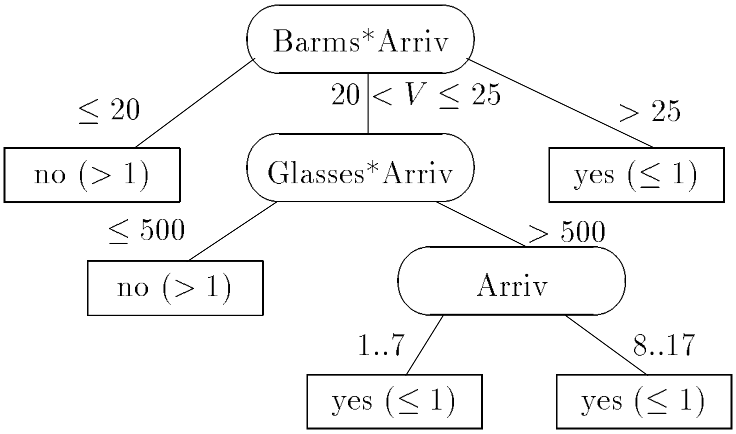

Figure 6: Classification tree for length of the customer wait queue (added attributes) generated by Assistant.

5 Conclusion

In the paper, we experimentally analysed the results of two discrete event simulators by means of machine learning programs. Two attribute value and one Inductive Logic Programming systems were used on six domains (three for each simulator). The obtained results are encouraging in that the induced descriptions provide some useful insights into the dependencies between the parameters of a discrete event system and its performance. Descriptions obtained by the ILP approach can be particularly concise and interesting.

The induced descriptions have been confirmed to be of interest for the simulation domain experts. This is important because the success of the present study should not be measured in terms of prediction accuracy, but in terms of interpretation value of the induced descriptions.

A methodological point of interest is the combination of ILP learning and attribute-based learning. The used ILP system MARKUS, could not really cope well with the noise in the learning data, but made important contribution as a generator of new combined attributes. These were then beneficially applied in the subsequent attribute-based learning.

6 Acknowledgements

We thank Aram Karalič for providing a copy of the RETIS program used in the experiments in this paper. This work was partially supported by the Slovenian Ministry for Science and Technology and the European ESPRIT III Basic Research Action Project No.6020 Inductive Logic Programming.

References

[1] Cestnik, B., Kononenko, I. and Bratko, I., (1987), ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users, in *Progress in Machine Learning*, ed. by I.Bratko and N.Lavrač, Sigma Press, Wilmslow.

[2] Grobelnik, M. (1992) Markus - an optimized Model Inference System, *ECAI-92 Workshop on Logical Approaches to Machine Learning*, Vienna, Avstrija.

[3] Karalič, A., Cestnik, B. (1991), The Bayesian Approach to Tree-Structured Regression. *Proceedings of ITI-91*, Cavtat, Croatia.

[4] Karalič, A. (1992) Employing Linear Regression in Regression Tree Leaves.*Proc. of ECAI 92*, pp. 440-441. Vienna, Avstrija, John Wiley & Sons.

[5] Lavrač, N., Džeroski, S. and Grobelnik, M. (1991) Learning nonrecursive definitions of relations with LINUS. *Proc. of EWSL 91*, pp. 265-281. Porto, Portugal, Springer-Verlag.

[6] Mladenić, D. (1990) Machine learning system Magnus Assistant (in Slovene). *BSc Thesis*, University of Ljubljana, Faculty of Electrical Engineering and Computer Science.

[7] Paul, R.J., Balmer, D.W. (1991) *Simulation Modelling*, Chartwell-Bratt, Stockholm.

[8] Quinlan, J.R. (1986) Induction of Decision Trees, *Machine Learning Journal*, Vol. 1, pp. 81-106.

[9] Shapiro, E.Y. (1983) *Algorithmic Program Debugging*. Cambridge, MA: MIT Press.