Volume 25 Number 1 April 2001 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:
Neural Networks and their Applications

The Slovene Society Informatika, Ljubljana, Slovenia

1977

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at USA: http://, Europe: http://ai.ijs.si/informatica, Asia: http://www.comp.nus.edu.sg/ liuh/Informatica/index.html.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2001 (Volume 25) is
- USD 80 for institutions,
- USD 40 for individuals, and
- USD 20 for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, Kranj, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 1000 Ljubljana, Slovenia.

Post tax payed at post 1102 Ljubljana. Slovenia taxe Percue.

Introduction: Neural networks and their applications

Dear Readers,

We offer you a collection of papers devoted to *Neural Networks and their Applications*. The idea for this Special Issue originated when we decided to commemorate the retirement of one of the most prominent researchers in the area of Artificial Neural Network (ANN), Professor Teuvo Kohonen. We issued a Call for Papers for a Special Issue of the *International Journal of Computer Research* and received too many quality submissions to be able to fit them all into a single journal issue. We thus decided to split the material and through the courtesy of Professor Matjaz Gams, Managing Scientific Editor of *Informatica*, we are able to publish all the top papers. We believe this is fortunate as there are now two semi-independent Special Issues devoted to recognizing Professor Kohonen. This Special Issue has papers divided into three distinctive though well-connected parts. They are devoted to: Methodological Considerations, Applications and High Performance Realizations of ANN's.

In the part devoted to Methodological Considerations we have two papers. First, Tadeusiewicz and Lula in *Neural Network Analysis of Time Series Data* present an overview of issues involved in developing neural network based models of one dimensional time series. This paper can be viewed as a step-by-step tutorial on how to proceed and what to take into consideration when applying neural networks to problems involving time series analysis. In the second paper *Applying Neural Networks to Practical Problems - Methodological Considerations*, Paprzycki, et.al., consider that most neural network research is concerned with depth rather than breadth of investigation. They apply seven different neural network architectures (resulting in a total of 15 variants) to the same data representing a simplified pattern recognition problem and compare their performance. These results are used to support the view that more studies of this type are needed to gain further understanding of the interrelations between the problems and the neural network solvers.

The Applications part consists of four papers covering a broad spectrum of ANN usages. First, in *Artificial Neural Network Approach to Data Analysis and Parameter Estimation in Experimental Spectroscopy*, M. Kantardzic et.al. discuss how ANN's can be applied to the film thickness measurement problem, which is one of the crucial steps for many surface-oriented applications. In their approach, initial data is preprocessed to reduce dimensionality and then a backpropagation trained network is applied. Second, in *Control of a One-legged Three-dimensional Hopping Movement System with Multi-layer-perceptron Neural Networks*, Maier et.al. show how an ANN can be applied to control movement of a robotic system based on the dynamics of biological hopping and running. Multi-layered-perceptrons proved suitable for control of movement in such a system, even in a challenging case of movements on an uneven ground. Third, Magoulas et.al, in *Neuro-fuzzy Synergism for Planning the Content in a Web-based Course*, show how to apply neural networks to the decisions in planning the content of a Web-based course. Here, the ANN system is used to adapt the content of the lesson based on the progress of the learner. Finally, in *Identification and Appllication of Logical and Probabilistic Models of Risk in Business* Solojentsev and Karasev discuss how a neural network system can be used in risk assessment. They apply techniques well known in engineering to economic modeling.

The last paper is devoted to the special case when the performance of the artificial neural network is of crucial importance and thus special hardware based techniques are desired. Vitela et.al., in *Performance Analysis of a Parallel Neural Network Training Code for Control of Dynamical Systems* study the performance characteristics of a neural network controlled implementation using the MPI message passing library. The experiments are carried out on an SGI Origin 2000 supercomputer.

We are also grateful and would like to, once more, thank the following referees who helped us in making these special issues of a very high standard indeed.

Peter Butko	Mikko Lehtokangas	Rudy Setiono
Adrian Flanagan	Hongjun Lu	Juha Vesanto
Roger Gaborski	Ron Holyer	Lipo Wang
Jukka Heikkonen	Colin Reeves	Jon Williams
Sami Kaski	Galya Rogova	
Teuvo Kohonen	Ingrid Russell	

Their help was invaluable and we really appreciate their work. Thank you!

Editors of the Special Issue,

P. G. Anderson	G. Klein	E. Oja
N. C. Steele	G. Antoniou	V. Mladenov
M. Paprzycki		

Neural network analysis of time series data

Ryszard Tadeusiewicz
University of Mining and Metallurgy
Mickiewicza Av. 30, 30–059 Cracov, Poland
Phone: + 48 (12) 633–49–98, Fax: (0–12) 633–46–72
E-mail: rtad@biocyb.ia.agh.edu.pl,
neural@uci.agh.edu.pl.

Paweł Lula
Department of Computer Science
University of Economics
av. Rakowiecka 27, 31–510 Cracov, Poland
E-mail: eilula@cyf-kr-edu.pl.

Keywords: neural models, neural networks

Received: June 7, 2000

Neural networks have properties known to be effective in the modeling of economic phenomena. The process of constructing neural models that represent one-dimensional time series is reviewed and demonstrated. Explicit attention is paid to the evaluation of the models. All steps required by theory and practice are demonstrated through an example.

1 Introduction

The primary topic of the present lecture are the possibilities of artificial neural networks[1] applications in the analysis of one–dimensional time series, which are to be understood as sets of values of a certain variable ordered in time. The analysis of the time sequence data requires the application of specific methods – methods taking into account the data variation observed in time and able to describe the respective dependencies. The survey of problems connected with forecasting time series may be found, among others in the works of [Cieślak, 1997 and Zeliaś, 1997]. Examples of applying neural networks to predict various types of time series have been described in the works of [Lai et al., 1999 and Nianyi et al., 1999].

In economy problems, the share of data recorded as time sequences is steadily growing (the phenomenon is mainly connected with the development of capital markets). The nature of these time sequences is also changing – at present they often form long time series, consisting of high frequency data and exhibiting a complex, nonlinear structure. The methods used for analysis, modeling and prediction of such data types are also being improved. Among many research tools applied for description of the data ordered in time the neural networks are also mentioned – mainly because of their ability to describe the nonlinear regularities.

[1]The subject of neural networks has also been mentioned in: [Hertz et al., 1993], [Osowski, 1996], [Tadeusiewicz, 1993], [Tadeusiewicz, 1998]. The possibilities of applying neural models in predicting time series are the subject matter of the following: [Azoff, 1993], [Azoff, 1994], [Baestaens et al., 1994], [Gately, 1996], [Refenes, 1995a], [Thomason, 1997a].

This publication is addressed to persons dealing with theoretical and practical aspects of modeling and forecasting time series. In its further parts, the authors evaluate possibilities of applying neural networks in the modeling of economy phenomena, demonstrate the process of constructing neural models used in modeling of one–dimensional time series and indicate methods of evaluating constructed models. In the final part, theoretical considerations are supported by practical example within which the results of applying the described method in modeling process and forecasting a real economic series are demonstrated.

2 The characteristics of neural models

The neural networks exhibit a set of features, due to which they can become a useful tool for modeling and prediction of socioeconomic phenomena. The propriety of their application follows from certain properties describing the phenomena mentioned above as well as from the structure and functioning of the neural models:

a) Great numbers of phenomena considered in the field of economy (e.g. in finance problems) exhibit nonlinear characteristics, what settles a basic suggestion that tools inherently capable of dealing with nonlinear dependencies should be used for their modeling. Among the tools meeting that requirement are the feed-forward neural networks. They exhibit the ability to approximate arbitrary nonlinear dependencies as well as the ability of generalization.

b) The process of construction of a neural model consists of exploration of all the available data sets and then evaluation of a model able to describe the regularities found in the data. The application of such models does not require the knowledge of the exact form of the function describing the actual dependencies. Because of that property, the neural models can be successfully applied in all the cases when the exact law, describing the formation of the studied dependencies, is not known. It does not exclude the possibility of the network application when the mathematical formulas describing the studied aspect of reality are known, but then the effort connected with evaluation of the neural model can be greater then the efforts required for calculation of parameters of the respective law given in the form of equation.

c) The neural models exhibit adaptive characteristics. They can be used for description of dependencies which vary in time. At the moment when new data arrive, necessary process of additional network training can take place and the information from the recent observation data can be included in the constructed model.

d) The neural network can be regarded not only as a mechanism describing the previous course of the phenomenon and predicting its future values. It can also provide the means for comprehensive analysis of the studied part of reality. The basic information about the system can be acquired by application of model sensitivity analysis. It allows the presentation of nature of the relation between the studied quantity and specific factors which influence its behaviour.

e) The process of evaluation and application of neural models can be carried out concurrently in multiprocessor systems or in a set of computers connected in a computer network. Such a way of realisation of the neural model calculation results in a considerable reduction of time required for the execution of all the necessary calculations.

The application of neural models in not always justified. Such models should not be applied when:

a) the number of observations available to the investigator is not sufficient as a basis for the model evaluation. The evaluation of networks with relatively many parameters using a very scarce learning set most often leads to construction of models which adapt to the learning data, but do not exhibit the ability to describe the general regularities found in the data.

b) the justification for application of the neural models can be hardly found, as in cases when nature of the actual dependence is known (e.g. the function describing the analysed phenomenon can be explicitly written). In such a case the evaluation of the function parameters requires less effort and leads to construction of a model, which is easier in interpretation.

Even when the conditions justifying the application of neural models are fulfilled it should be remembered that in addition to their many advantages the neural models exhibit also some weak points which can be the source of problems during the stage of the model construction. The most important disadvantages of neural networks include:

a) the requirement of proper data preparation – dependent on the nature of the data and the type of network being applied,

b) the problems with selection of the proper neural model structure (type of the network, the assumed neuron model, the number of neurons and the way they are connected),

c) the requirement of proper choice of the network's learning algorithm,

d) relatively high time expenditures connected with evaluation of the neural model,

e) the absence (in most cases) of a direct interpretation of the particular neural model coefficients.

Many works can be indicated, which have been devoted to applications of neural networks in the analysis of economy data (e.g. such problems are dealt with in the following monographs: [Trippi et al., 1993], [Baestaens et al., 1994], [Azoff, 1994], [Refenes et al. 1995]; these problems are also described in papers printed in Journal of Computational Intelligence in Finance, previously known as NeuroVe$t Journal).

3 The construction scheme for a neural model of time series

Several basic stages can be distinguished in the process of construction of a neural model of time series:

- the preliminary data analysis (preprocessing) and the decomposition of the time series
- the construction of partial models, describing the behaviour of each extracted component
- the construction of the aggregate model
- the evaluation of the model's correctness

Fig. 1 is a graphic illustration of the above presented stages of time series analysis.

We will show greater detail particular elements of the suggested method, illustrated by the above scheme can be

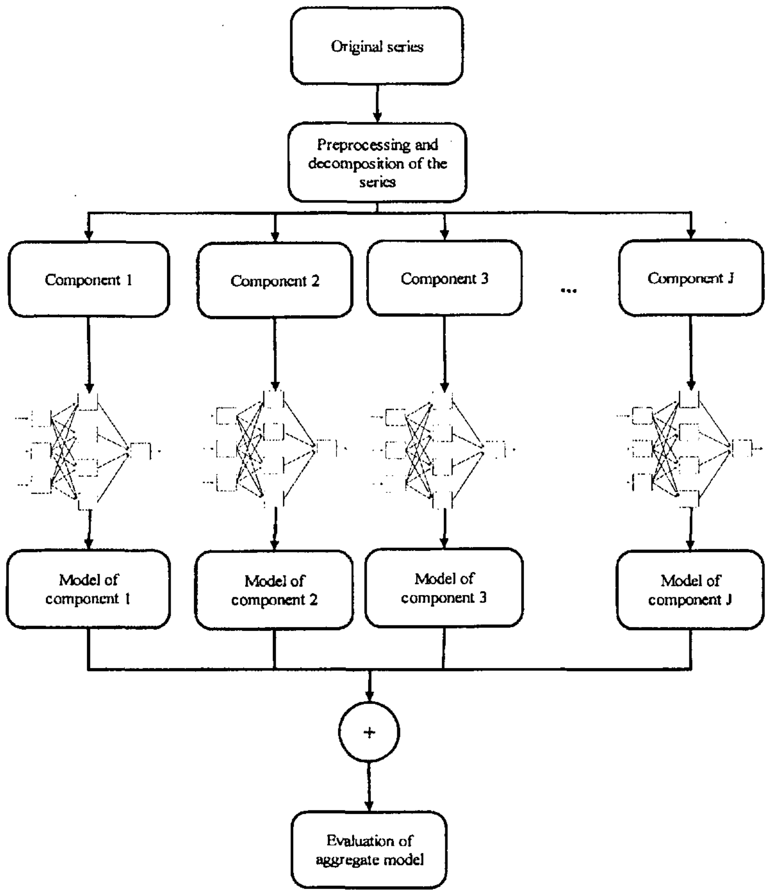


Figure 1: The analysis of one–dimensional time series

found. Preprocessing of original data and decomposition of time series will be discussed in point 4 of the paper. Point 5 demonstrates a construction method of models for particular components. The aggregate model is presented in point 6, whereas the evaluation methods can be found in point 7.

4 The preliminary data analysis and decomposition of the time series

The process of preliminary data analysis and decomposition of the time series includes the following operations:

- the study of data correctness – procedure verifying the data correctness intended to minimize the probability of errors or blanks in the data. Primary methods used in such procedure are: comparison of the data obtained from various sources, graphical analysis of the original time series or a derived series (e.g. relative increments), application of methods used for evaluation of the missing data [Pawełek et al. 1996].

- process of making the data operational - algebraic transformations applied to the original data set in order to enhance certain, particularly important, characteristics of the studied variables. In the case of time series of economic origin, the process of making the data operational usually consists of calculation of the series of relative or absolute increments, dynamic indices, taking the logarithm or square root of the data, or extraction of information concerning the data sign itself.

- preliminary determination of series characteristics – operation aimed at a preliminary test, checking whether any regularities can be found in the data. For the case of finance time series analysis, the preliminary determination of the series characteristics is related to the study of the market efficiency ([Jajuga et al., 1997], [Peters, 1997]). The basic methods used at this stage of analysis are: Hurst exponent, autocorrelation coefficients, spectral analysis, and statistical tests (e.g. BDS test [Lin, 1997]).

- the series decomposition – analysis aimed at the extraction from the original series components with structures simpler than the structure of original data. The series decomposition can be carried out by: fitting the trend function and extraction of the differences, filtration of the data, or determination of the season variation indices. The studies carried out by the authors indicate that the discrete wavelet transform can be particularly useful ([Thomason, 1997]). For the case of time series with simple structures, the series decomposition is not applied.

- preprocessing of the series components – consists of a transformation of values of the extracted components before they are processed by the neural network. The necessity of such transformation follows from the specific features of the applied neural models (the necessity to adapt the variability range of the processed values to the range of values generated by the output neurons). The way in which the transformation is performed in most cases is not related to the system generating the data.

5 The construction of partial models

The next stage of the analysis assumes the construction of **k** independent models describing each of the extracted components. Because of the diversified character of data forming each of the components it can be expected that the structure of particular models will be different. Still the construction procedure for each of the models is similar and requires the solution of many similar problems. The basic questions which have to be answered during the construction of each model include:

- choice of the neural network type – the possibilities usually taken into account include the application of feed–forward multi-layer networks (multi-layer perceptrons), the networks with radial basic functions, linear networks or generalized regression neural network. The choice of the proper network type depends mainly on the character of the described phenomenon and the structure of data.

- choice of the input variables – the models of one–dimensional time series are usually of autoregressive nature (they describe the dependence between the present value in the time series and its previous occurrences). The construction of a properly working model requires the identifications of the previous values necessary for evaluation of the consecutive values in the series. The results of the studies carried out by the authors confirm the usefulness in that field of methods employing the genetic algorithms.

- specification of the model structure – consisting mainly of the choice of hidden part of the neural network (the specification of the number of layers and the number of neurons in the hidden part and choice of neuron model). Particularly promising results can be obtained using the genetic algorithm. However, the considerable computer time requirements of such model construction should be kept in mind.

- the network learning – evaluation of the model parameters in a way leading to minimization of the assumed error function.

- evaluation of the correctness of the partial model – mainly by using instruments based on the differences between the actual and theoretical values (sum of squares or modules of the differences, mean-square error, linear correlation coefficient).

The order of consecutive stages is not always consistent with the sequence presented above, because some phases of the process are often performed many times during the construction of a single model. As an example, one can consider the network's learning, which is often performed during the procedure of selection of the model's input information as well as during the specification of the appropriate model structure. Therefore, unsatisfying results obtained from the network's learning can be a direct suggestion for the structure modification or even for the application of another network type.

The model construction process has to be repeated for every separate component extracted from the data.

6 Construction of the aggregate model

After estimation of all the partial models an aggregate model can be constructed, which cumulates all the results obtained using the partial models. The data aggregation is carried out in a way reverse to the process of the original series decomposition (usually by executing the summation or multiplication of the partial results).

The theoretical values of the aggregate model are obtained by aggregation of the results calculated by the partial models.

If the elements of all extracted components have been uniformly included in the learning, validation and testing sets, then the respective sections of series of theoretical results of the aggregate model can be attributed to learning, validation and testing characteristics. Due to that relation a possibility emerges to estimate the model quality measures independently for each set.

7 Evaluation of model's correctness

The aggregate model allows the calculation of theoretical values for the series. If the investigator wants to evaluate the model's correctness, then the results obtained from application of the model can be compared with the stored real cases. The employed evaluation process should involve many aspects of the problem. On one hand it should take into account the obtained values by applying the chosen error measures and prefer the models for which the error value (or values of other statistics based on the error value) is the least. This group of methods of correctness evaluation is of universal nature, because it can be applied to arbitrary series, irrespective of their origin. The second group of methods requires substantial knowledge of the system generating the data. It provides evaluation of the obtained predictions, which takes into account the nature of the studied phenomenon (the evaluation of the same numeric values of errors will be different in a system predicting the market prices and in a system, which controls the production of some precise devices).

Universal measures of the prediction quality can be applied for evaluation of predictions of future values for an arbitrary time series, obtained by arbitrary method. These measures are based on the comparison of the actual value a_t with the theoretical value p_t.

Among the classical measures of the prediction correctness one can find ([Caldwell, 1995]) average error, mean square error, square root of the mean square error, normalized mean square error, the absolute error, the percentage error, correlation coefficient, determination coefficient, and the Theil's I^2 coefficient ([Zeliaś, 1997], [Cieślak, 1997]).

During the model's evaluation one can also make use of the *relative measures*, comparing the quality of prediction obtained from the constructed model and the model which is used as a reference. The quality of neural model's predictions is usually compared to the predictions obtained from linear models or to naive predictions.

A widely applied technique for the analysis of the model's quality is the analysis of the series of remainders, aimed at checking whether:

- the predictions generated by the model are not biased
- the average error of the prediction is uniformly distributed within the studied range of time
- the consecutive terms of the series of remainders are not mutually correlated
- the predictions obtained from other models are not better than the ones obtained from the studied model.

The analysis of the series of remainders can be done visually, or can be carried out using more sophisticated statistical tools.

The universal methods of evaluation of the model's correctness do not always fully reflect the quality of economic prognostic systems, because the estimated quality measures mainly provide information about the precision of the obtained predictions, which does not always provide a good basis for evaluation of the model's utility in computer–aided decision making. For evaluation of a prognostic system the accuracy of the decisions taken with its help should be evaluated in the first place. The respective model evaluation system depends on the field of the model's application.

Financial prediction quality measures determine the model's utility in the process of finance decision making. The constructed instruments are closely related to the field of application and the way the model's output results are interpreted. In finance applications, greater importance is often attributed to the correctness of the predicted trends than to the precision of the calculated prognosis. Therefore the Directional Symmetry (DS) coefficient can be useful, as it calculates the percentage of cases in which the actual trend was in accordance with predicted trend. The exact value of the instrument is defined as follows:

$$DS = 100 \cdot \frac{\sum_{t=1}^{n} d_t}{n}$$

where

$$d_t = \begin{cases} 1 & \text{if } (a_t - a_{t-1})(p_t - p_{t-1}) > 0 \\ 0 & \text{otherwise} \end{cases} .$$

It is worth noticing that the DS coefficient value of 50exactly half of the cases have been predicted correctly, or equivalent to the result that can be obtained by choosing the trend direction at random.

If a prognostic system is expected to be helpful in investment decision aiding on the finance markets, then a basic measure of the prediction system quality should be the profit resulting from the realization of investments made basing on the calculated predictions. Then the efficiency of the studied investment strategy is usually compared with strategies used as a certain reference system (ideal strategy, the buy and keep strategy or others). Model systems of investment strategy evaluation have been presented in [Lirov, 1993], [Caldwell, 1995] and other works.

8 Construction of the model of real time series

To illustrate the above proposed method of modeling one–dimensional time series, the construction process of a certain definite model will be demonstrated. The authors have decided to present a neural model of time series describing a monthly number of foreign airline passengers. Data have been derived from the work of [Box et al., 1983] and therefore a practical illustration of functioning methods as well as its evaluation based on standard data in which results obtained through other methods are possible. The length of the time series amounts to 144 observations.

In the preprocessing stage, scaling has been performed according to:

$$x_i^s = \frac{x_i}{\max_i(x_i)} \tag{1}$$

The scaling method is justified because all values of discussed series are greater than zero. Calculations connected with the transformation of original data have resulted in a set of values from the range:

$$\left[\frac{\min_i(x_i)}{\max_i(x_i)}; 1\right] .$$

Next the scaled series has been submitted to the decomposition process by means of a selected method of splitting the series into particular components. For this purpose discrete wavelet transformation was to be applied. All necessary calculations were carried out by means of *TimeStat* program in version 1.2. Decomposition process created seven component series taking into account variations of different frequencies.

The following step in the calculation was the evaluation of neural models indicating regularities occurring in each of the seven components. For this purpose each obtained series was transformed into a set of pairs, $\{\langle\text{explaining values}\rangle; \langle\text{explained value}\rangle\}$, where explained value was a consequent element of series. In the character of explaining variables a series of 24 preceding values was adopted. Taken into consideration of the number of delayed elements of the signal is the seasonal character in given consecutive months of the year. Therefore a set of data comprising two years may constitute a basis for creating and learning particular network. It should be stated, that transformation of vector consisting of 144 elements on 120 pairs $\{\langle\text{vector}\rangle; \langle\text{scalar}\rangle\}$ was conducted in the same way for each component.

The stage of initial transformation and decomposition was followed by a division of possessed 120 elements – at random – on a learning, validation and testing set. For this purpose 80 elements were drawn from the all set and added to a learning set. Next 20 elements were drawn from the remaining set and included in the validation set. Remaining elements were included in the testing set. It has to be stressed that an identical division was applied to each of the different components.

A further stage aimed at defining the structure and learning, describes the behaviour of consecutive components. On the basis of preliminary studies, each of the components was described by a 24–4–1 model. Neurons in the hidden layer had a tan–sigmoid activation function and input neurons were equipped with a linear one. The proposed model had 105 parameters. The number of parameters was too large when compared with the length of learning series but it was assumed that in the next stage of studies useless parameters of the model would be eliminated. Next step was the evaluation of parameters of the models describing behaviour of distinguished components. Learning was carried out in parallel to the optimisation of model structures. In the process of learning and generating proper structure, the RPROP algorithm and the genetic algorithm were applied. The genetically minimised fitting function was dependent on the error value for the learning set (describing ability to approximate) and for the validation set (describing ability to generalize) as well as the number on inputs and neurons in a hidden layer. The applied algorithm aimed at the minimisation of all mentioned elements of the fitting function. During calculations a population of 30 chromosomes was used. Each of them described the structure of one network model. The above method was repeated 7 times – separately for each component.

Table 1: Basic parameters of models describing components distinguished in a studied series

Component	Number inputs	Number of hidden neurons	Number of weights
1	4	2	12
2	4	2	13
3	5	2	12
4	2	2	7
5	1	2	7
6	1	2	7
7	1	1	4

The outcomes of that procedure were seven models describing behaviour of each distinguished component. Obtained models were different in terms of number of inputs and neurons. Total specification of information concerning obtained networks has been presented in table 1.

The analysis of the obtained network models indicates that in order to calculate theoretical values of consecutive components, past values are used differently-delayed in relation to a described value. To describe the values of components with index t the following are applied:

- in model of the component 1 the values with indexes: $t-1, t-3, t-8, t-12$ are used,
- in model of the component 2 the values with indexes: $t-1, t-2, t-15, t-24$ are used,
- in model of the component 3 the values with indexes: $t-5, t-9, t-16, t-20, t-22$ are used,
- in model of the component 4 the values with indexes: $t-1, t-14$ are used,
- in model of the component 5 the value with index: $t-1$ is used,
- in model of the component 6 the value with index: $t-3$ is used,
- in model of the component 7 the value with index: $t-1$ is used.

It has to be stressed that the method of selecting input variables based on a genetic algorithm may be applied independently of the neural network.

After evaluating all partial models an aggregate model was created by summing up results obtained by means of each model component. The basis for the evaluation of an aggregate model may be values of selected measures (these values were estimated by means of using original data) presented in table 2.

Obtained results confirm the utility of the adopted method of creating time series models.

Table 2: Measures of quality for an aggregate model (calculated on the basis of original data)

Measure	Learning set	Validation set	Testing set
SSE	37476	8839,3	13672
MSE	468,4526	441,9673	683,6135
RMSE	21,6438	21,023	26,1460
NRMSE	0,1865	0,1986	0,2942
R^2	0,9648	0,9585	0,9089

9 Final conclusions

In the present paper a method of creating neural models of time series has been presented. The authors have shown the results of its evaluation on the basis of a data set, which is a standard benchmark in ranking various methods of modeling of time series. It is worth stressing that the presented procedure of modeling is in fact a hybrid method as it makes use of possibilities offered by neural networks, genetic algorithms and discrete wavelet transformation. Combinations of features of data analysis may lead to the construction of models describing time series with complex structure. It is interesting for both theoreticians and practitioners. Models like these have various applications of which the most essential are various kinds of forecasting. Having an effectively working model the prediction of future values is possible on the basis of observing sequence of earlier data. It is particularly useful in forecasting future values of economic indexes.

Acknowledgements. Authors thank Gary Klein, Marcin Paprzycki and the anonymous reviewer for comments, suggestions and improvements in the text.

References

[1] Azoff E. M. (1993) Reducing Error in Neural Network Time Series Forecasting. *Neural Computing and Applications*, 1.

[2] Azoff E. M. (1994) Neural Network Time Series Forecasting of Financial Markets. *John Wiley & Sons.*

[3] Baestaens D. E., van den Bergh W. M. & Wood D. (1994) Neural Network Solutions for Trading in Financial Markets. *Pitman Publishing*, London.

[4] Bandy H. B. (1995) The Adaptive Moving Average. *NeuroVe$t Journal*, July/August.

[5] Box G. E. P. & Jenkins G. M. (1983) Analiza szeregów czasowych. Prognozowanie i sterowanie. *PWN*, Warszawa.

[6] Caldwell R. (1995) Performance Metrics for Neural Network–based Trading System Development. *NeuroVe$t Journal*, March/April.

[7] Cieślak M. (ed.) (1997) Prognozowanie gospodarcze. Metody i zastosowania. *Wydawnictwo Naukowe PWN*, Warszawa.

[8] Gately E. J. (1996) Neural Networks for Financial Forecasting. *John Wiley & Sons.*

[9] Hertz J., Krogh A. & Palmer R. G. (1993) Wstęp do teorii obliczeń neuronowych. *WNT*, Warszawa.

[10] Jajuga K. & Jajuga T. (1997) Inwestycje – instrumenty finansowe, ryzyko finansowe, inżynieria finansowa. *Wydawnictwo Naukowe PWN*, Warszawa.

[11] Lai L. L., Subasinghe H., Rajkumar N., Vaseekar E., Gwyn B.J. & Sood V.K. (1999) Object–oriented genetic algorithm based artificial neural network for load forecasting. *Second Asia–Pacific Conference on Simulated Evolution and Learning*, SEAL'98. Selected Papers. Springer–Verlag, Berlin, pp. 462-469.

[12] Lin K. (1997) The ABC's of BDS. *Journal of Computational Intelligence in Finance*, July/August.

[13] Lirov Y. (1993) Performance Evaluation of Automated Investment Systems. *NeuroVe$t Journal*, November/December.

[14] Masters T. (1995) The General Regression Neural Network. *NeuroVe$t Journal*, September/October.

[15] Nianyi Chen., Wang W. & Zhu D.D. (1999) A study on stock data mining by map recognition. *Proceedings of the Second International Conference on Information Fusion.* Part 2, pp. 1001-1007.

[16] Osowski S. (1996) Sieci neuronowe w ujęciu algorytmicznym. *WNT*, Warszawa.

[17] Pawełek B. & Zeliaś A. (1996) Obserwacje nietypowe w badaniach ekonometrycznych. *Badania operacyjne i decyzje*, 2.

[18] Peters E., E. (1997) Teoria chaosu a rynki kapitałowe. *WIG - Press*, Warszawa.

[19] Refenes A. P. & Zaidi A. (1995) Managing Exchange – Rate Prediction Strategies with Neural Networks. in: [Refenes, 1995]

[20] Refenes A. P. (ed.) (1995a) Neural Networks in the Capital Markets. *John Wiley & Sons.*

[21] Tadeusiewicz R. (1993) Sieci neuronowe. *Akademicka Oficyna Wydawnicza RM*, Warszawa.

[22] Tadeusiewicz R. (1998) Elementarne wprowadzenie do sieci neuronowych z przykładowymi programami. *Akademicka Oficyna Wydawnicza*, Warszawa.

[23] Thomason M. R. (1997) Financial Forecasting with Wavelet Filters and Neural Networks. *Journal of Computational Intelligence in Finance*, March/April.

[24] Thomason M. R. (1997a) Residual Analysis for Neural Network Financial Predictors: An Introduction. *Journal of Computational Intelligence in Finance*, July/August.

[25] Trippi R., Turban E., (eds) (1993) Neural Networks in Finance and Investing. *Probus Publishing Company.*

[26] Wong F. (1994) Neurogenetic Computing Technology. *NeuroVe$t Journal*, July/August.

[27] Zeliaś A. (1997) Teoria prognozy. *Polskie Wydawnictwo Ekonomiczne*, Warszawa.

Applying neural networks to practical problems — methodological considerations

Marcin Paprzycki, Aaron Costeines, William Douglas, Paul Gatling, Rick Niess and Lenny Scardino
Department of Computer Science and Statistics,
University of Southern Mississippi,
Hattiesburg, MS 39406-5106, USA
Phone: +1 601 266 6639, Fax: +1 601 266 6452
E-mail: m.paprzycki@usm.edu

Keywords: neural networks, performance comparison, simplified multifont recognition problem

Received: May 30, 2000

Results of performance comparison of seven neural network architectures applied to a simplified multifont recognition problem are used to illustrate the need for more comparative performance studies. This need is a result, among others, of increasing computational power of modern PC's and development of commercial neural network software packages that combine multiple NN architectures in one easy-to-use environment. Experimental data supporting our conjecture is presented and discussed.

1 Introduction

Neural networks (NNs) are one of the more popular tools for broadly understood *patternrecognition*. They are applied to problems in many different disciplines (see for instance [6, 11] and the references collected there). When re-introduced into the computational practice [16], mostly a multilayer perceptron architecture and backpropagation training algorithm were in the center of attention. Since then a large number of NN architectures and training algorithms have been proposed and experimented with. Some important differences are in the number and size and connection patterns of layers of neurons, the weight adjustment functions applied during the learning process, and the way that the learning process proceeds (e.g. with or without supervision). While intial interest in neural networks was centered around research in psychology, philosophy and cognitive science, this interest seems to have somewhat faded, recently. At the same time, neural networks have been embraced by practicioners who are capable of succesfully applying them to problems in a number of fields (control, economics, business, vision etc.). However, as our review of literature and the Internet indicates, only a limited amount of work has been devoted to comparing performances of various NN architectures and training algorithms. Rather, researchers concentrate their attention on a single NN architecture and its training method as applied to a given problem. Their goal is to fine-tune the performance of the network to achieve the best results for the problem in hand. In addition, before the data is shown to the neural network, it is often pre-processed by a feature extracting software (to reduce the size/domain of the problem). Obviously, this adds to the difficulty of fairly comparing the performance claims of different NN architectures (as related to the original problem) as the performance of the neural network is now mediated by the performance of the feature extractor.

In this context we would like to make two observations. First, neural networks have moved away from academia and a number of relatively affordable commercial products have been developed. Some of these products, like NeuroShell and Trajan, are stand alone NN packages while others, like the NN kit for the MATLAB mathematical package, extend the capabilities of other software products. In all of them, a number of neural network architectures are presented together in a single package with a common interface, thus making it easy for an inexperienced user to try to use the software to solve a given problem. This makes the question of which architecture to apply even more important. Second, the power of a typical desktop PC has increased enormously since the days of the Rumelhardt and McClelland book. In the early days some NN architectures were considered almost impossible to use in computational practice due to the time it took to train them (e.g. recurrent neural networks [9]), or the amount of memory required (probabilistic neural networks [19]). These limitations, among others, were responsible for the need to drastically reduce the size of the input via feature extraction software. The question that we would like to address is: given the advance in computational resources since that time, are these limitations still valid?

To address these questions, we report on our attempt to compare the performance characteristics of a number of neural network architectures available in Ward Systems' NeuroShell 2.0 package [13] as applied to a simplified pattern recognition problem. This software has been used in earlier work [1, 2, 3, 4]. Results reported there were obtained for a relatively small size input data and/or for a selected few NN architectures. In this paper, we try to assemble a more complete picture by reporting on experiments (a) with all NN architectures available in the NeuroShell package that are applicable to our problem and (b) with a significantly larger set of training data. Similarly to the results collected earlier, we used a simplified model-problem of recognition of computer-printed characters (26 upper case letters of the Latin alphabet). This arrangement approximates the situation of a computer generated printed character being scanned, digitized, and presented to a computer program to be recognized. It is assumed that one letter is presented at a time and that each image presented does indeed represent a letter. Since we are interested in the performance of the NN architectures themselves and since we would like to establish feasibility of such approach, we have applied them directly to the digitized data (without any pre-processing).

The remaining parts of the paper are organized as follows. In the next section we very briefly introduce the NN architectures used in our experiments. We follow it with a description of how the input data was generated and the experimental set-up. Finally, we present and discuss the experimental data.

2 Neural network architectures

In our work, we considered four architectures that use backpropagation as the training algorithm (multilayer perceptron, jump connection networks, recurrent networks and Ward networks). In the NeuroShell package, each of these architectures is available in three variants (discussed below). In addition, we report on the results collected when the probabilistic, general regression and Kohonen networks were applied to the problem. Let us now briefly describe each of these architectures (consult references, as well as [6, 11], for more details).

2.1 Multilayer perceptron (MPN)

The mutilayer perceptron combined with backpropagation training is one of the best known and most studied NN architectures. It consists of an input layer, at least one hidden layer, and an output layer. NeuroShell provides three implementations of the MPN, with one, two, and three hidden layers (referred to as "MPN(1)", "MPN(2)", and "MPN(3)", respectively, hereafter). The default number of nodes in the hidden layer is calculated using the formula $\#nodes = 1/2(Inputs + Outputs) + \sqrt{\#\ patterns\ in\ training\ set}$. By default, the **total** number of nodes in all hidden layers remains the same in all three implementations. Thus if a network with one hidden layer has 300 nodes in it, a network with two layers will have 150 nodes per layer and a network with three layers will have 100 nodes per layer. A *linear* $[-1, 1]$ activation function is used in the input layer and the *logistic* function is used in hidden and output layers.

2.2 Jump connection networks (JCN)

Jump connection networks differ from the MPN architectures in that a number of additional forward connections are added [13]. NeuroShell includes JCNs with one, two, and three hidden layers (we follow the naming convention set forth in the MPN description above). As illustrated in Figure 1, in the case of a jump connection network with three hidden layers, the input layer is connected forward with the remaining four layers (three hidden layers and the output layer), the first hidden layer is connected with the second and third hidden layers and the output layer, etc. Because of this, it is assumed that each successive layer will have a more complete view of the data from several different stages of processing. The number of nodes in the hidden layers is governed by the same formula as the MPNs. Also, backpropagation is used as a training algorithm and the activation functions remain the same as with the MPNs

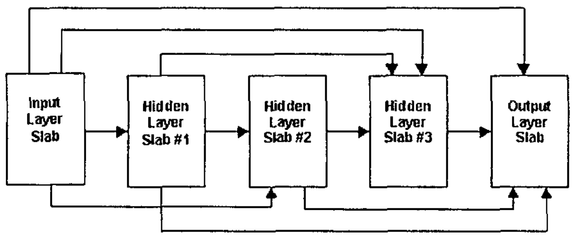


Figure 1: Jump connection neural network architecture, JCN(3) version.

2.3 Ward networks (WNN)

Ward networks are a proprietary NN architecture available in the NeuroShell package [13]. They are a modification of the simple MPN architecture in that they have multiple parallel "slabs" (independent groups of nodes) in a single hidden layer. As with the simple MPN, each of these slabs receives input from the input layer and is connected directly to the output layer. As previously, the same formula guides the selection of the total number of nodes in the hidden layer. However, WNNs differ from the standard MPN schema in that each of the hidden layer slabs uses a different activation function. Because of this, it is assumed that each slab will be responsible for recognizing different features of the input pattern. There are three different variants of this architecture available in the NeuroShell package. Following the naming convention set forth in the MPN description above, the WNN(2) is constructed using two parallel slabs in the hidden layer which use *Gaussian* and *Gaussian compliment* activation functions respectively. The WNN(3) has three parallel slabs which use the *Gaussian*, *Gaussian compliment*, and *tanh* functions respectively (see Figure 2). Finally, the WNN(2j) is identical to the WNN(2) except that it has an additional "jump-connection" from the input layer directly to the output layer.

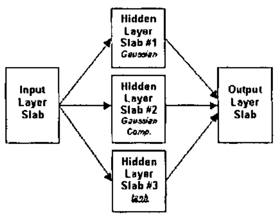

Input Layer Slab
Hidden Layer Slab #1 Gaussian
Hidden Layer Slab #2 Gaussian Comp.
Hidden Layer Slab #3 tanh
Output Layer Slab

Figure 2: Ward neural network architecture, WNN(3) version.

2.4 Recurrent neural networks (RNN)

Recurrent networks with dampened input layer feedback (Jordan-Elman networks, [5]) are a modification of t2he MPN in that an additional slab of neurons is added to represent a "long-term memory effect." In NeuroShell, this slab of neurons always gives output to the hidden layer, but can receive input from the input layer, hidden layer, or output layer. (We will referr to these as "RNN(il)", "RNN(hl)", and "RNN(ol)", respectively.) For instance, when the additional slab is to interact with input patterns, it combines inputs from the external data and the standard input slab as well as the inputs originating from itself (via dampened loop-back connections - memory effect) and gives output to the hidden layer. Figure 3 represents the RNN(hl) version of the recurrent architecture available in the NeuroShell package.

While a regular feed forward network responds to a given input pattern with exactly the same output every time the given input pattern is presented, a recurrent network may respond to the same input pattern differently at different times, depending on the input patterns which have been presented to it previously (memory-effect). This often makes this network desirable for time-series evaluation [8]. Recurrent networks are trained the same way as standard MPNs except that training patterns must always be presented in the same order. Activation functions for the input, hidden, and ouput layers also remain the same as with the MPN.

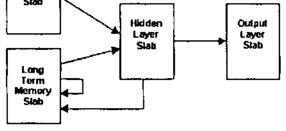


Figure 3: Recurrent neural network architecture, RNN(hl) version.

2.5 Probabilistic neural networks (PNN)

Probabilistic Neural Networks are based on an application of a classification theory based on the minimization of the "expected risk" function [17, 18, 19]. Here the network's determination is based on a series of measurements and can be expressed in terms of Bayesian decision rules. The key to the classification process is the ability to estimate the probability density functions (PDFs). For NN applications this means that the PDFs have to be estimated on the basis of the training patterns (which are the only source of available information about the data). In 1962, Parzen introduced a class of estimators that asymptotically approach the real density as long as it is smooth and continuous [15] and Specht used it in the PNN design. PNN requires that all information from the training set be stored and used during testing. In other words, for each input data there is a node in the hidden layer (which can lead to a subtantial amount of memory necessary to implement this architecture). Training is relatively fast as each input is shown to the network only once, however the time necessary to process the data after the network is already trained is directly proportional to the size of the training set.

2.6 General regression networks (GRN)

The general regression networks are very similar to the PNN's as they are also based on estimation of probability density functions. Originally the concept was developed in the statistics literature and known as the Nadaraya-Watson kernel regression and was translated to the NN environment by D. Specht [20]. The most important difference between PNN and GRN is in the fact that the PNN distinguishes between discrete states, while the GRN generates continuous output. As with the PNN, the GRN uses as many neurons in the hidden layer as there are input elements and trains quickly. However, processing time after the network has been trained is also directly proportional to the number of patterns used in training.

2.7 Kohonen networks (KNN)

Kohonen's networks were introduced in 1989 [10]. They are rather "simple," with only two layers (input and output), and are based on competitive unsupervised learning. When the training data is shown to the network it tries to separate it into a specified number of categories. The weight adjustment process during learning is based on the Kohonen self-organizing map and attempts at categorizing data elements into discrete groups based on the perceived similarity between elements.

3 Data generation

To generate the input data we developed a font-digitizer based on the FreeType 1.2, an open-source TrueType font rendering library [7]. The resulting "bitmaps" (sequences of integers 0 and 1) represented centered, similarly scaled letters allowing a "one-pixel" border around the map (the effective letter image was at most 18 × 18). Initially, we collected 2450 fonts. These fonts were then manually compared to remove identical and similar fonts that exist under

	MPN(1)			MPN(2)			MPN(3)		
# nodes	min	max	ave	min	max	ave	min	max	ave
100	88.39	88.99	88.61	88.50	89.79	89.43	88.08	88.95	88.61
200	88.99	89.76	89.47	90.80	91.05	90.95	90.84	91.01	90.94
300	89.51	90.59	90.05	91.22	91.78	91.58	91.47	92.03	91.65
400	89.37	91.28	90.18	91.54	91.78	91.67	91.61	92.27	91.88
defaulti(324)	88.67	90.24	89.72	90.98	91.81	91.44	91.05	91.43	91.23

Table 1: Performance comparison; multilayer perceptron; one, two, and three hidden layers; results in % of correct answers

	JCN(1)			JCN(2)			JCN(3)		
# nodes	min	max	ave	min	max	ave	min	max	ave
100	83.78	86.96	85.04	77.83	84.72	81.57	84.13	86.22	85.21
200	81.40	86.08	84.11	83.53	85.38	84.58	82.87	85.03	83.82
300	80.00	82.97	82.21	73.81	80.98	78.07	74.16	84.41	78.20
400	80.70	74.27	79.61	77.87	80.45	79.77	80.28	84.37	82.07
default(324)	74.97	84.22	80.48	78.85	76.36	76.93	79.55	82.66	81.33

Table 2: Performance comparison; jump connection networks; 1,2,3 hidden layers; results in % of correct answers

different names. Only the basic forms of each font were allowed (no italic or script versions have been used). After three screenings, we ended up with 710 unique fonts. The fonts were then processed by our digitizer to obtain data in the format required by the NeuroShell environment. For all networks we used 400 input nodes ($20 \times 20 = 400$, each node corresponding to one element of the input vector) and 26 output nodes (corresponding to the 26 letters of the alphabet).

Initially, the input data was divided into 6 groups of 100, 200, ..., 600 fonts which were used for training. We used the remaining 110 fonts for testing. The data was divided into groups alphabetically (based on the font names). However, since there is no relationship between the name of the font and its shape, this approach had no effect on the results. The preliminary results of these experiments were reported in [14]. As expected, as the number of available inputs increased, the quality of answers improved. In this paper we report mostly results for the largest input data set, comprising of 600 fonts ($26 \times 600 = 15600 input - vectors of length 400$).

In all backpropagation based networks, the input data is divided into training and verification sets. We have accepted the NeuroShell default of 20% of input data being devoted to in-training verification. To observe the stability of the results, we have run our experiments for each architecture using 5 different random divisions of data into training (80%) and verification (20%) sets. We indicate the variability of the results below.

For all networks, unless otherwise indicated, we have used the deafult values of various parameters e.g. learning rate, stopping criteria etc. While the effect of these parameters on the quality of resulting network is an interesting subject warranting possibly a separate study, it goes well beyond the scope of this paper.

4 Experimental results

We will now report on the results of our experiments. In Table 1 we summarize the results obtained when the MPN(1), MPN(2), and MPN(3) multilayer perceptron variations were applied to the problem. Five training trials were conducted for each network using five different divisions of the data between the training and verification sets. We report the minimum, the maximum and the average performance.

The behavior of the network is relatively good with the recognition rate at about 88-92%. Slightly unexpectedly, the MPN architecture with three hidden layers outperformed others. In addition, we are finding out that the default value of the neurons in the hidden layer does not lead to the best performance. Here, the best recognition rate is achieved for 400 nodes in the hidden layers (regardless of the number of hidden layers). This is surprising since 400 is more than the default number suggested by the environment. The results are also relatively stable. In all cases the difference between the best and the worst result is of the order of one to two percent.

In Table 2 we summarize the results collected for the jump connection networks with one, two and three hidden layers. The results are reported in the same form as Table 1.

The performance of JCN is substantially worse than that of MPN. In addition, in a very large departure from the suggested default number of nodes in the hidden layer, the best performance for is achieved for networks with 100 nodes (approximately 1/3 of the nodes suggested by the default function). Here, instead of improving, performance appears to degrade when more nodes are added to the hidden layer. This behaviour is interesting and merits further study. Finally, it is difficult to pick an overall winner since, depending on the number of the hidden layers and nodes in them, different architectures carry the best result. This

	WNN(2)			WNN(3)			WNN(2j)		
# nodes	min	max	ave	min	max	ave	min	max	ave
100	1.19	2.93	1.85	82.59	84.13	83.30	55.52	63.15	58.58
200	2.34	7.93	5.09	85.52	86.68	86.11	53.60	58.56	56.84
300	4.34	5.66	5.27	86.40	88.11	87.01	49.41	60.63	54.86
400	3.63	7.83	5.73	83.60	88.46	87.05	50.28	59.55	56.32
default(324)	2.10	7.06	5.15	86.15	88.22	87.54	55.86	60.80	56.48

Table 3: Performance comparison; Ward networks; three architectures; results in % of correct answers

	RNN(il)			RNN(hl)			RNN(ol)		
# nodes	min	max	ave	min	max	ave	min	max	ave
100	93.74	93.77	93.75	91.92	92.97	92.36	89.97	90.84	90.34
200	93.70	94.05	93.81	92.48	93.74	92.81	89.51	90.70	90.11
300	93.98	94.75	94.23	92.03	93.85	93.02	90.03	90.66	90.30
400	93.94	94.02	93.98	92.55	93.53	93.05	90.52	91.29	90.88
default	94.02	94.33	94.17	92.76	93.32	92.97	90.56	90.87	90.7

Table 4: Performance comparison; recurrent neural networks; three architectures; results in % of correct answers

effect is combined with a relative "instability" of the network architectures. The differences between the best and the worst performance are as much as ten percent.

In Table 3 we summarize the performance of the three Ward networks architectures. Again, the results are reported in the same form as Table 1.

The results are surprising. The WNN(2) architecture behaves almost as it has not learned at all. One can assume that accuracy of 5% could almost be achieved by randomly guessing the answer (since there were 110 fonts and thus 2860 letters, then if letters were selected randomly with probability 1/26 a total of 110 letters on average should have been selected resulting in accuracy of about 4.1%).

The WNN(2j) architecture is better, but reaches only 59%. Only the WNN(3) architecture, with three parallel slabs in the hidden layer performs well reaching a recognition level of almost 88%. It is also the most stable architecture of the three, with the differences between the best and worst performance staying below 4%. Interestingly, it is the default number of neurons in the hidden layer that results in the best performance for this architecure.

In Table 4, we summarize the results for the last of the backpropagation-trained neural networks - recurrent neural networks. The column names match the information about which layer was connected with the additional input slab (il - input layer, hl - hidden layer, ol - output layer).

It can be observed that the performance of RNN is very stable; the differences between trials stays below 2%. The best performance is achieved for the architecture where the memory slab is attached to the input layer. This is slightly different than expected as the network with the memory slab attached to the hidden layer (layer where the features are dealt with) was expected to outperform others. It is impossible to specify a particular number of nodes in the hidden layer that outperforms others as this number depends on the architecture used (it is 300 for RNN(il) and RNN(hl), but 400 for the RNN(ol)). It can, however, be said that the differences are small enough to achieve satisfactory performance using the default setup.

Table 5 contains the data collected for the three non-backpropagation based architectures. Here a slightly different set of data is presented. Since all three architectures use all available data to build the neural network we did not have different extracts to use. We have thus decided to present the effect on the performance of adding more and more fonts to the training set (100, 200, ..., 600). Usage of the PNN and GRN leads, however, to an interesting "dilemma." After the training their predictive power depends on the value of the smoothing factor. If the smoothing factor is too small than the network cannot recognize some of the patterns shown to it (seeing them as too distinct from the data it has been trained with). If the smoothing factor is too large, the network starts to overgeneralize and thus misclassifies patterns. In theory, each trained network can have its own optimal value of smoothing factor and this value cannot be known a'priori. To deal with this problem, a calibration procedure is provided. Here, a part of the training data is used to estimate an appropriate smoothing factor. It is assumed that if the patterns, which will be shown to the network in the future, are similar to those used for calibration (and thus similar to those used in training), then the selected smoothing factor will be close to optimal. We thus report the efficiency of PNN and GRN when the best smoothing factor was found "hands-on" (best) and then the automatic calibration was used (auto).

As reported in our earlier work and in the literature [3, 6] the performance of the Kohonen network is very bad. The self organizing map is incapable of properly grouping the letters. While for 100 fonts the PNN slightly outperforms the GRN. When 600 fonts are reached, the situation reverses. The calibration process, which was not doing well when the number of fonts was small [1], becomes a relatively efficient method for estimating the smoothing factor for the large number of fonts. For 600 fonts it is able to

	PNN		GRN		KNN
# fonts	auto	best	auto	best	
100	81.61	83.52	75.20	83.45	2.83
200	84.58	86.90	78.05	86.96	3.07
300	87.68	88.02	83.27	88.34	2.79
400	88.68	89.40	85.52	89.55	3.56
500	89.31	90.06	86.46	90.06	1.60
600	89.81	90.49	87.14	90.81	5.62

Table 5: Performance comparison; PNN; GRN; KNN; results in % of correct answers

reach within one percent of the best possible performance for PNN and within four percent for GRN.

One of the interesting observations made above was the fact that the specified default number of nodes in the hidden layer did not necessarily lead to the best performance. We could see that this number does not depend only on the number of input nodes, output nodes and number of elements in the input data, but also on the NN architecture. We have decided to investigate this more closely. We have thus selected the "best" NN architectures: MPN(3), JCN(1), WNN(3) and RNN(il) and run additional experiments for 150, 250 and 350 nodes in the hidden layer(s). We have combined these results with these reported earlier in Tables 1-4 and the PNN and GRN results and summarize them in Figure 4 (since tere is only one data point for PNN and GRN their results are represented as a straight line).

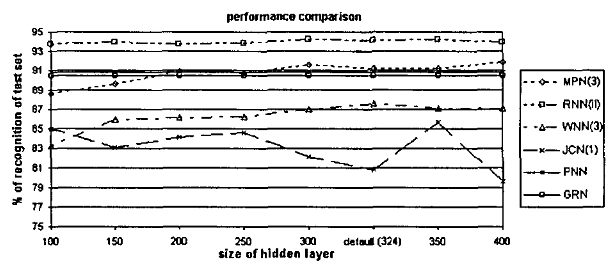


Figure 4: Effect of changing number of nodes in the hidden layer on the network performance.

We can clearly see that, overall, the RNN(il) is the most stable and most capable of these six NNs. Not far behind, however, is the MPN(3) with 400 nodes followed by the PNN and GRN. The "unstable" performance of JCN is even more pronounced with the performance, for larger number of nodes in the hidden layer, bouncing back and forth. This observation seems to suggest that one more factor which plays a role in finding an optimal architecture is the number of connections. Finally, as suggested above, we can see that the default number of nodes chosen by NeuroShell is not always the omptimal value.

5 Concluding remarks

In this paper we have reported on the results of applying seven neural networks architectures (in 15 variants) to the simplified pattern recognition problem. These 15 architectures are all the applicable architectures available in the NeuroShell package.

We have found that:

- out of all architectures the KNN and two variants of WNNs are not appropriate for the problem
- the JCNs do not perform up to par with other architectures and their performance depends very highly on the training and verification data as well as the number of nodes in the hidden layer
- the remaining architectures do a reasonably good job with the RNN seemingly the best
- the number of nodes in the hidden layer provided by the default formula depending only on the number of input nodes, output nodes and number of input patterns, seems to work well only for the WNN
- experimental data indicates that to find the optimal number of nodes the NN architecture and the number of connections in the network should also be taken into account
- as the size of training data increases the automatic calibration used in GRN and PNN is slowly reaching the level of hands-on search
- it is possible to use all of the NN architectures in the NeuroShell packge in a reasonable time (less than 30 hours of training) for a relatively large data set (both in terms of number of elements and their sizes)

We believe that these results, while not necessarily inidicative of results to be obtained for other potential problems, do indicate that experimental data needs to be taken into account when evaluating the usability of NNs for a given problem. First, due to the increasing speed and large amount of memory of modern PCs, it is now possible to apply NN directly to substantially larger problems than in the past. Therefore it is possible that preprocessing and feature extraction have to be applied only to much larger problems than in the past (and only an appropriately smaller reduction of the problem space is required). Second, NN software has considerably matured and writing one's own NN code is no longer required. Thus more work needs to be done on establishing guidelines which NN architecture should be used for which types of problems, so that those who are not NN experts will be able to use appropriate networks for their problems. To find these guidelines much more experimental work is required.

References

[1] Bowers S., Costeines A. & Paprzycki M. (1999) Applying Probabilistic Neural Networks to the Multifont

Recognition Problem with Large Training Set, in: Kumar A. N., et. al. (eds.) *Proceedings of the Twelfth International Florida AI Research Society Conference*, AAAI Press, Menlo Park, p. 336-339.

[2] Bowers S., Costeines A. & Paprzycki M. (1999) Evaluating the Performance of Three Neural Network Architectures Applied to the Pattern Recognition Problem, *Proceedings of the 15th Annual Conference on Applied Mathematics*, University of Central Oklahoma, Edmond, p. 29-36.

[3] Bowers S., Morrison J. & Paprzycki, M. (2000) Comparing Performance of Neural Networks Recognizing Machine Generated Characters, *Proceedings of the First Southern Symposium on Computing*, University of Southern Mississippi, Hattiesburg, CD, file bowers-etal.ps

[4] Bowers S. & Paprzycki M. (1999) Applying PNN's to the Multifont Pattern Recognition Problem, in: Bainov D. (ed.) *Proceedings of the 9th Colloquium on Difference Equations*, VSP, Utrecht, p. 311-318.

[5] Elman J.L. (1990) Finding structure in time. *Cognitive Science*, 14, p. 179-211.

[6] Fausett L. (1994) *Fundamentals of Neural Networks*, Prentice Hall, Upper Saddle River.

[7] FreeType library available at *http://www.freetype.org/*.

[8] Kimura M. & Nakano R. (1995) Learning dynamical systems from trajectories by continuous time recurrent neural networks, *Proc. of IEEE International Conference on Neural Networks (ICNN'95)*, p. 2992-2997.

[9] Kubat M. (1999) personal communication.

[10] Kohonen T. (1989) *Self Organization and Associative Memory*, Springer-Verlag, Berlin.

[11] Looney C.G. (1997) *Pattern Recognition Using Neural Networks*, Oxford University Press, New York.

[12] Nadaraya E.A. (1964) On estimating regression, *Theory Probab. Applic.*, 10, p. 186-90.

[13] NeuroShell 2.0 (1998), Ward Systems Inc., http://www.wardsystems.com.

[14] Paprzycki M., Niess R., Thomas J., Scardino L. & Douglass W. (2000) Comparing Performance of Neural Networks Applied to a Simplified Recognition Problem, *Proceedings of the 13th International Florida AI Research Society Conference*, AAAI Press, Menlo Park, p. 235-239.

[15] Parzen E. (1962) On Estimation of a Probability Density Function and Mode, *Ann. Math. Stat.*, 33, p. 1065-1076.

[16] Rumelhardt D.E., McClelland D.L. and the PDCP Research Group (1986) *Parallel Distributed Processing*, MIT Press, Cambridge.

[17] Specht D. (1989) Probabilistic Neural Networks for Classification, Mapping, or Associative Memory, *Proceedings IEEE Conference on Neural Networks*, vol. 1, IEEE Press, Los Alamitos, p. 525-532.

[18] Specht D. (1990) Probabilistic Neural Networks, *Neural Networks*, 3, p. 109-118.

[19] Specht D. (1990) Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification, *IEEE Transactions on Neural Networks*, 1(1), p. 111-121.

[20] Specht D. (1991) A General Regression Neural Network, *IEEE Transactions on Neural Networks*, 2, p. 568-576.

[21] Watson G.S. (1964) Smooth regression analysis *Sankhya*, Series A, 26, p. 359-72.

Artificial neural network approach to data analysis and parameter estimation in experimental spectroscopy

Mehmed M. Kantardzic, Anna Goldenberg and Troy E. Howe
CECS Department, Speed Scientific School
University of Louisville, Louisville, KY 40292

Peter Faguy
Department of Chemistry
University of Louisville, Louisville, KY 40292

Keywords: neural networks, spectroscopy

Received: January 21, 2000

The P- and S-polarized Infrared Reflection Absorption Spectra (IRRAS) can be experimentally measured and, if the optical constants for the three phases are known, the film thickness can be determined. Film thickness measurements are crucial for many surface applications. We propose the methodology of thickness measurement based on artificial neural network (ANN) technology. Using retro modeling we reduce the dimensionality of the problem, and prepare training data set for an ANN. Simulation and 3D visualization based on proposed model help us to validate the robustness of the trained ANN, and to estimate unknown parameters of instrumentation in experimental spectroscopy. Experimental IRRAS data were the inputs for an ANN testing phase, where the output is the estimated value of the film thickness.

1 Introduction

One of the most powerful tools available to scientists and engineers interested in molecular structure at metal-solution interfaces as are found in batteries, fuel cells and corroding materials is in situ Fourier transform infrared reflection absorption spectroscopy, is-FTIRRAS. The technique is in situ because the data is collected under conditions of electrochemical control. The rest of the technique's name stems from how the instrument collects the data and the nature of interaction of the infrared light with the electrode surface. The huge amount of collected experimental data, in the form of sets of spectra, are extremely complex. Often information desired from the measurement cannot be extracted from the overall response of the electrochemical system; that is, the signal of interest is buried in a sea of other signals (Fagey & Balachandar 1997, Hansen 1968).

Film thickness is an important parameter of a stratified medium of several layers, and its measurement based on experimental infrared reflection spectra is a problem for which we are proposing a solution in this paper. Although there exists an analytical model that relates film thickness parameter to the experimental IRRAS data, hand calculations are almost impossible. Two main reasons are (Fagey & Marinkovic 1995, Kantardzic et. al. 1999):

1. Imprecision of experimental data due to instrumentation errors and ambiguity in some experimental parameters. Even a minimal error in measurements will have composite effects, and analytically obtained result of a film thickness will include exponentially increased total error. For the film thickness with values in the range between 10^{-7} and $10^{-6}m$, this approach becomes impractical.
2. Complexity of the mathematical model, which relates experimental data of reflection spectra (R) and thickness of middle layer (h) even in a simple case of three-layered medium, so that explicit relation $h = f(R)$ is difficult, almost impossible to establish. The difficulty lies in the form of modeling equations. It is not possible to solve for the parameters of interest explicitly.

The immense experimental data set, ambiguity and imprecision of calculations, along with difficulties in expressing analytical relations between given inputs and required outputs explicitly, served as a basis for moving away from classical methods to an ANN approach in estimation of film thickness for a given model. The I/O transformation is schematically presented on Figure 1. However, this idea also has a fundamental problem. If we want to use ANN technology for the problem at hand we need a training data set that consists of input-output pairs. Experimental spectroscopy does not offer a technique or method to experimentally measure thickness. That means, we have inputs for an ANN, but do not have corresponding output values. Based solely on experimental data we do not have a possibility to train ANN. Therefore we had to slightly modify our approach. Although our proposed methodology is complex and consists of several phases, the basic idea could be described as follows:

1. Train an ANN with a set of data obtained from the

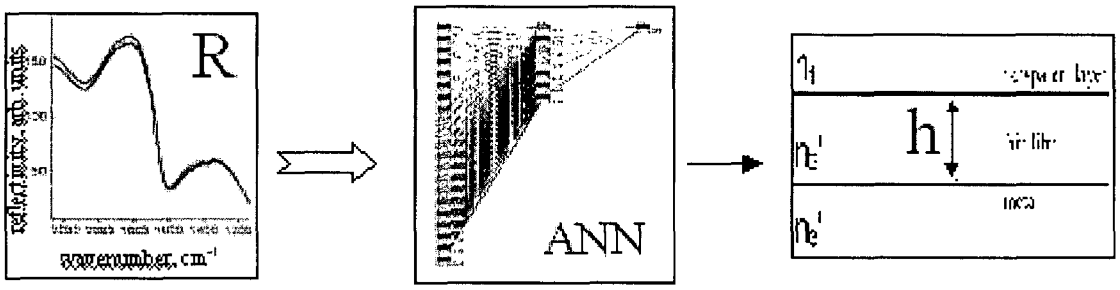


Figure 1: Film thickness measurement using ANN technology

model, which is based on Fresnel's equations.

2. Use experimental data to validate and test the trained network, where the output of the trained network will estimate the film thickness.

Proposed methodology requires careful use of experimental and model-derived data as a combination in some phases. This paper describes methods, tools, and techniques used to prepare for ANN-based methodology, where the actual neural network is the final step in automation of a process of estimation of a film thickness parameter based on experimental infrared reflection spectra. To prepare the data for an ANN training process, we will at first concentrate our efforts on modeling of infrared reflection spectra for a three-layered medium.

2 Infrared spectroelectrochemical model

Modeling the interaction of electromagnetic radiation with a stratified medium of several layers, like the electrode-solution interface, is possible if certain parameters of the system are known. Such a mathematical model completely describes the reflection, refraction and attenuation of light in a stratified media. There are studies in literature, that construct hypothetical data as if measured using is-FTIRRAS (Fagey & Fawcett 1995). What has not been done to date is the use of the experimental data from an in situ measurement to derive some of the parameters used in the modeling process, i.e. a retro-modeling process. The reflection, refraction and attenuation of light in a stratified media are completely described by Fresnel's equations in terms of a transverse electromagnetic wave with an electric field strength component parallel to the plane of incidence, P-polarized light, and perpendicular to the plane of incidence, S-polarized light. The system consists of a thin weakly absorbing layer that is presented between two semi-infinite layers, where one is a conductor and the other is in a transparent phase with a low refractive index. A model for a system is shown in Figure 2(Fagey & Marinkovic 1995, Kantardzic et. al. 1999). Analytical expressions for the model are functions of the wavenumber,ν, the angle of incidence, θ, the linear polarization state, P or S, the

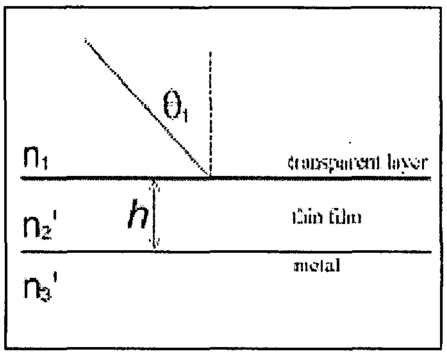


Figure 2: Schematic diagram of a three layer interfacial model for a lubricant film

film thickness, h, and the optical constants of the media, n' (Fagey & Marinkovic 1995, Hansen. 1968). Note that several variables in these equations are complex values, reflecting absorbing nature of the sample:

$$\theta_m = \arcsin(\frac{n'_{m-1}}{n'_m}\sin(\theta_{m-1})) \quad (1)$$

$$\beta = 2\pi\frac{h}{\lambda}n'_2\cos(\theta_2) \quad (2)$$

$$\mathbf{r}_\perp = \frac{\mathbf{r}_{\perp 12} + \mathbf{r}_{\perp 23}e^{2i\beta}}{1 + \mathbf{r}_{\perp 12}\cdot\mathbf{r}_{\perp 23}e^{2i\beta}} \quad (3)$$

$$\mathbf{r}_\parallel = \frac{\mathbf{r}_{\parallel 12} + \mathbf{r}_{\parallel 23}e^{2i\beta}}{1 + \mathbf{r}_{\parallel 12}\cdot\mathbf{r}_{\parallel 23}e^{2i\beta}} \quad (4)$$

$$R_\parallel = |\mathbf{r}_\parallel|^2 \quad (5)$$

The optical constants for different layers are actually not real constants but complex quantities in a domain of wavenumbers (ν) given by:

$$n'(\nu) = n(\nu) + ik(\nu) \quad (6)$$

where n is the refractive index and k is the absorptive index. The P- and S-polarized IRRA spectra can be analytically determined based on other known parameters of a model. In order to establish a model, optical constants $n'(\nu)$, with their real and imaginary frequency-dependent parts are determined experimentally, as results of previous

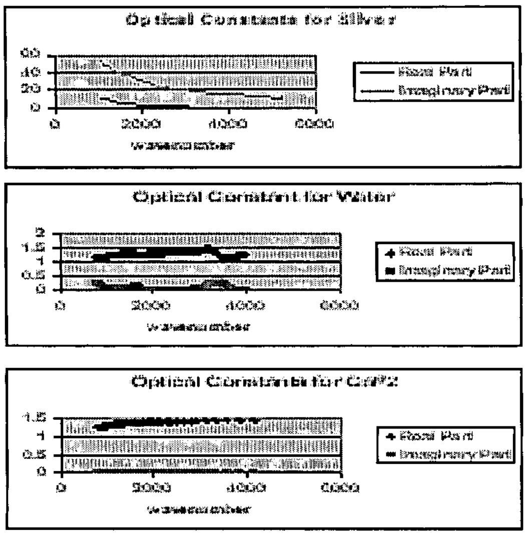


Figure 3: Interpolated functions for optical constants for: Ag, H_2O,and CaF_2

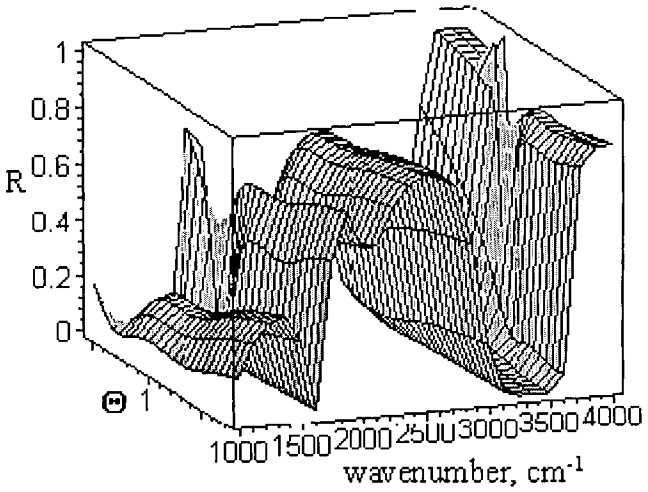


Figure 4: Model of Fresnel's equations for film thickness $h = 5 \times 10^{-6}m$

research (Fagey & Marinkovic 1995). We used these constants for our retro-modeling and 3D visualization of the model.

3 Dimensionality reduction based on visualization of Fresnel's equations

Vast amounts of discrete data generated by computer-based retro-model of Fresnel's equations, and large data sets, which are collected by IRRAS experimentation, make the problem of analysis and interpretation difficult. Is there a possibility to reduce the amount of data, both from experiments and discrete model, preserving information about relations between parameters of the process? That is especially important if we want to use artificial neural network technology for estimation of the middle layer thickness based on other parameters determined experimentally or through the model (wavenumber ν, the angle of incidence θ, the linear polarization state P or S, and the optical constants of the media Sn'S). To reduce the amount of data for further analysis we used 2D and 3D visualization techniques. The purpose of infrared spectroelectrochemical model visualization is to use theoretical calculations based on Fresnel's equations that completely describe the reflection, refraction and attenuation of light in a stratified media, to identify optimal configuration for theoretical and experimental determining of the film thickness. Optimization in this context means reduction of necessary input while maintaining enough sensitivity to predict output value of the film thickness with satisfying precision (Kantardzic et. al. 1999, Marefat et. al. 1997).

To perform these analyses and make decisions about most informative parameters and their ranges, in the first phase of a proposed methodology we used retro modeling of Fresnel's equations and corresponding visualization techniques for a three-layered medium: CaF_2 / H_2O / Ag. Based on experimental discrete values of optical constants given in a wavenumber domain for all three layers: CaF_2, H_2O, and Ag, we made linear functional interpolation through these values to get a continuous function necessary for further modeling. Optical constant functions are given on Figure 3.

Using Maple V Release 5.1 as a software tool, we created 3D model and corresponding visualizations of Fresnel's equations for the given materials in three layers. One dimension in the model was wavenumber ν, second was the angle of incidence θ, and the third, dependent variable of the model represented the real amplitude of Fresnel's coefficient, which led to the reflectance R. It is important that the same quantities can be experimentally measured, so that theoretical and experimental values can be compared. For every computation the film thickness h was a constant value. Example of one 3D representation of the model is given on Figure 4.

To select the most sensitive region of the model, with respect to h, we analyzed continuous visualizations of the model for different h values (Kantardzic et al. 1999). As an example we presented on Figure 5 three graphs for different h values, $h = 0.1$, 5 and $10 \times 10^{-6}m$. Based on graphical representation of data we analyzed the differences ΔR between continuous infrared reflection spectra for given h values. If we compare the results for $h = 0.1 \times 10^{-6}m$ and other h values, we can see significant differences ΔR in entire domain, for all ν and θ values. But comparison of ΔR for $h = 5 \times 10^{-6}m$ and $h = 10 \times 10^{-6}m$ gives another conclusion. ΔR differences, which we accepted as a measure of sensitivity of our model with respect to film thickness value h, are higher for ν values less than 3000,

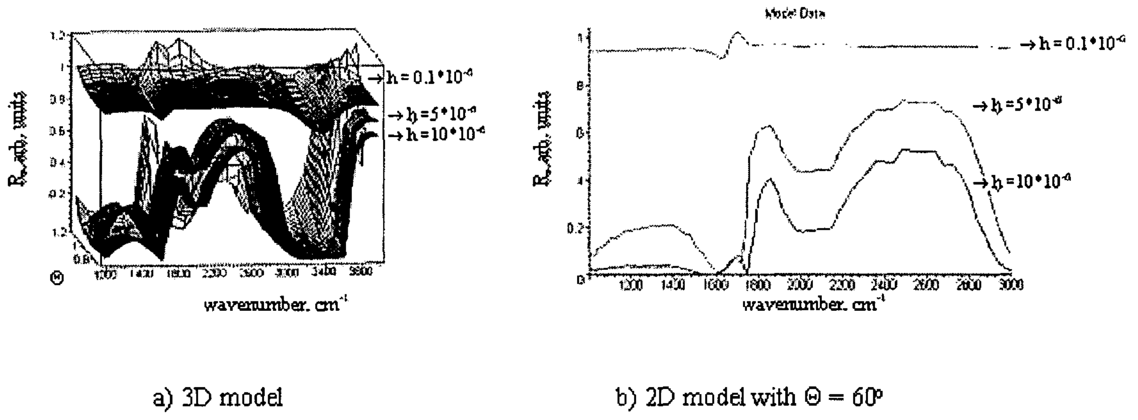


Figure 5: 3D and 2D models of three surfaces for different h values: $h = 0.1, 5,$ and 10μ.

$\theta\backslash\nu$	1200	2000	2600	3000
40°	.219974056	.2338258560	.1267890738	.1629288071
50°	.1682207130	.2212274364	.145693822854	.1316719846
60°	.1523269448	.2531187111	.21239446928	.061906043
70°	.0685264388	.1342928253	.02510797692	.0685173970

Table 1: ΔR (for $h = 5 \times 10^{-6}m$ and $10 \times 10^{-6}m$) values as a measure of sensitivity for different θ and ν values.

and sensitivity is smaller for larger ν values. Also, highest level of sensitivity is for angles θ about 60°. A preliminary analysis is supported by numerical data given in Table 1. An important conclusion from the visualization of the model is that the model is highly sensitive on film thickness changes for wavenumbers ranging between 1000 and 2800. Maximum sensitivity at the same time is for input angle of about 60°. These conclusions are the bases for selection of input parameters for artificial neural network training. Without this analysis a total number of input values for ANN training is discrete 3D surface with 480 values (40 discrete values for θ dimension times 12 discrete values for θ dimension). Selection of the most sensitive 2D curve on a given surface reduces the number of inputs to 21. The only constraint is on the set of experimental data that has to be collected. The ANN will be trained with fixed θ value of 60°, so the experiments should have the same value for the angle of incidence.

4 ANN training

To evaluate the applicability of ANN technology for prediction of a film thickness based on experimental spectra, we prepared the data for ANN training and testing for a three-layered media: CaF_2/H_2O / Ag. Feature extraction and data selection processes, described with more details in previous section, defined 21 discrete input values of R for fixed, equidistant values of wavenumbers ν. Because the modeling process generates continuous 3D representation of R spectra, additional discretization was necessary (Dubitzky et al. 1999, Kantardzic 1999). Several discrete spectra of reflectivity (R), for different film thickness values h, were generated from the model and some of them are presented on Figure 6.

Neural network implementation, as a part of our methodology, is based on Stuttgart Neural Network System (SNNS) software package for HP workstations under UNIX operating system. The training process was carried out using standard feedforward/back-propagation neural network with one hidden layer. The commonly used sigmoid function was selected as the transfer function, and the learning coefficient rate was set to 0.2. The topology of the artificial neural network was determined by 21 input units representing spectra features, 21 processing units on a hidden layer (the number was selected after several experiments), and one output value representing estimated film thickness parameter.
The number of neurons on the hidden layer was chosen after a series of experiments. The hidden layers of 6, 7, 8 and 21 neurons were tried in turn in search for the best network performance. It was determined experimentally, that the network has performed satisfactory having 7 neurons on the hidden layer, however it performed the best having 21 neurons on the hidden layer. The two types of ANN architecture are shown below.

The network was trained with 100 input-output patterns. Every pattern (22 dimensional vector) was selected from the model to represent different output (film thickness) values in a range between 10^{-7} and $10^{-5}m$. Equidistant strategy is used again, with a step of 10^{-7} m in output values in two consecutive patterns. The training process required 2000 cycles of back-propagation algorithm to obtain satisfactory mean square error (MSE < 0.001). Dy-

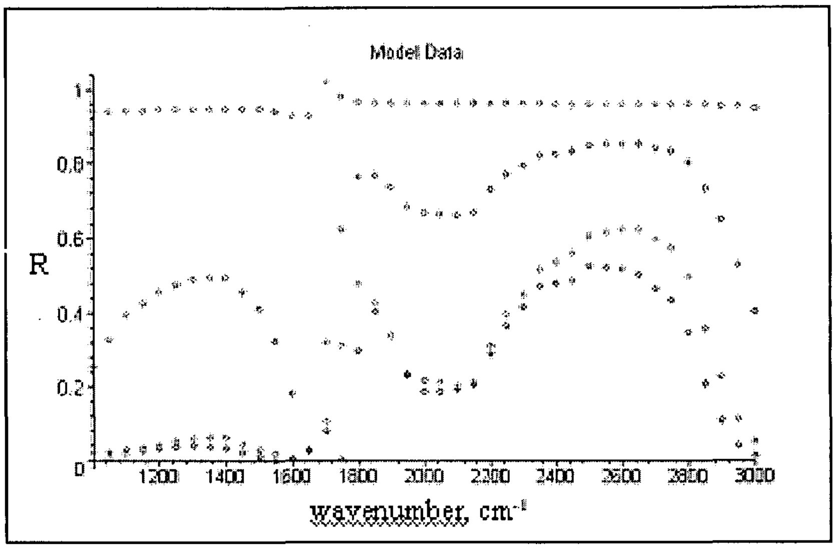


Figure 6: Input-output patterns for ANN training

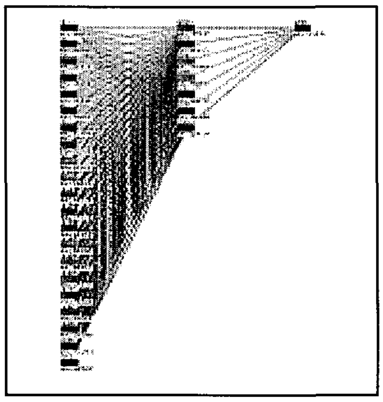
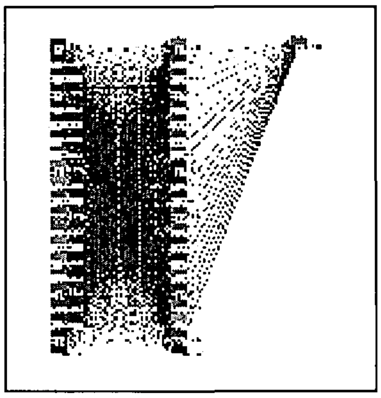
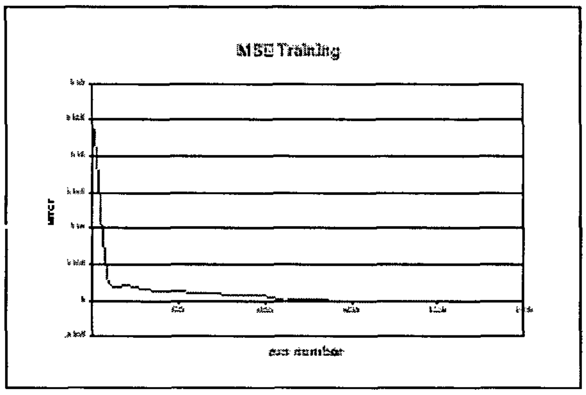



Figure 7: Two experimental ANN architectures: 7 and 21 neurons on the hidden layer

namic changes in MSE and training parameters are shown on Figure 8. One of the most important advantages of neural network technology that we emphasize in our methodology, is its ability of fault tolerance, meaning that partial destruction of the network connections or even more important imprecision in input data, does not lead to corresponding degradation of ANN performances and quality of results. For our experiments it is important to prove that small changes in input data, caused by error in instrumentation or imprecision in measurements, will not have cumulative and destructive effects on a final, predicted value of an ANN output h (Dubitzky et al. 1999).

To validate this hypothesis we could not use experimental data because we do not have corresponding outputs for the given, measured inputs. Therefore, the only possible approach found was based on simulations using established model. We started with an assumption that maximal errors in measurements are on a level of 10%. Using this constraint we generated artificial, simulated changes in model data using random number generator. These changes were maximum of $\pm 10\%$ for: a) a randomly selected point in each of the input R spectra pattern where noise does not have a zero mean, and b) all points in the spectra where noise has zero mean. They simulated noisy experimental values of spectra for a given output value of film thickness. Validation of trained ANN was made for all 100 sets of simulated spectra with 10% error). Testing of the ANN showed significant error reduction in outputs. Graphical representation of all output based on model data and simulated, with additional error generated input spectra, is given on Figure 9. Additional analysis shows that mean square error at the ANN computed output is reduced to the level of 5.21×10^{-6}. These results supported our hypothesis about ANN robustness. Using ANN technology error level in estimation of film thickness as output is reduced compared to the level of error based on experimental R spectra.



Training parameters:

Backpropagation Training Algorithm

cycles	: 2000
λ training constant	: 0.20000
#o-units	: 1
#patterns	: 100

Figure 8: Mean Square Error (MSE) and training parameters

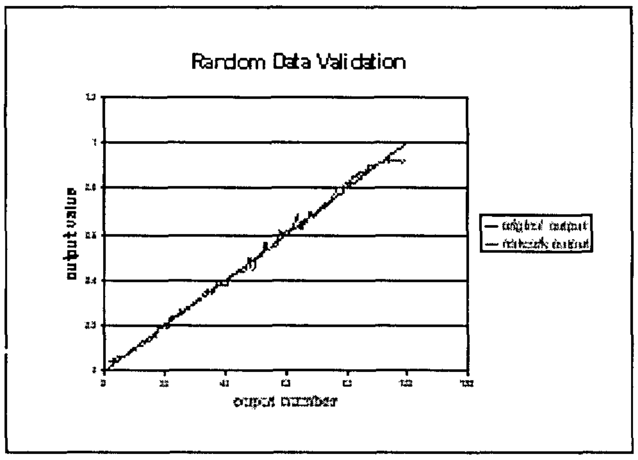


Figure 9: Comparison of ANN outputs of the model vs. simulated experimental outputs

5 ANN testing using experimental spectroscopic data

Before final testing of our trained ANN with experimental spectroscopic data we had to solve one additional problem. We had to determine the value of one additional parameter, so called instrumentation factor C. Experimentally measured IRRAS spectra always gives an output in a form of relative emissivity: $\Delta R/R$. That means the spectra values are multiplied by a scaling factor that is not given in advance (as an instrumentation parameter), and is not possible to be determined experimentally. Therefore typical spectra obtained from the model and experiments show large visible differences as it is given on Figure 10. These significant differences are caused by unknown scaling factor C, where $(\Delta R/R)_{experimentally} = C(R)_{theoretically}$.

Although the factor C is by its nature nonlinear on a wavelength ν domain, we accepted the approximation of its constant value for ν between 1000 and 2800. To determine scaling factor C we developed additional procedure. The basic idea is to compare the shapes, not domains of

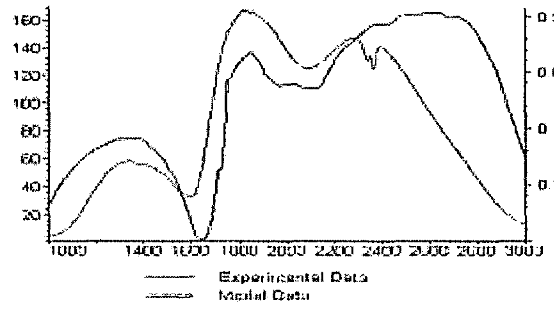


Figure 10: Differences in spectra values for model data and experimental data

experimental and model spectra curves. The shape of the normalized curve for experimental data is compared with shapes of all model curves. Based on minimal square error measure (MSE) the closest model curve is selected, and then corresponding instrumentation factor C is determined. The main steps of the procedure are:

1. Select (or correct previous) upper bound of the normalization range for experimental data.

2. Compute the new discrete values for normalized experimental spectra using standard formulae:

$$\bar{R} = \frac{R - R_{min}}{R_{max} - R_{min}} R_u. \qquad (7)$$

where $\bar{R}$ is the normalized value, R_u is the upper bound of the range, R_{min}=4.05394 and R_{max}=167.578 are given values for an experimental spectra.

3. For a given experimental spectra $X(\nu)$ normalized in step 2, the trained ANN make a computation of estimated film thickness value h.

4. For the estimated value h reconstruct the corresponding model spectra $Z(\nu)$.

5. Compare the differences between spectra $X(\nu)$ and $Z(\nu)$ but only for the ν domain between 1600 and

2600 (where C factor suppose to be linear). If they are close enough (mean square error MSE is minimal) stop the procedure with a final value of scaling factor C:

$$C = \frac{R_u}{R_{max} - R_{min}} \tag{8}$$

If the differences are still large, repeat steps 1-5 in an additional iteration with a new, corrected normalization range value.

We repeated previous procedure several times applying different normalization ranges for experimental data. Graphical representation of functions $X(\nu)$ and $Z(\nu)$ with corresponding h and MSE values is given on Figure 11 for three selected iterations of our procedure.

Selected (optimal) value for scaling factor C is: $C = \frac{1}{167.578-4.05394} = 0.0060781$ which corresponds to the scaling range of the experimental spectra [0, 1]. Final results of application of a pattern recognition technique for three layered media: CaF / H_2O / Ag using experimental data showed the best fitting of spectra curves $X(\nu)$ and $Z(\nu)$ for $C = 0.0060781$ value, because of minimal mean square error (MSE = .02194). It is visible in Figure 11 that these spectra are closer for optimal C value than for others (experiments with ranges [0, 0.8] and [0, 1.2]). We selected this value as a linear scaling factor of experimental data and used this factor to prepare testing data for ANN. To validate this factor we tested trained ANN with additional, new set of data obtained from the same class of experiments. Namely, using experimental spectroscopy technology we collected several sets of measurements for the same configuration of three-layered interface. Additional patterns were tested, and in all cases the final outputs of ANN, which are estimated h values, were the same (or close) to value $h = 1.42m$. That was a final step in our methodology.

We created software environment, which accepts as an input experimental spectroscopic data, and produce quantitative estimation of middle layer thickness as an output. Through several interactive steps these software tools transform raw input data into corresponding ANN inputs, and finally trained ANN based on these inputs estimate output h value. At this moment we do not have a possibility to establish independent validation for our methodology and obtained results because it does not exist any other technique for determining film thickness based on experimental data. Our methodology will be tested on large database of experimental measurements of spectra, and corresponding results will be analyzed and evaluated through the possible applications of obtained film thickness values.

6 Conclusions

In this paper we described a new software tool and a corresponding methodology for estimating a film thickness in a three-layered interface based on measurements of spectra obtained from infrared experimental spectroscopy. To develop the methodology we used different modeling, simulation, visualization, numerical approximation, interpolation, and pattern recognition techniques and tools. Different steps in proposed methodology helped us to understand better the behavior of spectra and estimate some parameters in experimental spectroscopy measurements such as sensitivity of refraction spectra and scaling factor of spectroscopic instrumentation. Proposed methodology is tested with experimental data where the computed output from the trained ANN represents estimation of unknown, film thickness value (h). Applied to a specific problem in this case, this approach can also be used for a broader range of problems. Any problem where the forward model is well known and understood, but the inverse is analytically intractable, the described techniques and developed software will become valuable tools. Some modifications connected with a new domain of application would be required.

In the next phases of our research we have to concentrate our efforts in two directions. First, using developed methodology we will create trained ANNs for different three-layered interface structures. In that case the methodology could be applicable for large amount of experimental data currently available. As a final result we expect development of integrated programming environment, which will be supported with large data base of analytical and ANN models for different media, with its optical constants and other necessary parameters. This package will give the user possibilities to analyze the interface through visualization and other computer based techniques. Also, based on experimental data the package will compute an electrolyte thickness in infrared spectroscopy with satisfactory precision. Second direction of our research will make an emphasis on additional experimentation to obtain better tuning of different parameter's approximations used in proposed methodology. Linearization and constraints on optical constant functions (n), instrument scaling function (C), wavelength domains (ν), and angles of incidence (θ) are only some of the parameters which require deeper analysis to obtain better, more precise results from our methodology.

7 Acknowledgement

This work was supported in part by IRIG-URG #628345.

References

[1] Chen J. X., Rine D.& Simon H. D. (1996), Advancing Interactive Visualization and Computational Steering,*IEEE Computational Science & Engineering*, Winter, pp. 13-17.

[2] Dubitzky W., Mariotti D. & Hyland M. (1999), Analyzing Plasma Ammision Spectra Using Neural Networks, *IJCA*, Vol. 6, No. 2, June 1999, pp. 88-93.

[3] Fagey P. W. & Balachandar N. (1997), Application of Two-dimensional Covariance Analysis of Infrared Spec-

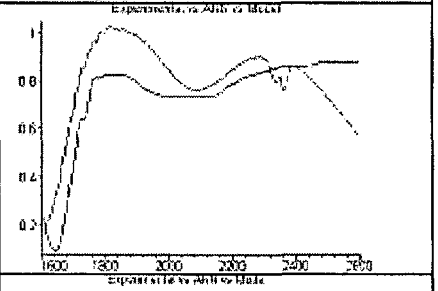
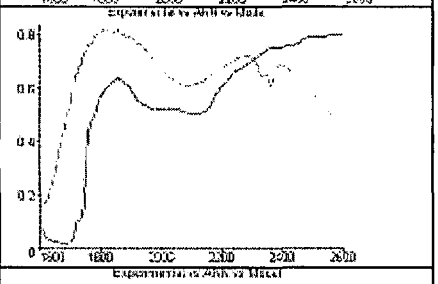
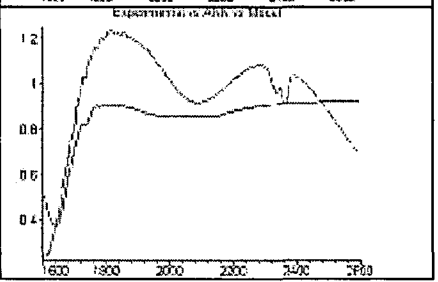
The Range after Normalization	**Thickness h**	**MSE**	**Graphs** Lighter gray – Experimental Data Darker gray/black – ANN Based Model
[0, 1]	$1.42 * 10^6$	.0219420977620916	
[0, 0.8]	$3.18 * 10^6$	.0509597236915296	
[0, 1.2]	$0.78 * 10^6$	.0310241554176692	

Figure 11: Evaluating normalization of experimental data

troscopic Studies of Electrolyte Solutions, *Analytical Communications*, Vol. 54, December 1997, pp. 405-408.

[4] Fagey P. W. & Fawcett R. W.(1990), Infrared Reflection-Absorption Spectroscopy on the Electrode/Electrolyte Solution Interface: Optical Consideration, *Applied Spectroscopy*, Vol. 44, No. 8, 1990, pp. 1309-1315.

[5] Fagey P. W. & Marinkovic N. S., Sensitivity and Reproducibility in Infrared Spectroscopic Measurements at Single-Crystal Electrode Surfaces, *Anal. Chem.*, Vol. 67, 1995, pp. 2791-2799.

[6] Hansen W. N. (1968), Electric Fields Produced by the Propagation of Plane Coherent Electromagnetic Radiation in a Stratified Medium, *Journal of the Optical Society of America*, Vol. 58, No. 3, March 1968, pp. 380-390.

[7] Kantardzic M. M., Goldenberg A. & Fagey P.W.(1999), Dimensionality Reduction in Experimental Spectroscopy using Visualization of Model-Derived Optical Parameters, *ISCA Proceedings of the 8th International Conference on Intelligent Systems*, Denver, CO, June 24-26, 1999, pp. 37-40.

[8] Marefat M. M., Varecka A. F., Yost J.(1997), An Intelligent Visualization Agent for Simulation-Based Decision Support, *IEEE Computational Science & Engineering*, July-September 1997, pp. 72-82.

Control of a one-legged three-dimensional hopping movement system with multi-layer-perceptron neural networks

Klaus D. Maier[1,3], Volkmar Glauche[2], Reinhard Blickhan[1], and Clemens Beckstein[2]
[1]Friedrich-Schiller-University Jena, Institute of Sports Science,
Biomechanics Group, D-07740 Jena, Germany
[2]Friedrich-Schiller-University Jena, Institute of Computer Science,
Artificial Intelligence Group, D-07740 Jena, Germany
[3]K.D. Maier is now with Infineon Technologies AG,
CMD CE, PO Box 800949, D-81609 Munich, Germany, Email: k.maier@ieee.org

Keywords: multi-layer perceptron, robot movement control, hopping movement system control

Received: January 10, 2000

Controlling the model of a movement system based on the dynamics of biological hopping and running is investigated. This movement system consists merely of a massless spring attached to a point mass. It is describing fast three-dimensional legged locomotion on even grounds. Rapidly moving legged autonomous systems require different hardware layouts and control approaches in contrast to slow moving ones. The spring mass system is a model that describes this principle movement as well as the principle control task. Multi-layer-perceptrons (MLPs) were used to implement neurocontrollers suitable for such a movement system. They prove to be suitable for exact control of the movement with a relatively small number of neurons. This is also shown by an experiment where the environment of the spring-mass system has been changed from even to uneven ground. The neurocontroller is performing well with this additional complexity without being trained for it.

1 Introduction

Control of movement is an issue for robotics and autonomous systems. For movement over ground, wheels as well as legs can be used. Most artificial movement systems use wheels while natural movement systems employ legged locomotion. Wheeled systems have some limitations like limited mobility or immobility on irregular ground surfaces. For this reason only about 50 percent of the worlds landmass can be entered by wheeled systems [1]. Legged systems have potential to overcome such limits: only discrete surface points are needed for leg placement, not continuos surfaces. The movement of the body is decoupled from the movement of the feet: This is a form of active suspension. For legged systems high movement velocities are desirable. To achieve this ballistic movement, i.e. flight phases are necessary. This results in landing impacts introduced into the systems. Legged systems must be able to cope with such impacts. This could be achieved by dimensioning bearings in a way that impact stress can be managed. Such a design would result in relatively heavy and large bearings. Also repeatedly introduced impact shocks reduce stand times.

Results from research on natural fast moving systems indicate that visco-elastic elements are used to effectively handle impact energy [2]. These are able to store and absorb impact energy. Using such elastic elements is an intelligent principle of construction that can be transferred to technical systems. Unfortunately elastic elements combined with ballistic movement phases make systems nonlinear and hard to control. This requires high computational efforts which complicates real time control. We therefore investigated in the suitability of multi-layer perceptrons (MLPs) [3] for implementing the control of dynamic movement systems. Characteristic properties of neural networks like fault tolerance, robustness and especially the ability to generalize make them a good choice for controller designs. They can be trained using available learning algorithms. Once trained they offer fast response. This allows to build control structures that provide online control which is essential for fast locomotion.

Using neural networks for the control of legged movement systems has been addressed by Beer et al. [8] for hexapod autonomous systems as well as by Berns et al. [9] for a six-legged robot modelled after the stick insect. Other groups are also working on control of legged systems with little or no elastic elements included. These systems are moving only slowly due to this construction principle. They control leg trajectories – when using elastic elements the trajectory is determined by material laws and only initial conditions are set by the controller. Exploiting ballistic movement for locomotion in the walking robots mentioned above is not desired nor possible. It would lead to very complex control problems. The approaches that demand static stability are another reason for the slow movement.

Simple dynamic movement has already been under investigation in many projects mostly in the form of the pole-balancing control task (e.g. like Anderson [1], Ritter [10], Widrow [12]). There are no applications for the design of walking machines though. Control of a dynamic movement system like the 3D-Hopper [5] has been partly addressed by Atkeson et al. [13] using neural networks to improve the original look-up table control.

We have investigated MLPs for the control of two dimensional spring mass systems [14]. They have proven to be successful for effective and exact control. Our investigation in Radial-Basis-Function Neural Networks for the control of such two dimensional spring-mass movement system resulted in the design of controllers of comparable performance but at the price of higher computational cost [15]. Experiments with controllers based on Self-Organising Motoric Maps (Ritter et al. [10]) have yielded unsatisfactory results [16].

Alternative approaches for control of hopping that are not using neural networks can be found with the hopping machines of Raibert were look up table implementation of polynominal fits of the systems movement equations were used and methods based on fuzzy logic as presented by Kluge[17].

In the following chapter the movement system model to be controlled is introduced. The structure of the used neurocontroller is described in chapter III, which is also addressing the training patterns for the neural networks of the neurocontroller and the respective network architecture of the used multi-layer perceptron neural networks. Then the results achieved with the neurocontroller on the example of several test scenarios are presented. The paper is completed by a conclusion and summary.

2 The movement system model

The movement system to be controlled is derived from a model that describes human and animal running and hopping [4]. It consists of a mass-less spring attached to a point mass that is jumping across even ground in three-dimensional space (Figure 1). The mass represents the body of the system model and the spring its single leg. The motion of this system model is divided into two phases: The 'flight phase' where the mass' trajectory is determined by gravity only and the 'ground phase' where the movement is determined by the properties of the spring as well as gravity. The system model is passive, i.e. mass, stiffness and length of the uncompressed spring remain constant. These constants where set to physiological relevant values (Table 1) taken from [4].

Table 1: System constants

Constant	m	k	l
Value	60 kg	10 kN/m	1 m

The equations of motion during the ground contact phase are:

$$\ddot{x} = x\frac{k}{m}\left(\frac{l}{\sqrt{x^2+y^2+z^2}}-1\right) \tag{1}$$

$$\ddot{y} = y\frac{k}{m}\left(\frac{l}{\sqrt{x^2+y^2+z^2}}-1\right) \tag{2}$$

$$\ddot{z} = z\frac{k}{m}\left(\frac{l}{\sqrt{x^2+y^2+z^2}}-1\right)-g \tag{3}$$

where x, y: horizontal deflections, z: vertical deflection of the spring, g: gravitational acceleration, k: spring stiffness, m: mass and l: uncompressed length of the spring.

During the flight phase the equations of motion become:

$$\ddot{x} = 0 \tag{4}$$

$$\ddot{y} = 0 \tag{5}$$

$$\ddot{z} = -g \tag{6}$$

This dynamic system model can be kept in motion (that is preventing ground contact of the mass) by active control only. The system model has characteristic properties similar to hopping machines developed by Raibert [5]. It allows to reduce some of the complexity of the real world system by maintaining the principle of locomotion and the control task. Scaling from the system model to a real world system should therefore be straightforward.

3 The neurocontroller

A feedback structure is used to implement the neurocontroller. The multi-layer perceptron has two classes of input signals: control parameters from a higher center (like desired speed or directions) and sensed information on the observable system states. Output signals are motor control data to drive the controllable states of the movement system. The movement system is situated in its environment (the world) and is experiencing the immediate feedback of the environment (situatedness and embodiment [6]).

3.1 Fixed and changing coordinates

The number of inputs parameters to the controller can be reduced by using a designated Cartesian coordinate system (a,b,c) for each ground contact phase (Figure 2). The direction of the a-axis is set so that it coincides with the direction of the horizontal velocity $(\dot{x},\dot{y})$ during the preceding flight phase. The origin of the (a,b,c) coordinate system is placed in the projection of the mass onto the (x,y)-plane at touchdown. With respect to this coordinate system the horizontal velocity $\dot{b}_{td}$ is always zero. For touch down the coordinates become

$$\dot{a}_{td} = \sqrt{(\dot{x}_{td})^2 + (\dot{y}_{td})^2} \tag{7}$$

$$\dot{b}_{td} = 0 \tag{8}$$

$$\dot{c}_{td} = \dot{z}_{td} \tag{9}$$

expressed in (x, y, z)-coordinates.

The variables perceived are the legs' angle of attack $(\Delta a_{to}, \Delta b_{to})$ as well as the velocities $(\dot{a}_{to}, \dot{c}_{to})$ of the mass at the point where the spring lifts off the ground, i.e. the transitions between ground phase and flight phase. The parameter to be controlled is the angle of attack expressed as $(\Delta a_{td}, \Delta b_{td})$ at 'touch down' that has to be set in a way that the desired horizontal velocity $(\dot{a}_d, \dot{b}_d)$ for the flight phase is reached (Figure 3).

The control task can be solved using numerical methods. A supervised learning approach is therefore appropriate to train the MLP. Training patterns for the control of the movement system have to be generated.

3.2 Generating training patterns

The spring-mass systems' equations of motion (1)-(6) have been solved numerically for representative values covering the whole parameter space using a fifth order Runge-Kutta-Gill method. The center of gravity trajectories have been numerically determined for all combinations of parameter values at touch down. Polar coordinates were used to solve the equations of motion of the spring mass system (Table 2). This was done in order to achieve an even spread of initial states. φ_{td} is the direction of the spring at the point of touch down in the x-y-plane, ϑ_{td} is the angle of attack of the leg with respect to the x-y-plane and ($\dot{a}_{td}$, $\dot{b}_{td}$, $\dot{c}_{td}$) are the velocity components at touch down.

Table 2: Initial states for numerical solution

Para-meter	φ_{td}	ϑ_{td}	$\dot{a}_{td}$	$\dot{b}_{td}$	$\dot{c}_{td}$
Range	$0 \le \varphi_{td} < 2\pi$	$-\pi/2 \le \vartheta_{td} \le \pi/2$	$-9 \le \dot{a}_{td} \le 9$	0	$0 \le \dot{c}_{td} \le 4$
In steps of	π/90	π/90	1m/s		1m/s

The initial and final states of a ground phase have to guarantee that the hopping height before landing and after take-off will be sufficient to move the leg into its new position during flight phase without hitting the ground. Data from center of gravity trajectories meeting these conditions have been used to compute the data patterns for the training set - based on the initial state, possible final states of the preceding ground phase were included as well as patterns for negative horizontal velocities. This lead to overall 40,000 training patterns that were also used for 'in-training verification'.

To ensure good control of the system MLPs have to be trained with patterns that are representative for the movement. For the spring-mass system, training patterns have to cover the whole variety of possible movement patterns - hopping in place, acceleration, hopping at constant speed, deceleration and change of direction. If these characteristic movements are not sufficiently represented by the training patterns, the trained MLPs will not be able to control the movement of the system: the desired horizontal velocity will not be reached. This may lead to undesired behavior of the system (e.g. unnecessary change of direction, too fast or too slow movement, falling down). Inadequate selection of training patterns has not only an impact on the control abilities of trained MLPs, but also on the number of training cycles needed to learn the patterns. We selected representative data that covered the desired range of horizontal velocities. This is underlined by simulations that were made with desired horizontal velocities much higher than those used during training. In this case - as expected - the MLPs were not able to control the movement of the system. In order to reduce the number of training patterns, MLPs were trained only with patterns for selected horizontal velocities at touch down as shown in Table 2.

3.3 MLP–design

We have examined several different MLP architectures with one or more hidden layers using sigmoid activation functions [3]. Each hidden layer consisted of between 5 and 10 neurons. The learning algorithm used was back-propagation with momentum [7]. After reaching the error goal, the MLPs were tested on the quality of generalization using the samples that were not used for training. The movement system was left running with randomly changing velocities for more than 10,000 steps to ensure stability of the controller.

One layer of hidden neurons was not sufficient to accomplish stable control of the spring mass system. All other investigated networks with two layers of hidden neurons were able to control the spring-mass-system in a way that the actual velocity followed the desired horizontal velocity closely: differences between desired and actual velocity when hopping at constant speed was dropping within a few steps to values below 0.3m/s and below 0.1m/s for the best performing MLP with 7 neurons in the first and 9 neurons in the second hidden layer. Comparing these values to results reached using the polynominal fits method of Raibert and the fuzzy logic based method by Kluge shows that in our approach significantly less computational power is needed to achieve similar or superior accuracy of control.

4 Results

For evaluating the trained networks test scenarios were developed. The neurocontroller has been tested under the following scenarios to gain information on it's

capabilities: hopping in place, hopping a spiral, hopping an eight and hopping a square.
In order to further evaluate the quality of the control provided by the neurocontroller we tested our trained MLP in an environment with uneven ground. We introduced a stepwise changing surface describing the surface height at the point of touchdown. The surface height is changed by adding normal distributed random height differences with a standard deviation of half the leg length. This results in slowly changing slope like surfaces that are interrupted by sudden 'bump-like' changes in height. For simulation purposes this stepwise changing surface has the same effect as a continuously changing ground, since only the height of the ground at the position of touch-down is relevant for the behavior. Still, the leg has to be brought forward into the direction of the touch down point. The system is of course limited in its movement by energy constraints. It has no means to change its total energy during hopping. Therefore it is not able to hop over unconstrained rough ground.

4.1.1 Hopping in place

Hopping in place is a difficult task compared to hopping at high speeds. Even small deviations from the vertical position of the spring result in a substantial horizontal component of the take-off velocity. For the typical human parameters of mass and spring constant a deviation of 1° can result in horizontal velocities or more than 0.5 m/s. Ideal hopping in place is therefore not achievable in practice.
The neurocontroller is able to keep the system close to the place the system should ideally be for hopping in place. Figure 4a and Figure 4b show the behavior of the system using a MLP with 7 neurons in the first and 9 neurons in the second hidden layer for hopping on even ground. The cyclic movement of the spring mass system around a stationary point is due to a small constant deviation in control output from the ideal output. This constant deviation is not related to floating point effects and the cyclic effect originates from to the turning of the (a,b,c) coordinate system in the way that the b component is set to zero.

4.1.2 Spiral

A combination of change of direction as well as change of speed hopping along a spiral shaped course on even ground has been chosen. Figure 5a shows that the controller is able to direct the movement along the desired velocity course and Figure 5b shows the spiral movement plotted over the x-y-plane.

4.1.3 Eight

Another complex movement task can be found in Figure 6a. The movement system is controlled to follow the shape of the number eight by hopping over uneven ground. The figure shows that the controller is clearly able to make the system follow the prescribed velocity curve. The influence of the uneven ground is disturbing the movement of the system but it is still capable of re-adjusting the movement of the spring mass system. Figure 6b shows the movement plotted over the x-y-plane. Even with the disturbances from the uneven ground, the figure eight can be clearly identified. The controller is performing well enough with its control of the time-velocity course, that its derivation, the time-place course remains in the desired shape.

4.1.4 Square

A movement course following a squared figure is giving information on the capability of the controller to keep certain directions. Figure 7a shows the time velocity course for several square figure across uneven ground. The disturbances to the movement introduced through the changes in the ground height can be clearly perceived. Changes from the prescribed movement course are regulated by the MLP to the desired values. The movement over the x-y-plane is shown in Figure 7b. The square figures are visible. Deviations from the ideal figure are due to the influence of the uneven ground. The system can not sense these deviations. This would be a task for a higher hierarchy controller: It can adjust the desired velocity that deviations can be compensated.
The neurocontroller is managing the untrained environment introduced through the uneven ground very well and allows for an exact control of the horizontal speed. The actual velocity follows the desired velocity by regulating the influence of the uneven ground within a few steps. Even extreme changes in the desired velocity are handled well - the MLP adjusts the system in a few steps.

5 Conclusion and outlook

The implemented neurocontroller based on MLPs is capable of exactly controlling the hopping and running system in three dimensional space. The neurocontroller learned from the given examples to generalise untrained input/output values, i.e. to run at untrained speeds. It did not only perform well on the trained flat surface but also on uneven terrain. MLPs are a suitable way to implement a control task that has been solved separately. Using this approach permits building fast controllers that can provide real time control This is essential for dynamic movement systems.
Where modeling the control task explicitly is not possible or too expensive reinforcement approaches might prove successful. We are currently investigating in reinforcement based learning algorithms for the control of movement systems: an approach using MLP based reinforcement learning like TD(λ)- [18] or Q-learning [19] seems to be promising for future work on non-supervised control of dynamic movement systems.
We plan to implement suitable MLP based control in an miniaturized opto-electronic device operating at high speed and consuming little space and energy.
Having control devices that allow real time control of fast moving dynamic systems which can be mounted on the system will be an issue of further development in autonomous movement systems. High speed of

locomotion is an issue for future artificial walking machines and elastic elements seem to be promising for implementation.

6 Acknowledgements

We would like to thank the anonymous referees and the editor Dr. M. Paprzycki for their valuable comments and suggestions.

7 References

[1] M. H. Raibert, "Legged Robots." In *Artificial Intelligence at MIT – Expanding Frontiers*, P.H. Winston, S.A. Shellard, (Eds.), Cambridge, Mass.: MIT Press, Vol. 2, pp. 149-179, 1990.

[2] R. Blickhan, R.J. Full, "Similarity in multilegged locomotion: Bouncing like a monopode." *Journal of Comparative Physiology A*, pp. 509-517, 1993.

[3] S. Haykin, *Neural Networks*, New York: Macmillan Publishing Company, 1994.

[4] R. Blickhan, "The spring-mass model for running and hopping.", *Journal of Biomechanics*, Vol. 22, No. 11/12, pp. 1217-1227.

[5] M. H. Raibert et al.: "Experiments in balance with a 3D one-legged hopping machine.", *International Journal of Robotics Research*, 3, pp. 75-92, 1984.

[6] R. A. Brooks, "New Approaches to Robotics", *Science*, Vol. 256, 1227-1232, 1991.

[7] D.E. Rummelhart et al., "Learning representations by back-propagating errors.", *Nature* (London), 323, pp. 533-536, 1986.

[8] R.D. Beer et al., "A distributed neural network architecture for hexapod robot locomotion.", *Neural Computation*, No. 4, pp. 356-365, 1992.

[9] U. Berns et al., "Learning control of a six-legged walking machine." In M. Jamashidi, et al. (Eds.), *Proceedings of the 5th International Symposium on Robotics & Manufacturing*, Vol. 5, New York: ASME Press, 1994.

[10] C.W. Anderson, "Learning to control an inverted pendulum using Neural Networks". *IEEE Control Systems Magazine*, Vol. 9, 3, pp.31-37, 1989.

[11] H. Ritter, K. Schulten, "Extending Kohonen's Self-Organizing Mapping Algorithm to Learn Ballistic Movements." *Neural Computers* (Eds. R. Eckmiller, and C. von der Malsburg), Heidelberg: Springer, pp. 393 – 406, 1987.

[12] B. Widrow, "The original adaptive neural network broom balancer." In *International Symposium on Circuits and Systems*, pp. 351-357, May 1987,.

[13] C.G. Atkeson et al., "Using associative content addressable memories to control robots". In Miller,T.W. III et al. (Eds.): *Neural Networks for Control*, Cambridge, Mass.: MIT Press, pp. 255-286, 1990.

[14] K.D. Maier, V. Glauche, C. Beckstein, R. Blickhan, "Controlling one-legged dynamic Movement with MLPs", *International ICSC / IFAC Symposium on Neural Computation (NC'98)*, Technical University in Vienna, Austria, pp. 784-790, September 23rd-25th, 1998.

[15] K.D. Maier, V. Glauche, C. Beckstein, R. Blickhan, "Control of Legged Planar Hopping with Radial Basis Function Neural Networks",; *Proceedings of the Second International Conference on Climbing and Walking Robots (CLAWAR 99)*, University of Portsmouth, 13-15 September 1999, Professional Engineering Publishing Ltd., Bury St Edmunds, pp. 133-141.

[16] K.D. Maier, V. Glauche, C. Beckstein, R. Blickhan, "Control of a one-legged Movement System: Artificial Neural Network Approach with Radial Basis Functions and Self-Organising Motoric Maps.", *Proceedings of the Euromech Colloquium 375, Biology and Technology of Walking*, pp. 82-89, March 23rd-25th 1998.

[17] T. Kluge, Modellbildung und Regelung einer einbeinigen Laufmaschine mit Fuzzy-Logik". Dissertation, Gerhard-Mercator-Universität-GH Duisburg, 1997.

[18] R.S. Sutton, "Learning to predict by the methods of temporal differences.", *Journal of Machine Learning* 3, pp. 9-44, 1988.

[19] C.J.C.H. Watkins, "Learning from delayed rewards", Ph.D. Thesis, University of Cambridge, UK, 1989.

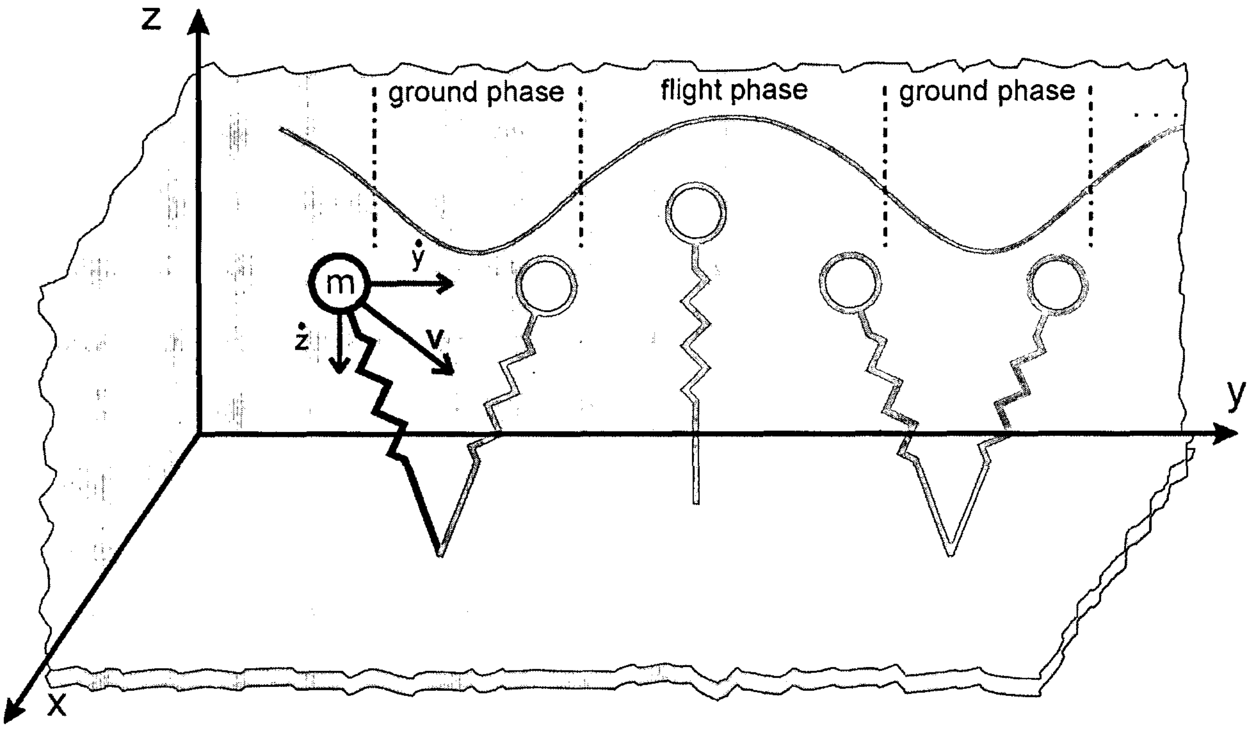


Figure 1: The spring mass movement system jumping across even ground in three dimensional space. (The figure shows the system hopping in a plane with x = constant).

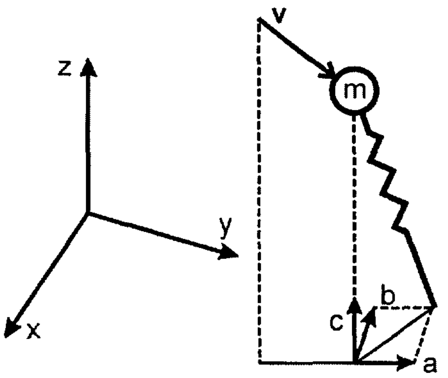


Figure 2: The fixed and changing coordinates.

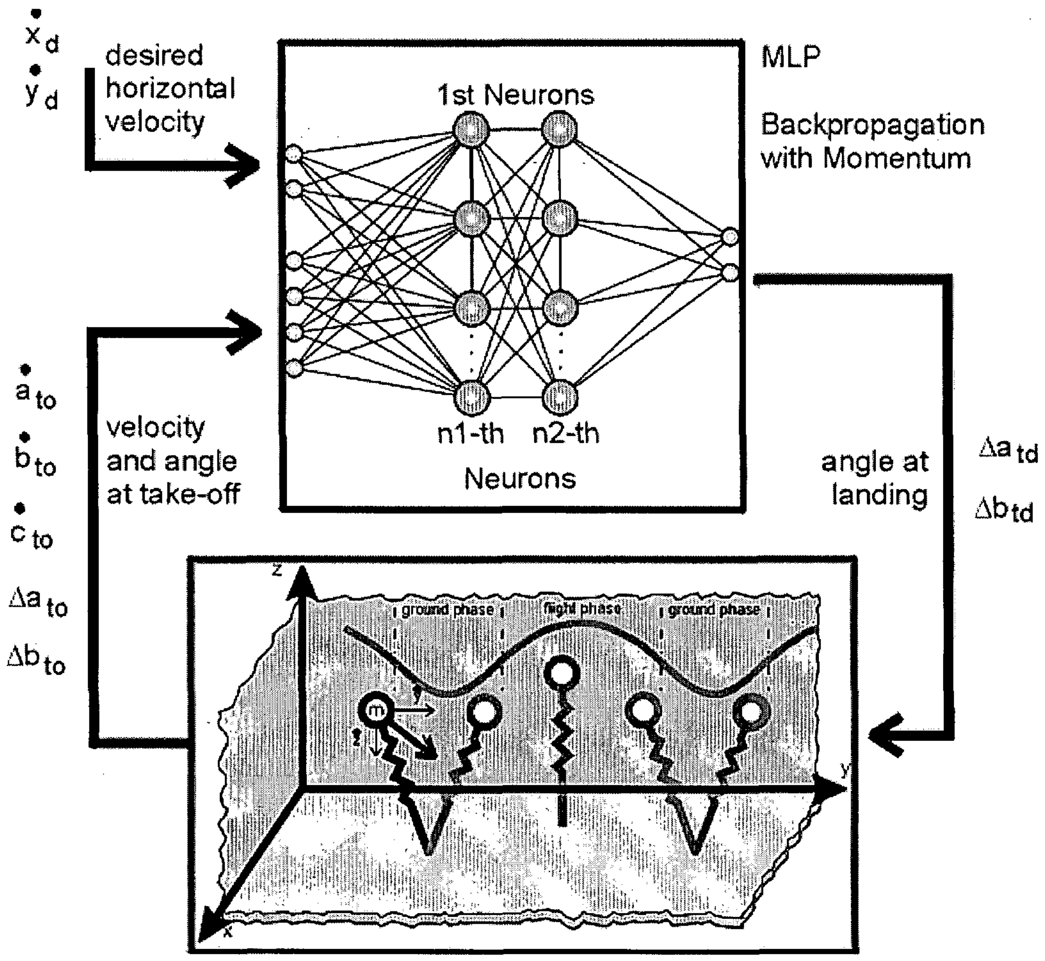


Figure 3: Control architecture: Inputs are the desired velocities, Outputs of the MLP are the landing angles of the leg and the observed states of the movement system are the velocities and angles at take-off.

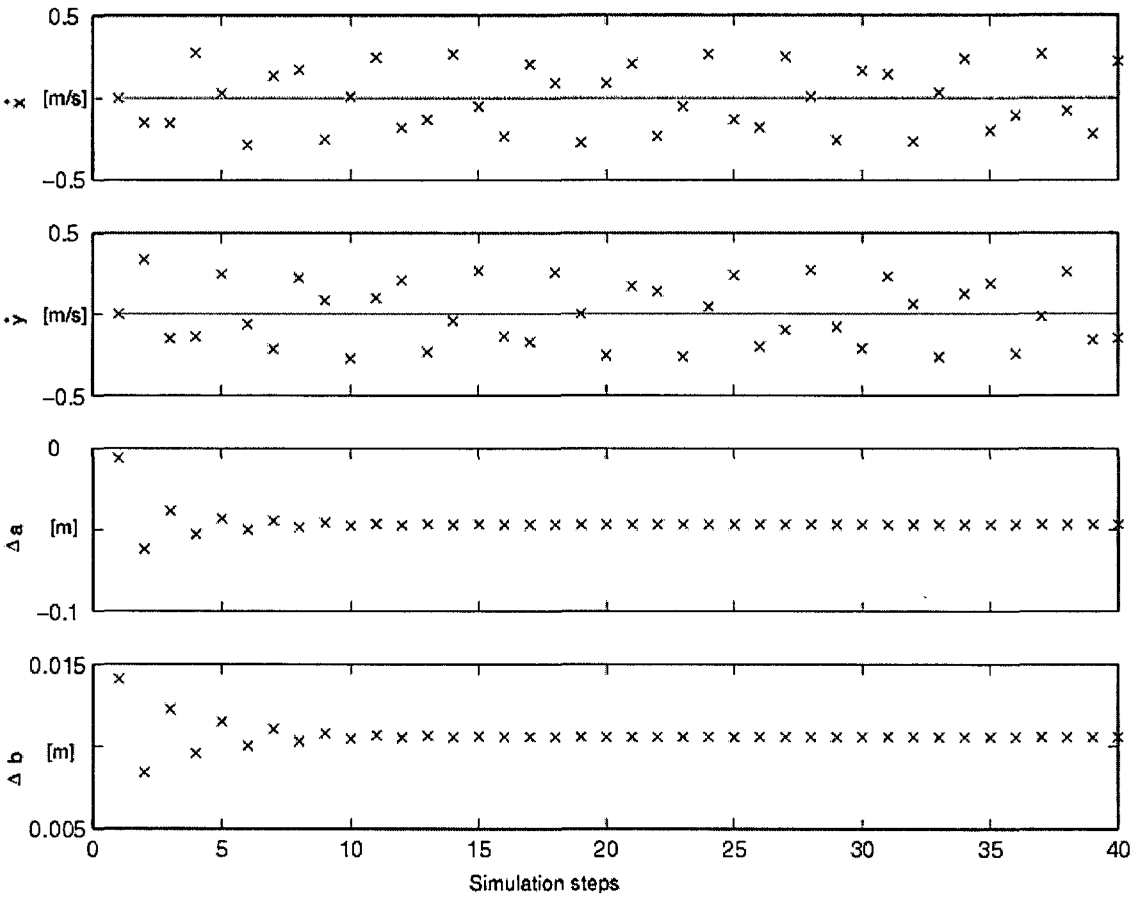


Figure 4a: Movement course for hopping in place.

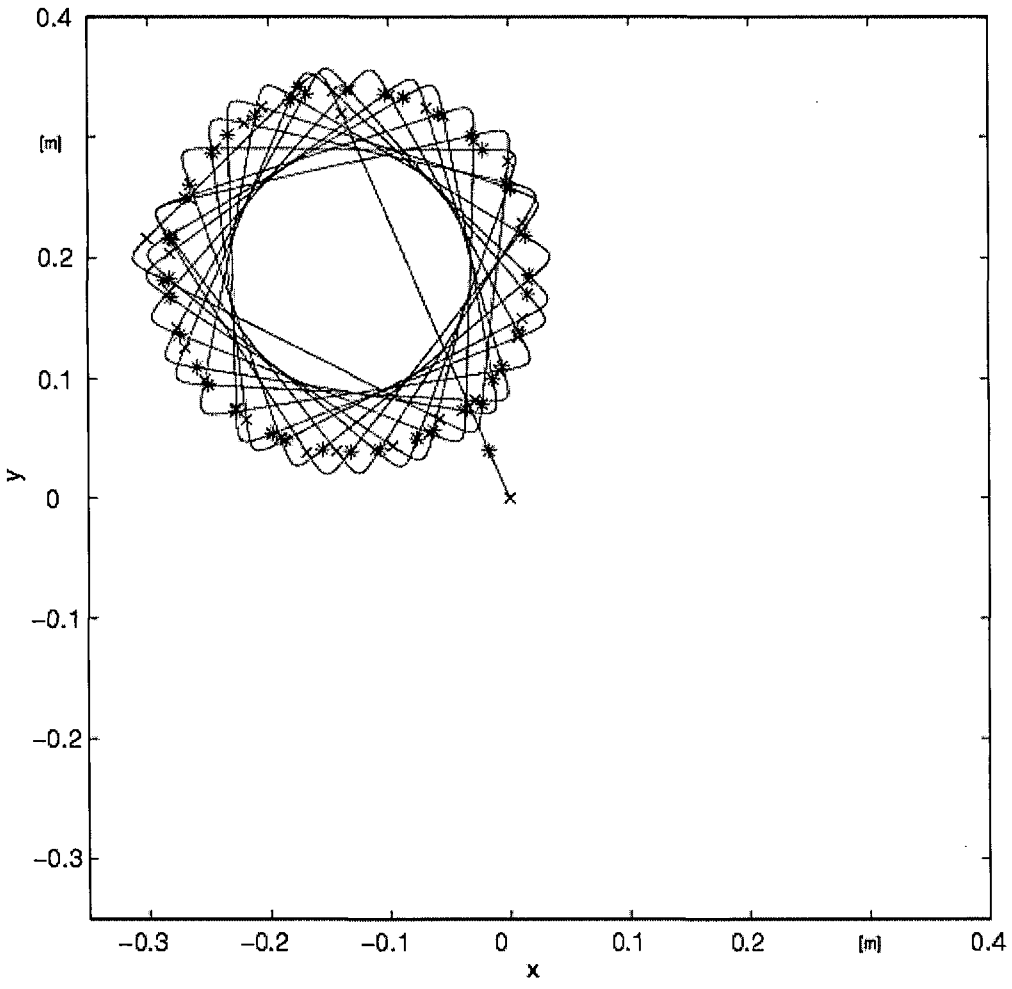


Figure 4b: Movement in the x-y plane for hopping in place.

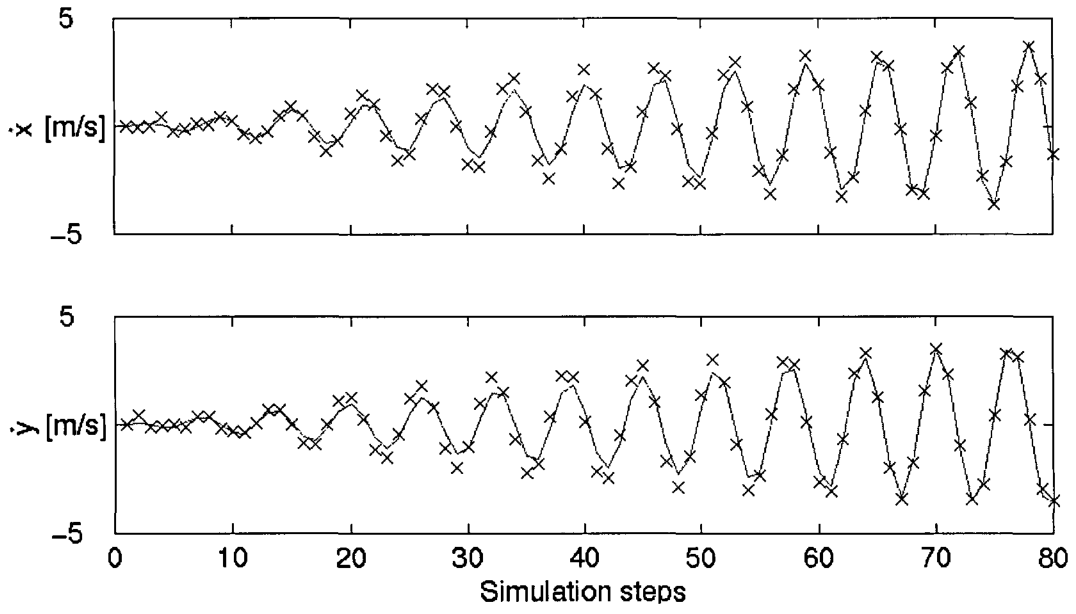


Figure 5a: Movement course for hopping a spiral.

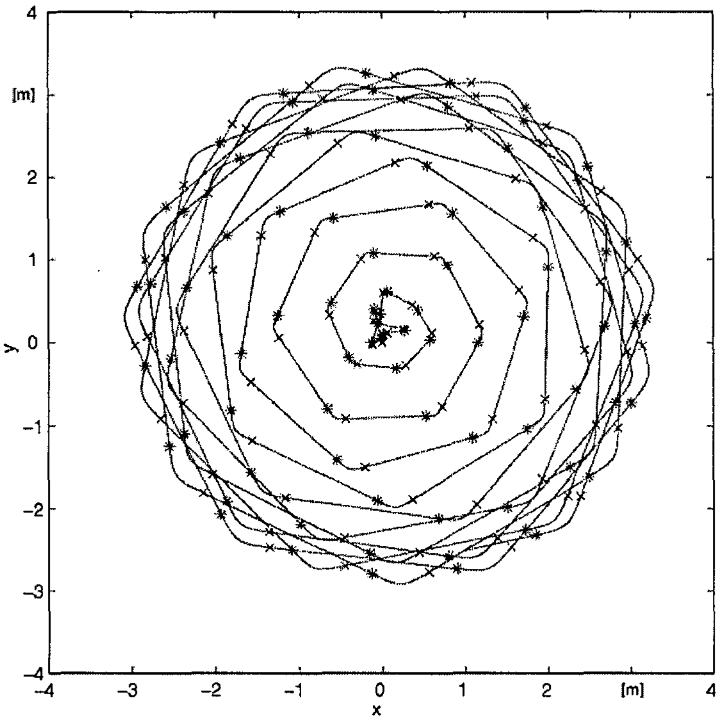


Figure 5b: Spiral hopping in the x-y-plane

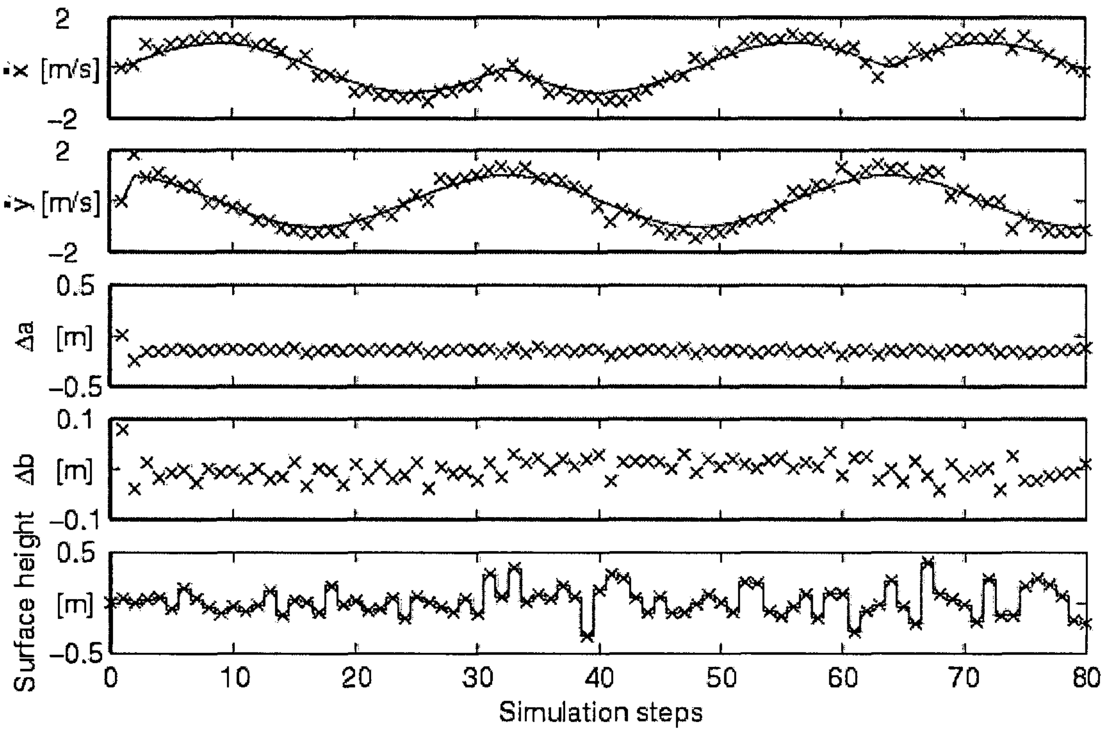


Figure 6a: Movement course for hopping of the figure eight over uneven ground.

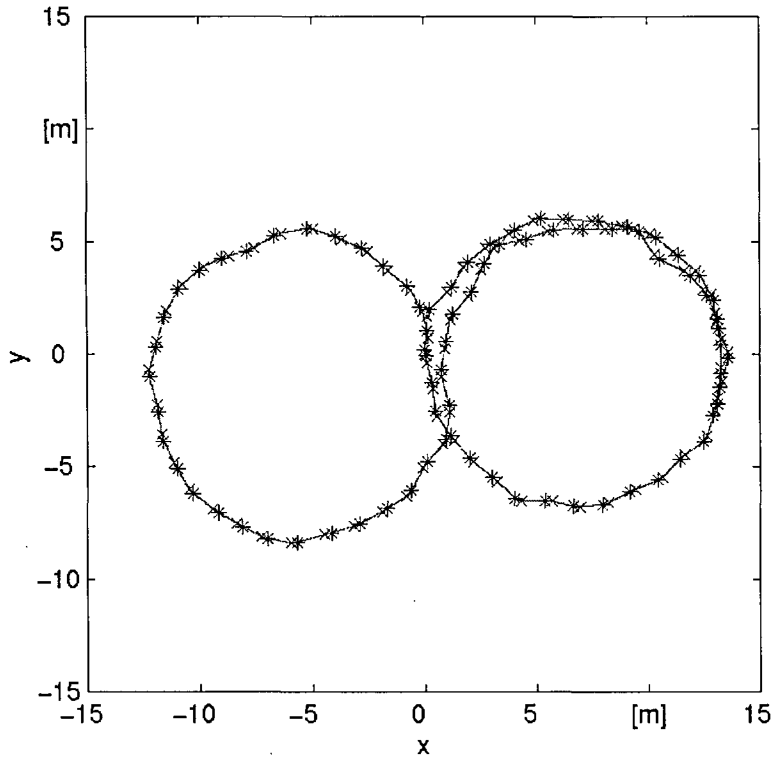


Figure 6b: Hopping an eight figure over uneven ground plotted in the x-y-plane.

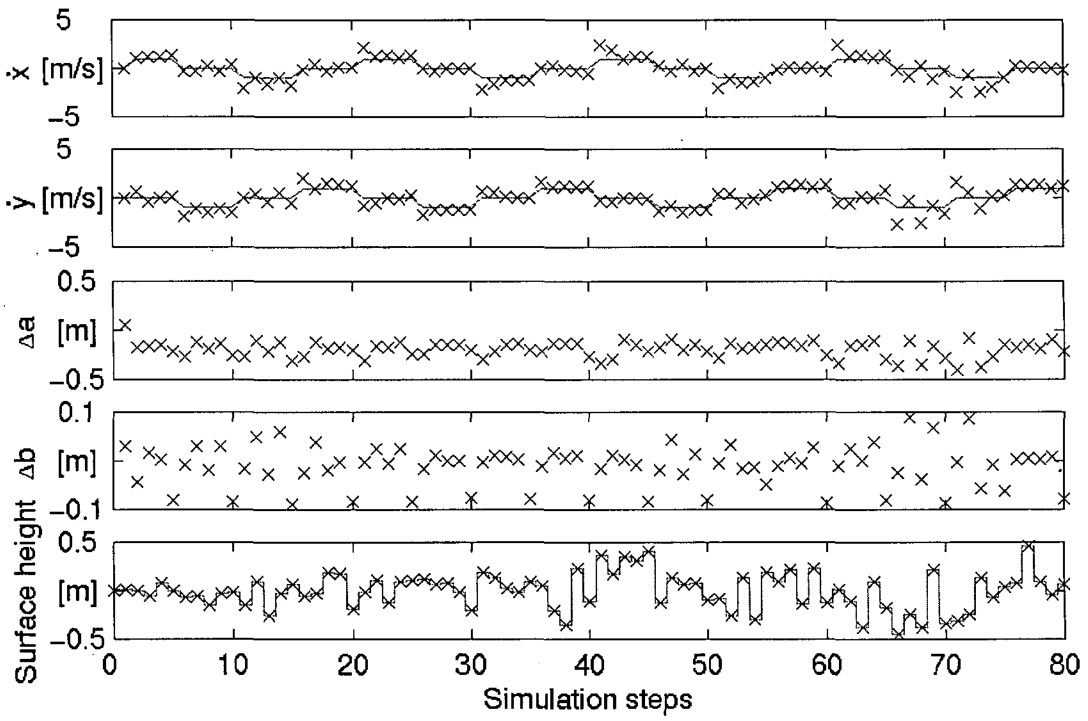


Figure 7a: Hopping squares over uneven ground.

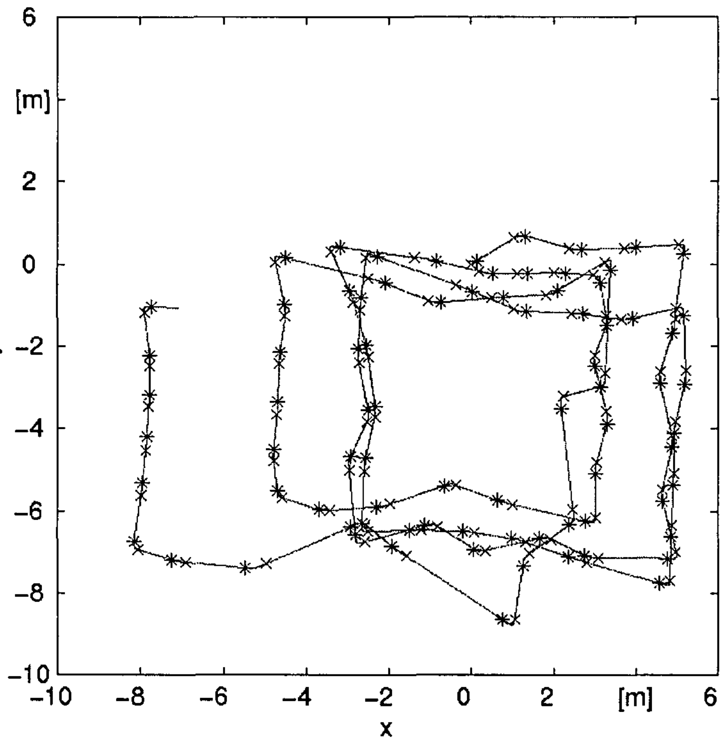


Figure 7b: Hopping square figures over uneven ground plotted in the x-y plane.

Neuro–fuzzy synergism for planning the content in a web–based course

G.D. Magoulas
Department of Information Systems and Computing,
Brunel University, UB8 3PH, United Kingdom
E–mail: George.Magoulas@brunel.ac.uk

K.A. Papanikolaou and M. Grigoriadou
Department of Informatics, University of Athens, T.Y.P.A. Buildings, GR–157 84 Athens, Greece.
E–mail:{spap, gregor}@di.uoa.gr

Keywords: adaptive learning environments, web–based course, adaptive lesson presentation, neural networks, fuzzy logic, learner evaluation, reasoning under uncertainty

Received: January 25, 2000

In this paper, neuro–fuzzy synergism is suggested as a means to implement intelligent decision making for planning the content in a web–based course. In this context, the content of the lesson is dynamically adapted to the learner's knowledge goals and level of expertise on the domain concepts s/he has already studied. Several issues that affect the effectiveness of the lesson adaptation scheme are investigated: the development of the educational material, the structure of the domain knowledge and the assessment of the learner under uncertainty. A connectionist–based structure is proposed for representing the domain knowledge and inferring the planning strategy for generating the lesson presentation from pieces of educational material. The learner's assessment is based on relating learner's behavior to appropriate knowledge and cognitive characterisations and on embedding the knowledge of the tutors on the learning and assessment processes into the system by defining appropriate fuzzy sets. The proposed neuro–fuzzy adaptation scheme is applied to a web–based learning environment to evaluate its behavior and performance.

1 Introduction

Distance Learning through the Web offers an instructional delivery system that connects learners with educational resources. Its main features are the separation of instructor and learner in space and/or time, the use of the educational media/technology to unite instructor and learner and transmit the course content, and the change of the teaching–learning environment from *tutor–centered* to *learner–centered.* The design of a Web–based learning environment includes informed decisions about what comprises the educational content and how this should be sequenced and synthesised, taught and learned. This process is essential in distance learning, where the instructor and the learners typically have minimal face–to–face contact.

The vision of a new generation of web–based learning environments, which possess the ability to make intelligent decisions about the interactions that take place during learning, encourages researchers to look at novel forms of co–operation and communication between tutors, learners, developers and computers and to investigate the technical possibilities for their realisation. Towards this direction, Adaptive Learning Environments (ALE) have instantiated a relatively recent research area that integrates two distinct technologies in computer assisted instruction: Intelligent Tutoring Systems (ITS) and Educational Hypermedia (EH) systems (Brusilovsky 1996). This is in effect a combination of two opposed approaches to computer assisted learning systems: the more directive tutor–centered style of traditional Artificial Intelligence (AI) based systems and the flexible learner–centered browsing approach of an EH system (Davidson 1999).

In this context, the notion of *adaptation* is defined as the concept of making adjustments in the educational environment to accommodate diversity in the learner needs and abilities, in order to maintain the appropriate context for interaction. To this end, in an ALE, the selection, sequencing, and synthesis of educational content takes into account the nature of the content, or task, that is to be taught and also the knowledge level of the learner. The whole procedure is based on understanding the learning and instructional process, as well as the learner characteristics and educational needs (Mc Cormack & Jones 1997).

In general, two methods are proposed in the literature for implementing adaptation in an educational environment: *adaptive presentation*, or *content sequencing*, and *adaptive navigation*, or *link–level adaptation* (Brusilovsky 1996). In the first case, the content of a hypermedia page is generated or assembled from pieces of educational material according to the knowledge state of the learner (Papanikolaou et al. 1999, Vassileva 1997). In the second case, altering visible links to support hyperspace navigation is suggested (Stephanidis et al. 1997, Weber & Specht 1997). Both methods generate a new form of co–operation and communication between learner and system, which is based on the ability of the educational environment to be adapted to the behavior of the learner and make intelligent decisions about the interactions that go on during tutoring. The purpose of adaptation is to avoid information disorientation and overload by presenting the educational material according to the learner's knowledge background and abilities, provide individualised tutoring and, finally, reduce the cognitive effort of learning (Kuhme, 1993).

Many questions are still open in this context. For example, questions related to the role of the tutor as well as of the learner in future ALEs. Furthermore, questions related to the requirements of these systems, the kind of interaction that the learner should have with the system, and the development of appropriate methods of assessing information about the behavior of the learner in the course of learner–system communication. Finally, the organisational and social problems that may arise from the application of these systems have not been investigated thoroughly.

This paper investigates the use of methods from computational intelligence, such as fuzzy logic and artificial neural networks, to handle inexact information about the learner, incorporate tutor's viewpoint into the educational environment and perform lesson adaptation. To this end, a neuro–fuzzy approach is proposed in order to adapt the content of the hypermedia page accessed by a particular learner to current knowledge level, goals, and characteristics of the learner. In this way the educational environment performs adjustments appropriate to each learner by restricting the navigation space in order to protect learners from information overflow. The proposed neuro–fuzzy adaptation scheme incorporates ideas from cognitive science to evaluate learner's knowledge under uncertainty and structure the domain knowledge of the course.

The paper is organised as follows. In Section 2, we present the procedure we have adopted for the development of the educational material. Sections 3, 4 and 5 suggest some novel alternatives to the reasoning and knowledge representation mechanisms in the context of ALE systems, which are based on the use of neuro–fuzzy methods. The connectionist knowledge representation model is presented in Section 3. Section 4 proposes an approach to instructional design that exploits the connectionist–based structure of the domain knowledge. Section 5 suggests neuro–fuzzy synergism to evaluate the knowledge of the learner on already studied concepts of the lesson. In Section 6, applications of these methods in the context of a Web–based learning environment for distance learning are presented. The paper ends, in Section 7, with a discussion and concluding remarks.

2 Developing the educational material

The learning process requires motivation, planning, and the ability to analyse and apply the educational material being taught. In a traditional lecture, the teacher relies on a number of visual cues from the students to enhance and adapt the instructional process. A quick glance, for example, can reveal who is attentively taking notes, pondering a difficult concept, or preparing to make a comment or a question. This type of feedback is missing from a distance learning course and the educational material has to accommodate, in a way, this entire interaction; this can be done, for example, by embedding all the possible questions and common learners' misunderstandings. Therefore, the selection, sequencing, and synthesis of the educational material of a Web–based course must be based on understanding the context of learning, the nature of the content, or task that is to be taught, the instructional objectives, the learners' characteristics, preferences and educational needs, the processes of learning and the constraints of the medium. Consequently, in a Web–based learning environment, the educational material has to incorporate different types of information and levels of explanation, address different learning styles and educational needs.

The following procedure for the development of the educational material has been proposed by Grigoriadou (Grigoriadou et. al. 1999b) and applied for the development of a web–based module named "Introduction to Computer Science and Telecommunications" (DIUA 1999):

- *Create the content outline* based on analysing the audience, defining instructional goals and objectives.
- *Review educational material* that has been proven effective in the traditional lectures.
- *Develop and organise* the educational material following a predefined structure. It is divided into manageable segments: chapters, units, sub–units, and pages. A chapter is a collection of units, while a unit is a collection of pages, tests and (optionally) sub–units. The educational material includes definitions of domain concepts, texts written in a user–friendly way incorporating various levels of explanation, diagrams–images, examples, exercises and simulations and adopts a hypermedia way of presentation.

The presentation of the domain knowledge follows principles that lead to the "deep approach" of learning, that is to relate new ideas to previous knowledge and new concepts to every day experience. Furthermore, this approach aims to organise and structure the content, supplement the theory with a variety of practical tasks and activities and, finally, provide learners with self–assessments and assessments to test their knowledge (Vosniadou et. al. 1996).

However, the greatest challenge in the presentation of the educational material is to build an environment in which the learners are motivated to assess their personal knowledge goals and objectives, and become active participants in the overall learning process. The research literature suggests that the appropriate match of the students to the learning experience has a significant impact on their achievement (Bennett et. al. 1984). Furthermore, instructors need to provide opportunities for students to learn in a way that suits their preferred style of learning (Ellis 1994). Adaptive lesson presentation is a promising research area towards the development of a learning environment in which the learners are motivated to assess their personal knowledge goals and objectives in a way that suits to their learning preferences and knowledge level.

3 Modeling the knowledge of the domain

A key point in producing a learner–adapted system, i.e. a system that meets the individual educational needs and objectives of each particular learner, is to structure the domain knowledge in such a way that it will be possible to do adaptations.

The structure of the domain knowledge is based on symbolic methods and is usually represented as a semantic network of domain concepts, or generally elementary pieces of knowledge for the given domain, related with different kinds of links (Brusilovsky 1996). Alternatively, the use of a concept level hierarchy (Anjaneyuly 1997), or a graph of concepts (Vassileva 1997) has been suggested.

A subsymbolic approach is proposed in this paper: a connectionist–based structure of the domain knowledge is presented that allows the adaptation of the educational material to the individual learner's level of understanding. The subsymbolic approach provides an attractive alternative to traditional symbolic AI methods since it exploits the well known generalisation capabilities of the artificial neural networks to handle the uncertainty in modeling the knowledge of the learner and introduce human–like reasoning (Kasabov 1996). Human–like reasoning aims to support adaptivity and provide ALE with the ability to enhance and adapt the instructional process according to the learner in the human–tutors way. The main characteristic of the proposed approach is that the decomposition of the domain knowledge in modules (see Figure 1), such as knowledge goals, concepts, educational material is incorporated in the connectionist architecture.

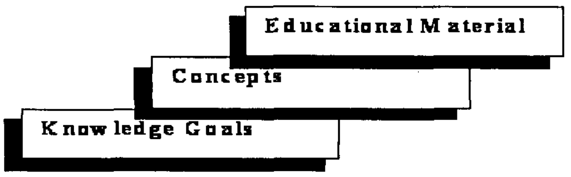


Figure 1: The domain knowledge of the lesson.

An important issue in the development of an environment that will support pedagogical decisions is to provide various types of educational material on the same knowledge (Grigoriadou et al. 1999b). As a first step towards this direction we have mapped the domain knowledge in the three layers of a connectionist model, as shown in Figure 2, with each layer providing a different type of information.

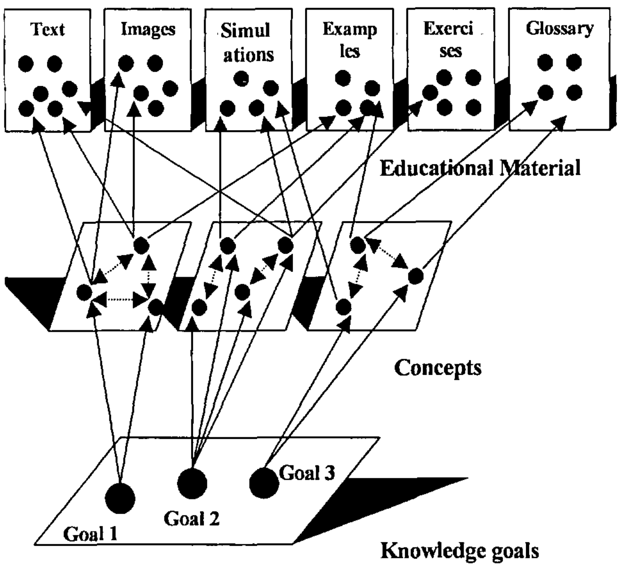


Figure 2: Connectionist–based structure for the domain knowledge.

The architecture is based on the notion of knowledge goals that learners willingly adopt, in an attempt to provide learners with a means of regulating the environment in which they learn. In addition, it gives them the opportunity to select the next knowledge goal according to their educational needs. To this end, in the first layer the knowledge goals, which are referred to a subset of the domain knowledge, are defined while the second layer consists of the concepts of the domain knowledge. In the third layer the educational material related to each concept is represented in different categories, such as text, images, simulations, examples, solved and unsolved–exercises and so on.

A knowledge goal, in the first layer, is associated with its corresponding concepts in the second layer. Each concept corresponds to a single concept node of a specially designed dynamic neural network for each goal, named Relationships Storage Network–RSN (Michos et al. 1995). Each RSN is described by:

$$\mathrm{x}(k+1) = sat(\mathrm{T}\mathrm{x}(k) + \mathrm{I}), \qquad (1)$$

where x is a real n–dimensional–vector with components x_i $(i = 1, ..., n)$, which denotes the state or activity of the i–th concept node; T is a $n \times n$ symmetric weight matrix with real components T(i,j); I is a constant vector with real components I(i) representing external inputs; sat is the saturation activation function ($sat(t) = 1$, if $t \geq 1$; $sat(t) = -1$, if $t \leq -1$; $sat(t) = t$ otherwise).

The training of each RSN is performed off–line using groups of patterns that establish relationships among concepts for a knowledge goal and are defined on $\{-1, 1\}^n$. A storage algorithm that utilises the *eigenstructure* method is used for specifying the appropriate T and I parameter values (Michel et al. 1991). This algorithm guarantees that patterns of concept combinations are stored as asymptotically stable equilibrium points of the RSN (see Michel et al. 1991 for a description of the algorithm). When two or more concepts are active in a pattern, i.e. the corresponding components of the pattern are $\{1\}$, this indicates that a relationship among these concepts has to be established. Relationships among concepts are represented by the internal connection weights (the matrix $\mathbf{T}$). Note that the groups of patterns are generated in accordance to particular strategies for planning the content of the lesson. The human instructional designer has determined these strategies (more details on the planning strategies will be presented in the next section).

The RSN operates in a saturated mode for all discrete points in time, $k = 0, 1, 2, \ldots$, and its equilibrium points are located on the boundary of the $[-1, 1]^n$ hypercube, i.e. the state space is the set of vertices of $[-1, 1]^n$:

$$\begin{aligned}\{-1,1\}^n &= \{\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \mid \\ &\quad x_i \in \{-1,1\}, \forall\, 1 \le i \le n\}.\end{aligned}$$

Note that during operation, the node inputs (initial states) are supplied after evaluating the learner's knowledge on the concepts of a knowledge goal; details on the evaluation procedure will be presented in Section 5. Depending to the pattern applied to the input, the state vector of the RSN is forced to move to a certain part of the concepts' space. In this way, the RSN performs associate inference depending on the input pattern and, unlike the general use of an associative memory, it operates synchronously: (*i*) it updates the states of its nodes simultaneously, and (*ii*) the input pattern is kept unchanged until convergence of the network (see Michel et al. 1991 for details on the operation of this type of dynamic neural network).

In Table 1, an example of knowledge goal, referred to the chapter *Network Architectures*, with its associated concepts (the corresponding RSN has $n = 26$ nodes) is exhibited. Note that the *Number* (No) for the outcome concepts in the table indicates their sequencing in the course. Other examples of knowledge goals are: Data Communication Basics, Transmission Means, Network Topology, Local Area Networks, Wide Area Networks, Internet Administration, etc.

In the third layer of the connectionist model, the educational material related to each concept is organised in categories. Weights connecting the second and the third layer are unique for each concept, and each concept may be associated with several categories of educational material. The educational material is then joined under a predefined form of presentation to generate a lesson. The associated concepts receive different characterisations depending on the knowledge goal: some of them, named *outcome* concepts, are fully explained in the HTML pages constructed for the corresponding goal using text, images, examples, exercises etc.; others, named *related* concepts, are simply mentioned in the HTML pages of the goal. These are related to specific outcome concepts but they are not so important for the selected goal. Finally, there are *prerequisite* concepts,

Table 1: The 26 concepts of the Knowledge goal "ISO Architecture".

No	Outcome	Prerequisite	Related
1	Multi–layer architecture	Layer, Communication protocol	
2	Open Systems Interconnection		I.S.O.
3	Physical layer	Transmission means, Synchronisation	Digital transmission
4	Data Link layer	Packet, Error detection and correction	Transmission means
5	Network layer	Packet routing	Packet
6	Transport layer	Flow control, Traffic metering	Packet
7	Session layer	Synchronisation, Communication half/full duplex	Packet
8	Presentation layer	Compression, Encryption	Compatibility
9	Application layer	File transfer protocol, Virtual terminal	

which are necessary for the learner to understand the outcome concepts of a goal. Thus, a generated lesson includes:

- complete presentation, in terms of text, images, examples, and simulations (if any), of the outcome concepts;
- links to the main HTML pages of the prerequisite concepts;
- links to the related concepts in a glossary;
- tasks and questions.

4 The instructional designer

The issue of instructional design is of major importance in the development of an educational environment (Burns & Capps 1988, Reusser 1996). In this context, *instructional planning* is the process of mapping out a global sequence of instructional goals and actions that provides consistency, coherence and continuity in the instructional process (Dijkstra et. al. 1992, Vassileva 1995). In our case this can be applied in two levels:

- by *planning the content*; that is, presenting concepts related to the selected knowledge goal by taking into account learner's background knowledge. In this way, the content of a hypermedia page is generated from pieces of educational material based on a goal–oriented way of teaching which is supposed to be adequate to adults who are motivated to learn a specific knowledge goal.

- by *planning the delivery* of the educational material. The planning of delivery is responsible for the optimal selection of the educational material, tutorial activity and presentation style, i.e. the appropriate teaching method. The use of multiple approaches in teaching methods increases the possibilities to meet the needs of a wide range of learners who have different learning styles, time constraints and abilities.

Below, we propose the formulation of the planning strategy retrieval for selecting the content of a knowledge goal in the context of the dynamics of the connectionist network.

To this end, different strategies for planning the content are implemented by means of the stored patterns of the RSN: a *strategy*, in the form of a collection of m patterns defined on $\{-1, 1\}^n$, is stored in the RSN using the eigenstructure method (Michel et. al. 1991); some examples are:

- *Strategy A*: learner achieves a knowledge goal when s/he studies successfully all the outcome concepts of this goal.
- *Strategy B*: learner has successfully studied all the prerequisite concepts of a knowledge goal. Then, in order to achieve this goal, s/he has to study only the outcome and the related concepts.
- *Strategy C*: learner has successfully studied several prerequisite or related concepts of a knowledge goal. Then, in order to achieve this goal, s/he has to study the entire outcome concepts and the rest of the prerequisite and related concepts.
- *Strategy D*: learner has failed in a number of outcome concepts. Then in order to achieve this goal, s/he has to study only these outcome concepts and their prerequisite and related ones.

As mentioned in the previous section, the patterns of relationships are stored as asymptotically stable equilibrium points of the RSN and the network is capable of organising its internal states in accordance to the underlying structure of the stored patterns. During tutoring, the evaluation of the learner's knowledge on the concepts of a goal (this will be described in the next section) formulates the input pattern (initial state vector) of the RSN. The input pattern, during the recall operation of the dynamic network, converges to an equilibrium point, i.e. to one of the m stored patterns that implement a planning strategy. This equilibrium point (i.e. the final state vector) defines the output response of the RSN and is used for deciding the content of the lesson, i.e. present the concepts of the knowledge goal that the learner has to learn next.

The planning of the delivery is also based on the results of the real–time recall operation and establishes instructional objectives (tutorial activities), which are specific steps leading to the knowledge goal attainment.

However, the optimal selection of the educational material should also take into account the relevant importance of each concept for achieving a knowledge goal, as well as the progress of the learner concluding from the learner–evaluating module. Currently, the selection of the educational material is based on weighted priorities, as these are defined by the weight values connecting the second and the third layer of the connectionist network.

5 Evaluating the learner

The assessment of learner's knowledge is based on an *overlay model*. The idea of the overlay model is to represent an individual learner's knowledge of the subject as an "overlay" of the domain knowledge. For each domain concept, an overlay model stores some value (binary/ qualitative measure/probability), which is an estimation of the learner knowledge level for this concept. Overlay models are domain independent and flexible and were originally developed in the area of ITSs and learner modeling (see the work of Wenger (Wenger 1987) for a review on ITSs). In our approach, each concept is associated with linguistic rating values characterising the learner's knowledge, i.e. *{EI, I, RI, RS, AS, S} = { Extremely Insufficient, Insufficient, Rather Insufficient, Rather Sufficient, Almost Sufficient, Sufficient }*. This scale has been experimentally found to provide evaluation results closer to human–tutors evaluation performance, when compared with previous work in the area (Panagiotou & Grigoriadou 1995, Stathacopoulou et al. 1999). The exact rating value is calculated by means of the three–stage evaluation procedure that will be described below.

Learner's knowledge assessment is based on two types of information: answers to questions that evaluate the cognitive part of the learner's knowledge (Nkambou 1999), and measurements that evaluate the behavior part, which is related to awareness, interest, attention, concern, and responsibility factors (Embenson, 1990; Krathwohl et al. 1964). In both cases, several factors contribute to uncertainty in the evaluation procedure, such as careless errors and lucky guesses in the learner's responses, changes in the learner knowledge due to learning and forgetting, and patterns of learner responses unanticipated by the designer of the learner model. Thus, the development of an accurate model for evaluating the learner's knowledge is based on uncertain information.

Various types of questions organised in categories can be used, e.g. multiple choice, fill–in–the–blanks, multiple correct answers. Each question is related to a subset of the domain knowledge that the learner should acquire and should have a weight representing its importance or complexity as regards the evaluation of a knowledge characteristic and ability of the learner (Bertles 1994):

- *Fact oriented questions*: Questions regarding memorising that are used to test the knowledge of concept definitions and topic aspects directly related to the content of the lesson.

- *Higher order comprehension questions*: They are used to test the understanding of the conceptual and semantic units of the lesson. Their aim is to test the conceptual model constructed by the learners and evaluate their misconceptions.

- *Generalisation questions*: Questions examining the ability of comparison, differentiation, abstraction and generalisation.

- *Questions related to the recognition of functional interrelations*: They test the ability of linking newly acquired with already existing knowledge on the concepts.

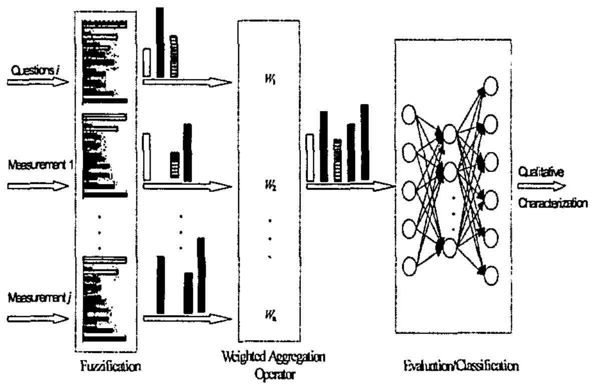
For example, in the web-based module "Introduction to Computer Science and Telecommunications" offered by the Department of Informatics of the University of Athens (DIUA 1999, Grigoriadou et al. 1999b), identifying comprehension regarding the concepts *data link layer*, *network layer*, and *session layer* is performed using questions like:

- *Which of the OSI layers handles each one of the following functionalities:*
  1. Breaking and transmitting bit streams into frames.
  2. Determining which route to use through the subnet.
  3. Providing synchronisation.

In order to answer the above question the learner has to recall his/her knowledge regarding the *multi-layer structure of the OSI model*. These questions help to test learner's comprehension of the functions undertaken by each layer of the OSI model.

In addition, several measurements are recorded from the learner–educational program interaction and used for evaluating the learner's behavior. For example, the number of questions and exercises that the learner tried to answer or solve, the points scored, the number of learner attempts before giving the correct answer, the frequency of the encountered misconceptions, the number of repetitions of a topic by the learner, the time s/he spends for self-assessment, the type of information the learner prefers (text, pictures, sound, video, simulations, URLs) and how often s/he navigates through the HTML pages of the educational material supplied for a knowledge goal. Thus, by analysing the learner's answers and by processing the various measurements conducted by the system, it is possible to trace gaps in the knowledge of the learner.

To this end, a three stages neuro–fuzzy procedure (see Figure 3), originally proposed in (Panagiotou & Grigoriadou 1995) and extended in (Grigoriadou et. al. 1999a), is applied. The first stage fuzzifies inputs that contribute to the evaluation of the level of understanding based on the estimations of experts to the degree of association between an observed input value and the learner's knowledge on the concepts. Note that a 9–level discretisation of the universe of discourse is applied to the inputs. Depending on the input, a fuzzy subset is generated for each measurement or answer contributing to the evaluation. The next



Questions i
Measurement 1
Measurement j
Fuzzification
W_1
W_2
W_k
Weighted Aggregation Operator
Evaluation/Classification
Qualitative Characterization

Figure 3: The three stages of the knowledge evaluation procedure.

stage realises a weighted aggregation operation utilising the intersection operator that processes these fuzzy subsets. The weights are evaluated using the Saaty's method (Saaty 1978) and determine the importance of each preliminary decision, expressed by a fuzzy subset, in evaluating the learner's knowledge. The last stage consists of a Multi–Layer Perceptron (MLP) that evaluates the knowledge of the learner with regard to a concept by classifying him to one of the categories $\{EI, I, RI, RS, AS, S\}$. To this end, the max rule is applied to the output vector of the MLP. The MLP is trained using the BPVS algorithm (Magoulas et al. 1997) to imitate the tutors evaluation procedure and can be adapted to a tutor's subjective evaluation procedure.

Depending on the concept, the qualitative characterisation of the learner's knowledge is converted to a numeric value in order to feed the RSN. In this way, an n–dimensional input pattern (initial state vector) is formulated, where n depends on the number of concepts of the knowledge goal and determines the number of the corresponding RSN nodes. This form of feedback from learner evaluation procedure guides the instructional method adopted though the RSN recall operation. Note that, when the learner's knowledge with regard to a concept is characterised as *Extremely Insufficient*, a value of approximately 1 is assigned to the corresponding component of the input pattern. This means that the learner certainly has to study this concept. On the other hand, a small value of approximately 0.1 is assigned when the learner's knowledge on a concept is evaluated as *Sufficient*. The exact magnitude for each value is heuristically chosen and depends on the importance of the concept in achieving a knowledge goal and on its characterisation (outcome, related, prerequisite). For example, the set:

$$\begin{aligned}\mu(x) &= \{1|1 + 0.9|2 + 0.75|3 + 0.65|4 \\ &\quad +0.59|5 + 0.1|6\}\end{aligned}$$

is used for the outcome concept *Multi–layer Architecture*, where an integer value $1, 2, \ldots, 6$ is mapped to a linguistic term $\{EI, I, RI, RS, AS, S\}$ and the symbols " | " and " + " are used only as syntactical constructors. This set can be interpreted as the degree of membership of the learner's knowledge on the concept *Multi–layer Architecture* in each of the fuzzy sets associated with the 6 linguistic terms. Some other examples are the sets:

$$\mu(x) = \{0.98|1 + 0.88|2 + 0.58|3 + 0.38|4 + 0.28|5 + 0.1|6\},$$

for the prerequisite concept *Communication protocol*, and

$$\mu(x) = \{0.95|1 + 0.74|2 + 0.55|3 + 0.24|4 + 0.14|5 + 0.1|6\},$$

for the related concept *Compatibility*.

The three stages neuro–fuzzy procedure allows us to incorporate both general and subjective knowledge in the evaluation procedure. General knowledge is incorporated in the definition of the fuzzy sets and in assigning weights to the different performance parameters, which assess the cognitive and behavior parts of the learner's knowledge. This general knowledge is based on the expertise of the tutor in determining the characteristics of the learner and is kept fixed during the learner's evaluation procedure. On the other hand, the MLP, which constitutes the final evaluation stage, represents the experience of the human tutor in evaluating learners and can be adapted by training to a tutor's personal way of evaluation. Furthermore, this hybrid approach permits the representation and processing of incomplete, imprecise and vague information about the learner, i.e. controversial answers and unstable behavior and the exploitation of the MLP generalisation capabilities.

6 Experiments

The Web offers a new way to receive and disseminate information for educational purposes. The chapter on *Network Architectures* of the Web–based module "Introduction to Computer Science and Telecommunications", (DIUA 1999, Grigoriadou et al. 1999b), offered by the Department of Informatics to first year undergraduate students, has been used for testing the proposed approach. In this chapter adult learners have to study 25 knowledge goals, each one containing 10–30 concepts. Experiments have been conducted to evaluate the behavior of the proposed model in adapting the content of a lesson.

Next, the performance of the neuro–fuzzy approach is illustrated in three cases. This low–level test of the system helps to show how the connectionist model and the knowledge evaluation procedure function together to create an operational system.

In the first case, the learner has selected the goal "ISO Architecture" and his performance has been evaluated as *Sufficient* with regard to several prerequisite and related concepts. The knowledge evaluation procedure supplies the corresponding RSN with an input pattern and initialises the RSN operation; the network will eventually come to rest at one of its equilibrium points. To this end, the RSN runs for several cycles and finally settles into a stable state defined by its 26–dimensional state vector. Since, a localist concept coding is used, i.e. each node stands for one domain concept that is related to the selected goal, it is easy to follow the nodes activity changes in response to an input pattern at each recall cycle. For example, 5 out of the 26 node activity levels are exhibited in Figure 4. Following planning strategy C the generated lesson includes all the concepts of Table 1 apart from the successfully studied prerequisite and related ones: *Layer*, *Data compression*, *Encryption*, *Virtual terminal*, *ISO*, *Traffic metering*.

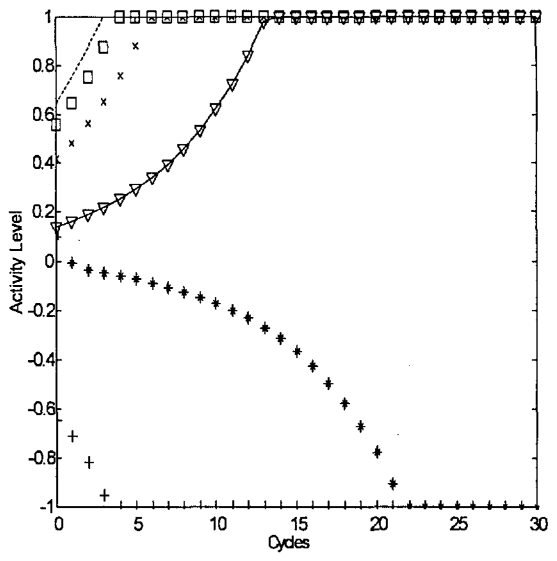


Figure 4: Example of planning strategy C. The concepts: *Communication half/full duplex* ( ), *Compatibility* ( ), *Layer* ($*$), *Network layer* ($\times$), *Packet* ($\triangle$), *Synchronisation* (- - -), *Virtual terminal* ($+$).

□

In Figure 4, it is shown that the activity of the concept node that represents the concept *Virtual terminal* goes to -1, which means that the node is deactivated and the educational material associated with this concept will not be presented. On the other hand, the activity level of the related concept *Compatibility*, in which learner's knowledge has been evaluated as *Insufficient*, goes to $+1$ and the material will be presented. Note that, a node activity level at cycle $= 0$, after transformation to the interval $(0, 1)$, is related to the result of the learner's knowledge evaluation procedure for the corresponding concept. Thus, the learner's knowledge has been evaluated as *Insufficient* for the *Compatibility* node, and the value of 0.74 has been assigned to the corresponding component of the input pattern (cf. with this concept's set in Section 5). Similarly, the concept node *Network layer* is activated since the learner has been evaluated as *Rather Sufficient* in this outcome concept.

In the second case, trying to acquire the same knowledge goal, a learner exhibits performance that is characterised as *Extremely Insufficient* with respect to several outcome concepts.

Following planning strategy D, a lesson is generated that presents these outcome concepts, their prerequisite and related ones. The following concepts of Table 1 are included in the lesson: *Multi–layer Architecture, Layer, Communication protocol, Physical layer, Transmission means, Synchronisation.*

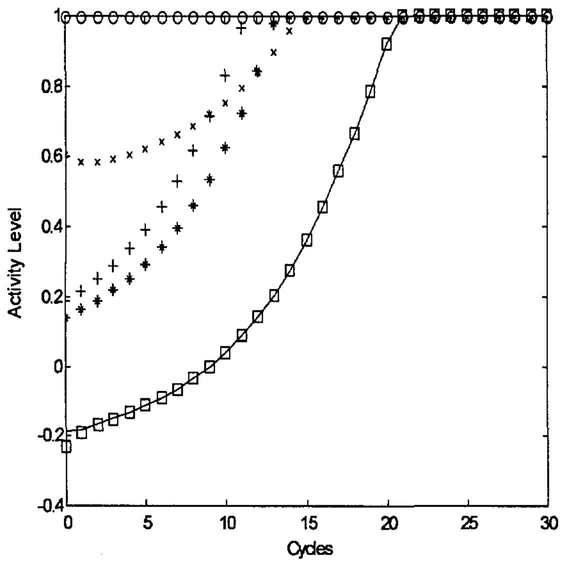


Figure 5: Example of planning strategy D. The concepts: *Communication protocol* ( ), *Layer* ( ), *Multi–layer Architecture* (×), *Physical layer* (o), *Synchronisation* (∗), *Transmission means* (+).

A sample of the RSN's response is shown in Figure 5, in terms of the nodes' activity level. The evaluation of the learner's knowledge with respect to the outcome concept *Physical layer* indicates that s/he has been *Extremely Insufficient*; thus, the corresponding node is rapidly activated. The node of the outcome concept *Multi–layer architecture* is also activated, as well as the nodes corresponding to its prerequisite concepts *Layer* and *Communication protocol*, in which the learner's knowledge has been evaluated as "0.38|*Rather Sufficient*".

The third case is an example of planning strategy B, i.e. a learner with a performance characterised as *Sufficient* regarding all the prerequisite concepts of a goal.

The generated lesson includes, apart from the prerequisite ones, all other concepts of the goal: *Multi–layer architecture, Open Systems Interconnection, International Standards Organisation, Physical layer, Digital transmission, Data Link layer, Network layer, Transport layer, Session layer, Presentation layer, Application layer, Compatibility.* The RSN response in 8 out of the 26 concept nodes is exhibited in Figure 6.

In general, our tests showed that the fuzziness associated with the evaluation of learner's knowledge was handled well by the connectionist model. The performance of the system with 80 learner profiles has been significantly high, almost reaching 100%.

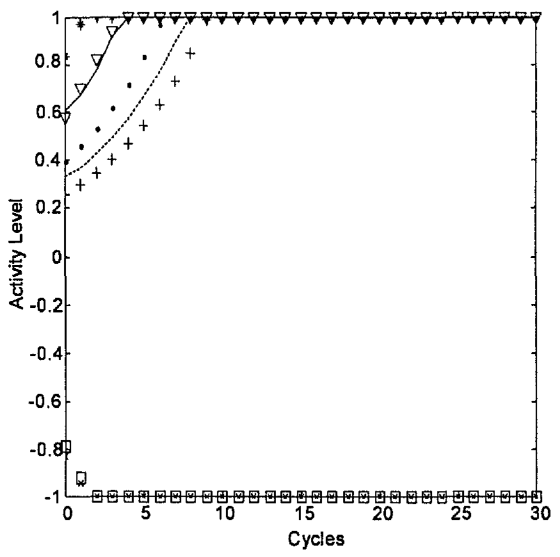


Figure 6: Example of planning strategy B. The concepts: *Application layer* ( ), *Communication half/full duplex* ( ), *Data Link Layer* (∗), *Layer* (×), *Multi–layer Architecture* (△), *Open System Interconnection* (- - -), *Physical layer* (•), *Presentation layer* (+).

7 Discussion and conclusions

Adaptive lesson presentation is a promising method for introducing flexibility into an educational environment. Lesson adaptation aims at minimising the information disorientation and overload of the learner by adapting the educational material to the learner's background knowledge. The design of the educational material supporting such functionality is an important issue contributing to the effectiveness of the adaptation of the overall educational environment.

This paper has described some important parts of the educational environment, which support the adaptivity, how they are designed and how they can be implemented. The paper has focused on two important aspects of developing an adaptive educational environment: the design, i.e. the structure of the domain model and the method of learner's assessment, and the low–level test of the system to show how the two components function together to create an operational system.

An important issue in system adaptation is to adopt a structure of the domain model that will facilitate adaptation of the lesson to the learner's needs. To this end, a specialised connectionist architecture has been developed and a formulation of the planning strategy retrieval for selecting the content of a knowledge goal in the context of the dynamics of the connectionist network has been proposed. This approach may help to accommodate the goal of improving learner's learning process by matching the lesson with a list of stated goals. In each of the goal–based lessons, examples related to the learner's field of experience or explanations in the context with which learner is almost familiar could be provided. For example, learners taking an introductory course might be interested in how it relates to their major course.

In the section about the learner assessment, neuro–fuzzy synergism has been applied to collect information and evaluate what the learner knows about the concepts of the domain. This information usually includes vague, ambiguous, incomplete and some times even contradictory terms; therefore a fuzzy logic–based assessment procedure has been developed. The proposed approach depends on the designer's ability to analyse the cognitive domain suitably, define fuzzy variables and appropriate membership functions for their fuzzy sets by co–operating with experts–tutors, and relate learner's response with appropriate knowledge and cognitive characteristics. The data produced by the learner assessment are only indicative of what the learner may know about the domain concepts. For example, data from the learner assessment might indicate that a learner has little (*Rather Sufficient*) or no knowledge (*Extremely Insufficient*) about concepts of a knowledge goal. These data should be interpreted as meaning that there is only a possibility of this observation being true. Nevertheless, the fuzziness associated with the assessment of the level of understanding seems to be handled well by the connectionist network that makes "decisions" about what concepts should be presented to the learner. Both the output of the network and the results of the assessment procedure can be used by the instructional component to decide when and what kind of educational material to give to the learner. The paper ended with a partial test of the system. A lesson was provided to a learner and the operation of the models was demonstrated. Additional examples, as well as examples of other knowledge goals have been reported in Papanikolaou et al. (2000).

The applicability of the proposed approach can be further extended by exploiting the training and generalisation capabilities of the artificial neural networks to extract information from learner performance parameters, such as the points scored, the time taken etc., to predict trends in performance. These performance patterns could help in deriving instructional strategies to optimally select the appropriate teaching method for each learner.

References

[1] Anjaneyuly K. (1997) Concept level modeling on the WWW. Brusilovsky P., Nakabayashi K., & Ritter S. (eds.) *Proceedings of the Workshop on Intelligent Educational Systems on the World Wide Web*, the 8th International Conference on Artificial Intelligence in Education, Kobe, Japan. Osaka: ISIR.

[2] Bennett N., Desforges C., Cockburn A. & Wilkinson B. (1984) *The Quality of Pupil Learning Experiences*. London: Lawrence Erlbaum Associates.

[3] Bertles K. (1994) A dynamic view on cognitive student modeling in computer programming. *J. of Artificial Intelligence in Education*, 5, 1.

[4] Brulikovsky P.(1996) Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6, 2–3, p. 87–129.

[5] Burns H. L. & Capps, C. G. (1988) Foundations of intelligent tutoring systems: an Introduction. Polson M. C. & Richardson J. J. (eds.) *Foundations of Intelligent Tutoring Systems*. London: Lawrence Erlbaum, p. 1–19.

[6] Mc Cormack C., & Jones D. (1997) *Building a Web–based Education System*. New York: Wiley Computer Publishing.

[7] Davidson K. (1999) *Education in the internet – Linking theory to reality*. Available on–line: http://www.oise.on.ca/~ kdavidson/cons.html, accessed December 1999.

[8] Dijkstra S., Krammer H. M. & Van Merrienboer, J. G. (eds.) (1992) *Instructional Models in Computer–based Learning Environments*, NATO ASI Series F, Vol. 104. New York: Springer–Verlag.

[9] DIUA (1999) *Introduction to Computer Science and Telecommunications*, Distance Learning Course, Department of Informatics, University of Athens. URL: http://hermes.di.uoa.gr

[10] Ellis R. (1994) Individual learner differences. Ellis, R. *The Study of Second Language Acquisition*. Oxford: Oxford University Press, p. 471–527.

[11] Embenson S. (1990) Diagnostic testing by measuring learning processes: psychometric considerations for dynamic testing. Frederiksen N. (ed.) *Diagnostic Monitoring of Skill and Knowledge Acquisition*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

[12] Grigoriadou M., Magoulas G. D. & Panagiotou M. (1999a) A hybrid decision making model for learner evaluation in intelligent tutoring systems. *Proceedings of the 5th International Conference of the Decision Sciences Institute*, Athens, Greece, p. 195–197.

[13] Grigoriadou M., Papanikolaou K., Cotronis Y., Velentzas Ch. & Filokyprou G. (1999b) Designing and implementing a web based course. Chapman G. M. (ed.) *Proceedings of International Conference of Computer based Learning in Science*, Enschede, Netherlands. Czech Republic: Pedagogical Faculty of University of Ostrava, p. H5.

[14] Kasabov N. (1996) *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, MA: MIT Press.

[15] Krathwohl D. R., Bloom B. S. & Masia B. B. (1964) *Taxonomy of Educational Objectives: the Classification of Educational Goals (Handbook II): Affective Domain.* New York: David McKay Co., Inc.

[16] Kuhme T. (1993) User–centered approach to adaptive user–interfaces. *Knowledge-Based Systems*, 6, 4, p. 239–248.

[17] Magoulas G. D., Vrahatis M. N. & Androulakis G. S. (1997) Effective back–propagation training with variable stepsize. *Neural Networks*, 10, p. 69–82.

[18] Magoulas G. D., Papanikolaou K. & Grigoriadou M. (1999) Towards a computationally intelligent lesson adaptation for a distance learning course. *Proceedings of the 11th International Conference on Tools with Artificial Intelligence* , Chicago, USA. Los Alamitos: IEEE Press, p. 5–12.

[19] Michel A., Si Y. & Yen G. (1991) Analysis and synthesis of a class of discrete–time neural networks described on hypercubes. *IEEE Transactions on Neural Networks*, 2, p. 2–46.

[20] Michos S. E., Magoulas G. D. & Fakotakis N. (1995) A hybrid knowledge representation model in a natural language interface to MS–DOS. *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*, Chicago, USA. Los Alamitos: IEEE Press, p. 480–483.

[21] Nkambou R. (1999) Managing inference process in student modeling for intelligent tutoring systems. *Proceedings of the 11th International Conference on Tools with Artificial Intelligence*, Chicago, USA. Los Alamitos: IEEE Press, p. 19–23.

[22] Panagiotou M. & Grigoriadou M. (1995) An application of fuzzy logic to student modeling. *Proceedings of the IFIP World Conference on Computers in Education*, Birmigham, UK.

[23] Papanikolaou K. A., Magoulas G. D. & Grigoriadou M. (1999) A connectionist approach for adaptive lesson presentation in a distance learning course. *Proceedings of International Joint Conference on Neural Networks*, Washington, USA. CD-ROM Proceedings, IEEE Catalog Number: 99CH36339C, paper # 680.

[24] Papanikolaou K. A., Magoulas G. D. & Grigoriadou M. (2000) Computational intelligence in adaptive educational hypermedia. *Proceedings of International Joint Conference on Neural Networks*, Como, Italy. Los Alamitos: IEEE Press, p. VI-629.

[25] Reusser K. (1996) From cognitive modeling to the design of pedagogical tools. Vosniadou S., De Corte E., Glaser R. & Mandl H. (eds.) *International Perspectives on the Design of Technology–Supported Learning Environments*. New York: Lawrence Erlbaum Associates, p. 81–103.

[26] Saaty T. L. (1978) Exploring the interface between hierarchies, multiple objectives and fuzzy sets. *Fuzzy Sets and Systems*, 1, p. 57–68.

[27] Stathacopoulou R., Magoulas G. D. & Grigoriadou M. (1999) Neural network–based fuzzy modeling of the student in intelligent tutoring systems. *Proceedings of the International Joint Conference on Neural Networks*, Washington USA. CD-ROM Proceedings, IEEE Catalog Number: 99CH36339C, paper # 679.

[28] Stephanidis C., Paramythis A., Karagiannidis C. & Savidis A. (1997) Supporting interface adaptation: the AVANTI Web–browser. Stephanidis C. & Carbonell N. (eds.) *Proceedings 3rd ERCIM Workshop on User Interfaces for All*, Obernai, France.

[29] Vassileva J. (1995) Reactive instructional planning to support interacting teaching strategies. *Proceedings of the 7th World Conference on AI and Education*, Washington, USA. Charlottesville: AACE, p. 334-342.

[30] Vassileva J. (1997) Dynamic course generation on the WWW. Brusilovsky P., Nakabayashi K. & Ritter S. (eds.) *Proceedings of the 8th International Conference Artificial Intelligence in Education*, Kobe, Japan. Osaka: ISIR, p. 498-505.

[31] Vosniadou S., De Corte E., Glaser R. & Mandl H. (Eds.) (1996) *International Perspectives on the Design of Technology–Supported Learning Environments*. New York: Lawrence Erlbaum Associates Publishers.

[32] Weber G. & Specht M. (1997) User modeling and adaptive navigation support in WWW–based tutoring systems. *Proceedings of the 6th International Conference on User Modeling*, Chia Laguna, Sardinia, Italy. Also in Jameson A., Paris C. & Tasso C. (eds) *User Modeling*. Wien: Springer–Verlag, p. 289–300.

[33] Wenger, E. (1987) *AI and Tutoring Systems: Computational and Cognitive Approaches to the Communication Knowledge*. California: M. Kaufmann Publishers, Inc, p. 20.

Risk logical and probabilistic models in business and identification of risk models

E.D. Solojentsev, V.V. Karasev
Institute of the Problems of Mechanical Engineering
Russian Academy of Sciences
Bolshoy pr. 61, St.Petersburg, 199178, Russia
Phone: 7 (812) 321 47 66, Fax: 7 (812) 321 47 71
E-mail: sol@sapr.ipme.ru

Keywords: logic, probability, risk, model, identification, business

Received: December 12, 1999

The apparatus of logical and probabilistic (LP) simulation, not very popular among mathematicians and economists, is developed and used to study risk in business. The logical operations (AND, OR, NOT) are applied to the initiating events (instead of the traditional arithmetical addition of values). We present statistical data about risk objects as a table of "object-signs". A risk object is described by a large number of signs, and every sign has up to 10 gradations. Events-signs are connected logically and event-gradations are treated as groups of incompatible events (GIE). The events are given clear probabilistic sense. Risk LP-models for banks, business, insurance and quality are built as associative models on the basis of common sense and are considered as the hypotheses

1 Introduction

We are developing a new theory of the numerical evaluation of risk, as well as risk analysis and management. This theory is based on a logical and probabilistic (LP) approach [1], which is often applied in engineering for the evaluation and analysis of reliability and safety of structurally complex systems [2,3]. The risk script or graph allows to allocate basic elements and their logic interrelation. Events, pertinent to the risk analysis, have the logical connections AND, OR, NOT. The risk logic (L) model can have cycles. The L-model is transformed by the method of orthogonalization into the risk probabilistic (P) model.

In business and quality control the risk is a usual and mass phenomenon. The risk object is described by a large number of signs, every sign has some gradations [4,5]. Signs and gradations connect with casual events, which lead to a failure. Events-signs have logical connections and events-gradations for each event-sign form groups of incompatible events (GIE). The final event for separate tasks of can be considered as deviation from the allowable rate of profit.

The accepted description of objects and events allows to reject the standard analytical laws of distribution of probabilities, usually accepted in the probabilistic theory and accuracy theory, and to use probabilities of events-gradations and events-signs as the discrete non-parametrical distributions, typical for logic.

The analysis of risk in case of a complex task includes [1]: the numerical evaluation of the object's failure risk (probability), the object's classification on risk, the analysis of contributions of initiating events in the object's risk, the price assignment for the risk, the analysis of contributions of initiating events to the average risk of a set of objects. The result of such analysis should be a risk management strategy.

2 The theory of the risk LP-simulation with GIE

Analytical expression for probabilities of risk P-model. Binary logical variable X_j is equal to 1 with probability P_j, if the j-th sign leads to failure, otherwise X_j is equal to 0 with probability $Q_j = 1 - P_j$. Binary vector $X(i) = (X_1, X_2, \ldots, X_n)$ corresponds to the vector of state or vectot of description of the i-th object. Logic variable X_{jr} corresponds to the r-th gradation of the j-th sign. The measured vector $X(j)$ for any specific object will then be defined by the gradationes of its signs.

For example, after orthogonalisation of L-model of credit risk [1]

$$Y = X_1 \vee X_2 \cdots \vee X_j \ldots X_n \tag{1}$$

we have the risk P-model

$$P = P_1 + P_2 * Q_1 + P_3 * Q_1 * Q_2 + \ldots . \tag{2}$$

In the common form, L-function of the final logical variable Y, accepts value $Y = 1$ for bad object and $Y = 0$ for good object. Let us write the logic function of failure for the object as

$$Y = \psi\,(X_1, X_2, \ldots, X_n) = \Psi(X) \tag{3}$$

and the probability function of failure for the object defined by vector $X(i)$ as

$$P_i = P\{Y = 1|X(i)\} = \phi\,(P_1, \dots, P_j, \dots, P_n). \quad (4)$$

For each event-gradation in GIE, let us consider three probabilities: W_{jr} is the relative frequence of gradations in objects of the "object-signs" table, $P1_{jr}$ is the probability of the event-gradation in GIE, $P1_{jr}$ is the probability of the event-gradation to be substituted into (4) instead of probability P_j. We define these probabilities to be:

$$W_{jr} = P\{X_{jr} = 1\}, \ \sum_{r=1}^{Nj} W_{jr} = 1; \quad (5)$$

$$P1_{jr} = P\{X_{jr} = 1|X_j = 1\}; \ \sum_{r=1}^{Nj} P1_{jr} = 1. \quad (6)$$

$$P_{jr} = P\{X_j = 1|X_{jr} = 1\}. \quad (7).$$

Here and below we will everywhere use indexes $j = 1, n$; $r = 1, N_j$ for formulae, where n is the number of events-gradations, N_j is the number of events-gradations in the j-th GIE.

For training of risk LP-model we will use the average values of the probabilities W_{jr}, P_{jr} and $P1_{jr}$ in the GIE. They are:

$$W_{jm} = 1/N_j;$$

$$P_{jm} = \sum_{r=1}^{Nj} P_{jr}\, W_{jr}; \quad (8)$$

$$P1_{jm} = \sum_{r=1}^{Nj} P1_{jr}\, W_{jr}. \quad (9)$$

Average apriori risk P_{av} for objects and computable risk of objects P_m equal to

$$P_{av} = N_b/N; \ P_m = (\sum_{i=1}^{N} P_i)/N, \quad (10)$$

where N is the gross number of objects in the table and N_b is the number of "bad" objects in the table.

Calculated admissible risk P_{ad} separates the objects into good and bad: if $P_i > P_{ad}$, an object is bad; if $P_i < P_{ad}$, an object is good.

Connection between probabilities P_{jr} and $P1_{jr}$. The risk of an object P_i is calculated using (4) by replacing P_{jr} with P_j. Let us estimate probabilities P_{jr} during the process of algorithmic iterative training (identification) of the P-model of risk using the data from the "object-signs" table.

P-model of risk can be trained under the GIE assuption, or without it. In these cases the connection between probabilities P_{jr} and $P1_{jr}$ is determined in different ways during the algorithmic iterative training of risk P-model.

A trivial relationship exists between P_{jr} and $P1_{jr}$ if GIE is ignored during training:

$$P1_{jr} = P_{jr} \,/ \sum_{r=1}^{Nj} P_{jr}. \quad (11)$$

This means that in the optimisation problem of algorithmic training of the P-model of risk based on the data from the "object-signs" table probabilities P_{jr} are estimated directly, while probabilities $P1_{jr}$ are computed using (11) for information. During such training the number of estimated independent probabilities P_{jr} is equal to: $N_{indep} = \sum_{j=1}^{n} N_j$.

In this and other schemes of training the probabilities P_{jr} must be corrected on each step of iterative training of the LP risk model by applying the average computed risk P_m and the average risk P_{av} in table (10):

$$P_{jr} = P_{jr} * (P_{av}/P_m). \quad (12)$$

If GIE is taken into account during training, then it is necessary to begin with the determination of probabilities $P1_{jr}$ satisfying (6), and then move from $P1_{jr}$ to P_{jr}. During such training the number of estimated independent probabilities P_{jr} is equal to: $N_{indep} = \sum_{j=1}^{n} N_j - n$ and the solutions P_{jr} are more stable because of a smaller number of independent evaluated probabilities P_{jr}. Let us discuss possible relationships between P_{jr} and $P1_{jr}$ during the training of P-model of risk with GIE. The relationship between probabilities P_{jr} and $P1_{jr}$ in each step of iterative training of the P-model of risk based on statistical data can be built based on (5-7) by utilizing a known rule for conditional probabilities:

$$P_{jr} = \frac{P1_{jr} * P_{jm}}{W_{jr}}. \quad (13)$$

If the table "object-signs" contained all possible distinct objects, whose number is approximately N_{jm}^{n}, where N_{jm} is the average number of gradations for signs, then the frequencies for each gradation in GIE would have been identical and equal to the average values of frequencies W_{jm} in GIE (10). In reality, the number of objects in the table is always substantially less than that, and thus one can encounter gradations with frequencies $W_{jr} = 0$ or negligibly small. This renders the Bayes formula (13) useless since W_{jr} is in the denominator of (13). Therefore, we have to make simplifications and receive another formulae for connection of probabilities P_{jr} and $P1_{jr}$.

In the first formula, the frequency of gradation W_{jr} was replaced by an average frequency of gradations W_{jm} in GIE. By increasing the number of objects in table the frequency W_{jr} tries to W_{jm}:

$$P_{jr} = \frac{P1_{jr} * P_{jm}}{W_{jm}}. \quad (14)$$

In the second formula the average frequency of gradations W_{jm} in GIE was substituted instead of W_{jr} for calculation Pjm:

$$P_{jr} = P1_{jr} * \sum_{r=1}^{Nj} P_{jr}. \qquad (15)$$

In the third formula we receive connection of probabilities P_{jr} and $P1_{jr}$ from the average values of their probabilities P_{jm} and $P1_{jm}$ in the table, calculated by formulae (8,9):

$$P_{jr} = \frac{P1_{jr} * P_{jm}}{P1_{jm}}. \qquad (16)$$

The replacement of the average frequency W_{jm} with the average probability $P1_{jm}$ for GIE can be considered a "soft" averaging, because the sum of frequencies W_{jr} and probabilities $P1_{jr}$ in GIE are equal to 1 (5,6). Applying formulas (11,15,16), allows us to develop stable algorithms for iterative training of the risk P-model. Accuracy and stability of the risk LP-model depends on formulas (11,15,16). We name K_j the factor of the probabilities connection $P_{jr} = K_j * P1_{jr}$. For example, for formula (16) it is equal:

$$K_j = P_{jm}/P1_{jm}.$$

Naturally, we could receive the formula (16) immediately by considering K_j=const. Formula (16) for connection of probabilities P_{jr} and $P1_{jr}$ will be used in case of limited statistical data.

3 Identification of the risk P-model

Identification of the risk P-model boils down to the determination of optimal probabilities $P_{jr},\ r = 1, N_j;\ j = 1, n$ [1], corresponding to the events-gradations. Let us formulate the identification (training) problem for a risk P-model.

Available data: the "object-signs" table with N_g good and N_b objects and the risk P-model (4);

Expected results: to determine probabilities $P_{jr},\ r = 1, N_j;\ j = 1, n$ for events-gradations and acceptable risk P_{ad}, dividing objects onto good and bad based on their risk.

We need: to maximize the cost function, which is the number of correctly classified objects:

$$F = N_{bs} + N_{gs} = MAX, \qquad (17)$$

where $N_{bs} and N_{gs}$ are the numbers of objects classified as bad and good using both the statistic and the risk P-model (4) (both estimates should coincide). Note that from (17) it follows that errors or accuracy indicators of the P-model of risk in the classification of good E_g and bad E_b objects and in the classification of the whole set E_m are equal:

$$E_g = (N_g - N_{gs})/N_g;\quad E_b = (N_b - N_{bs})/N_b;$$

$$E_m = (N - F)/N. \qquad (18)$$

Assumed restrictions:

1) probabilities $P_{jr} and P1_{jr},\ r = 1, N_j;\ j = 1, n$ must be larger than 0 and less than 1;

2) the average risks of objects P_m based on the risk P-model and on table P_{av} must be equal;

3) the admissible risk P_{ad} must be determined with the given ratio of incorrectly classified good and bad objects:

$$E_{gb} = (N_g - N_{gs})/(N_b - N_{bs}). \qquad (19)$$

In the identification problem of risk P-model formulated above, the cost function F depends on a large number of positive real probabilities P_{jr} (up to 94 in one of the risk problem that we considered). We should note that, because of the stepwise nature of the cost function, its derivatives with respect to P_{jr} are infinite, moreover, it has local extrema in the region of optimization. P_{jr} cannot be given positive or negative increments arbitrarily, since this will lead to a change in the average risk.

We construct an iterative algorithm for generating $P1_{jr}$ maximizing F as follows. Choose appropriate values for the computed number of good N_{gc} and bad $N_{bc} = N - N_{gc}$ objects. These values for procedure described below are constant. Let us note numbers of optimisation $0, 1, \ldots, v, \ldots, N_{opt}$.

* Initial step. Give initial values $P_{jr}^{(0)}$ and $P1_{jr}^{(0)}$ for $j \in S$ and $r \in G_j$; compute $P_{jm}^{(0)}$ and $P1_{jm}^{(0)}$ based on (8) and (9); assign a small value to the cost function, say, $F = P_{av} * N$ and compute $K_j = P_{jm}^0/P1_{jm}^0$.

* Subsequent steps. We will optimize the cost function iteratively for steps until the cost function increases N_{opt} times, which cannot be more than $N/2$. Generate small increments $dP1_{jr}^v, j \in S,\ r \in G_j$, and compute new normalized values $P1_{jr}^v$ from

$$P1_{jr}^{(v)} = (P1_{jr} + dP_{jr}^{(v)})/(\sum_{r=1}^{Nj}[P1_{jr} + dP1_{jr}]). \qquad (20)$$

Then compute $P_{jr}^{(v)}$ from (16) and $P_i^{(v)}$ using new $P_{jr}^{(v)}$. Calculate the average object risk $P_m^{(v)}$ from (10) and average risks $P1_{jm}^v$ and P_{jm}^v from (8,9). Determine P_{ad} from $P_i^{(v)}$ so as to attain N_{gc} and N_{bc}. Compute $F^{(v)}$ based on P_{ad}^v and $P_i^{(v)}, i = 1, \ldots, N$. If $F^{(v)} > F_{max}$, then

$$F_{max} = F^{(v)},\quad P1_{jr} = P1_{jr}^{(v)},\quad P_{jr} = P_{jr}^{(v)} \qquad (21)$$

and adjust P_{jr} using (12). If the cost function does not strictly increase after the chosen number of trials N_{mc}, then F_{max} is reduced by 2-4 units and optimization continues.

The described identification algorithm has the degree of complexity equal to n^3, where n is the length of entry into the identification problem, equal to the number of signs needed for describing the risk object. This algorithm also requires a minimum amount of PC memory, which is important in business applications with abundant statistical data.

For calculation $dP1_{jr}$ we use the following formula:

$$dP1_{jr} = k_1 * (k_2 - N_t) * k_3, \qquad (22)$$

where: k_1 is the coefficient of identification speed; k_2, N_t are the maximum possible and the current number of optimisation steps for the cost function; k_3 is a random number from $[-1, 1]$. New values $P1_{jr}$ and P_{jr} obtained with $F > F_{max}$ are considered optimal and stored. If certain $P1_{jr}$ become negative or exceed 1, we set them to 0 or 1 respectively. The convergence of the optimization method is guaranteed by the second factor in (22), which approaches 0 as N_t grows.

4 The risk measure and risk analysis methods

The objects risks can be submitted not only in the interval $[P_{min}, P_{max}]$, where P_{min}, P_{max} are the minimum and maximum risks of objects from the "object-signs" table, but also on [0,1]. For this purpose, let us compute the number of objects N_{ad} and N_i with risks less than P_{ad} and P_i and compute the following measures of goodness for the i-th object:

1) the relative number of objects whose risk is less than P_i or greater than P_i, where:

$$a_i = N_i/N; \; b_i = 1 - a_i; \qquad (23)$$

2) the relative number of good c_i and bad e_i objects with risk exceeding the risk of i-th object among good and bad objects:

$$c_i = (N_{ad} - N_i)/N_{ad}; \; e_i = (N_i - N_{ad})/(N - N_{ad}). \qquad (24)$$

It is obvious also the there is a natural measure of how good or bad the i-th object is. It is the distance between the allowable risk P_{ad} and the risk P_i for i-th object:

$$d_i = ||p_i - P_{ad}||. \qquad (25)$$

The introduced measures can be used to calculate the price of risk.

Suppose that a P-model of risk has been trained and the probabilities for events-gradations P_{jr} are known. For the purpose of analysis, let us define the contributions of events-signs and events-gradations in the object's risk, the failure average risk of a set of objects. Having these contributions it is possible to accept the decisions on risk management for both concrete object and set of objects (for example, to operate by credit risk in banks).

Contributions in risk can be achieved computationally using the difference between the values of given probabilities in the optimal regime and the values obtained through assigning zeros to the probabilities corresponding to the events-gradations. These contributions are:

dP_j, $j = 1, n$ are contributions of signs (all gradations) in the object's risk;

dP_{jm}, $j = 1, n$ are contributions of signs (all gradations) in the average risk P_m of a set of objects;

dP_{jrm}, $j = 1, n$; $r = 1, N_j$ are contributions of gradations to the average risk P_m of a set of objects.

Similarly we can calculate the double $(dP_{jrm} \wedge dP_{krm} : j, k = 1, n; \; j \neq k)$ and threefold contributions of gradations $(dP_{jrm} \wedge dP_{krm} \wedge dP_{erm} : j, k, e = 1, n : j \neq k \neq e)$ in the average risk P_m.

Let, probabilities of the events-gradations $P_{jr}, j = 1, n; \; r = 1, N_j$ be known, that is they are received by training of risk LP-model. For the analysis of accuracy of risk LP-model and decision of tasks of structural identification of the risk LP-models, we calculate the individual and group contributions of the events-signs and events-gradations in cost function F_{max} of the identification problem.

We calculate the individual and group contributions of the events-signs and the events-gradations using a computer. We compute the difference between the maximal value of the cost function, optimal trained risk LP-model, and the cost function, received by zero values of the appropriate probabilities of events-gradations. Thus, we can calculate:

dF_j is an individual contribution of the $j - th$ event-sign to cost function F_{max};

dF_{jr} is an individual contribution of the $jr - th$ event-gradation to cost function F_{max};

Mistakes of classification from events-gradations may be calculated too using analogy with formula (18).

5 Global extremum and the parameters of optimisation

It is considered, that a training method, which uses random search, warrants locating the global extremum for the continuous cost function [6]. However, in case of an integer cost function, as it is in our case, it is not possible to give any guarantee.

We can obtain a decision $P1_{jr}$ and P_{jr}, using algorithm [6], in one of the local minimum. The decision P_{jr} and $P1_{jr}, j = 1, n; r = 1, N_j$ cannot go out the local minimum, beginning with certain number of optimisation, because the $dP1_{jr}$ have small values.

In different local minima the decision $P1_{jr}$ and P_{jr} is different and risks of N credits have different distribution. Each distribution may be described by parameter $dP_c = P_{i\,max} - P_{i\,min}$, where $P_{i\,max}, P_{i\,min}$ are the maximal and minimal credit risks among N objects. We can control by parameter dP_c, changing of coefficients k_1, k_2, k_3 in formula (22). We made the charts of changing F_{max} in function of parameter dP_c and chose the optimal value of this parameter.

We built also, using a set of $N = 1000$ credits of individuals, the dependence for the cost function on the number of good credits N_{gc}. It allowed to choose the calculated optimal number of good credits N_{gc}.

Credits have non-coincided sets of gradations, that is the classes of good and bad credits do not cross. Using results of research, we set the parameter dP_c is equal about 0.1. In average credits differ by value $dP_{ad} = dP_c/N = 0.1/1000 = 0.0001$. Interval $[P_m, P_{ad}]$ has the density of risk distribution more in $M = 100$ times. Credits, which are neighbouring by risk, should differ from each to other in interval $[P_m, P_{ad}]$ by value:

$$dP_{ad} = dP_{ad}/M = 0.0001/100 = 0.000001. \qquad (26)$$

The algorithm of risk model training was constructed under the given number of good credits N_{gc} and iterative selection of allowable risk P_{ad}. Let, set in first approximation allowable risk is equal to average risk P_m. Then we compute the calculated number of good credits N_g. If N_g is not equal N_{gc}, we shall increase allowable risk: $P_{ad} = P_{ad} + dP_{ad}$. After that, we compute N_g again etc. until condition $N_{gc} = N_g$ is executed.

Now it is clear, that the dP_{ad} value should not contain risks for more than one credit. Calculated accuracy of allowable risk is equal to $dP_{ad} = 0.000001$. Now we can evaluate the accuracy of calculation of probabilities $P_{jr}, j = 1, n; r = 1, N_j$. The accuracy of calculation P_{jr} is equal to : $dP_{jr} = dP_{ad}/20 = 0.00000005$. It is necessary to know the accuracy of dP_{jr} for choice of training speed coefficient k_1 and printing.

LP-technique for risk valuation was approved for a set of 1000 credits of individuals. As another techniques [5], the risk P-model was trained by training the sample of 700 credits and was checked by the sample of 300 credits. The training file has 500 good and 200 bad credits. After training of risk LP-model, probabilities $P_{jr}, P1_{jr}$ and the allowable risk P_{ad} are used for evaluation of accuracy of LP-technique by control sample. The results of testing were very good.

6 Evaluation of accuracy of LP-models

The analysis of the results. For risk model training were used the following numbers of good credits (table 1): $N_{gc} = 550, 580, 610, 650, 700, 750, 800$. We computed: risks of 1000 credits, maximum of object function F_{max} and its components N_{gs} and N_{bs}, mistakes in recognition of credits E_b, E_g, E_m, the admissible credit risk P_{ad}; the ratio of numbers of not recognised bad b/g and good g/b credits E_{gb}. The mistakes of risk LP-model for optimal variant $N_{gc} = 610$ are $E_b = 0.167, E_g = 0.201, E_m = 0.191$ and for $N_{gc} = 650$ are $E_b = 0.207, E_g = 0.161, E_m = 0.175$. Optimal decision is established by admissible value of ratio E_{gb}. Optimal decisions by $N_{gc} = 610$ and $N_{gc} = 650$ have symmetrically different mistakes of evaluations E_b, E_g.

Table 1. The results of investigations of credit risk

N_{gc}	550	580	610	650	700	750	800
F_{max}	767	787	809	825	829	831	819
N_{gs}	508	533	559	587	614	640	659
N_{bs}	259	254	250	238	215	191	160
E_g	.274	.238	.201	.161	.123	.10	.58
E_b	.137	.153	.167	.207	.283	.363	.466
E_m	.233	.213	.191	.175	.171	.169	.181
P_{ad}	.301	.301	.302	.302	.303	.304	.305
Egb	4.55	3.53	2.65	1.77	1.0	.51	.213

The credit orders are considered, using of the trained risk model. Bank estimates the risk of the new credit P_i, and if it will exceed the admissible risk P_{ad}, established by training, then the credit is estimated as bad. The probability of mistakes in evaluation of credits in this time will equal to $E_b = 0.167, E_g = 0.201$ only. It is less than $P_{av} = 0.3$ in statistics. Such, bank can reduce per cent for credit and assign it in dependence on credit risk P_i. If we get new statistics about credits later, then the average risk will be about $P_m = 0.2$.

Comparison with western techniques. The comparison of different risk evaluation methods should be based on the same indicators of accuracy of object classification (18) and carried out on the same statistical data. Let us illustrate such a comparrison based on example [5].

The accuracy of an LP-model of risk was tested on standard statistical data oncluding 1000 credits, out of wich 700 were good and 300 were bad. This data was used to evaluate the accuracy of nearly 10 different classification method [5] based on linear (LDA) and quadratic (QDA) discriminant analysis, cluster analysis (CA) and neural networks (NN).

The LP-models of credit risk is presented by formulas (1), (2). The failure occurs when any one, two,..., or all initializing events take place. It has 20 events-signs (corresponding to GIE) and 96 events-gradations.

During the training of the LP-model of risk (11) (without GIE) and (16) (with GIE) where used to tie P_{jr} and $P1_{jr}$.

The results of LP-method were calculated for given number of good credits $N_{gc} = 610$ and bad credits $N_{bc} = 390$, but in bank statistical data we have $N_g = 700$ and $N_b = 300$. We made it because the losses due to mistake in classification of bad credit is larger than one due to mistake in classification of good credit. The ratio of losses is equal to $E_{gb} = N_{b/g} N_{g/b} = 2.6$.

We present some other results also. In Var.1 we estimated 94 probabilities P_{jr} (without GIE). In Var.2 we estimated only 74 probabilities P_{jr} (with GIE). The object function in Var.1 is equal to $F_{max} = 809$ and in Var.2 is equal to $F_{max} = 801$.

The results of accuracy comparison of different methods based on the same statistical data (Table 2) show that the LP-model of risk is almost 1.5 times more accurate than other classification methods.

The introduction of the price of credit risk can be jusstified if we build the graph of risks for 1000 credits after sorting objects by risks. Approximately 15 % of credits are very good, while 15 % of credits are very bad, which naturally leads us to believe that the price of credit should be dependent on its risk.

Table 2. Parameters of accuracy of classification of credits by different methods

Used method	For bad objects, E_b	For good objects, E_g	Average mistake, E_m
LDA Resubstitution	0.26	0.279	0.273
LDA Leaving-one-out	0.287	0.291	0.29
QDA Resubstitution	0.183	0.283	0.253
QDA Leaving-one-out	0.283	0.34	0.323
CART	0.277	0.289	0.285
Neural Network NN1	0.38	0.24	0.282
Neural Network NN2	0.24	0.312	0.29
LP-model (Var.1)	0.167	0.201	0.191
LP-model (Var.2)	0.176	0.204	0.196

In developed software three tasks are decided: the quantitative assessment of credit risk and its analysis; the analysis of credit works of bank; the training of logical and probabilistic risk models by statistical data of bank about credit defaults.

7 Stability of classification of objects

Depending on the method and training parameters, one of the two different LP-models may classify an object as good, while the other may classify it as bad. Consider the following example.

A stability estimate for a P-model of risk was carried out using the data from Example 1 and the method of pairwise of pairwise comparison of different versions of the solution in credit classification. The comparison was based on the number of inconsistencies of the estimates of good n_g, bad n_b and unrecognized n_{gb} objects.

During the training of the P-model of risk with GIE ($N_{gc} = 610; N_{bc} = 390; Nind = 76$) three different solutions were obtained: $1 - F_{max} = 794, 2 - F_{max} = 796$ and 3 - $F_{max} = 798$. The results of this comparison are presented in Table.3.

Table 3. Pairwise comparisons of different training procedures on the stability of solutions (with GIE)

Variants	Number of consistencies by "good", n_g	Number of consistencies by "bad", n_b	Number of inconsistencies, n_{gb}
1 2	601	381	18
2 3	601	381	18
1 3	602	382	16

The stability indicator for risk LP-models with GIE for these three procedures is calculated as

$$K_{s1} = (18 + 18 + 16)/(1000 * 3) = 0.018.$$

During the training of the P-model of risk without GIE ($N_{gc} = 610; N_{bc} = 390; Nind = 96$) four different solutions were obtained: $1 - F_{max} = 805, 2 - F_{max} = 807$, $3 - F_{max} = 809$, $4 - F_{max} = 811$. The results of comparisons are presented in Table. 4. The stability indicator for LP-models of risk without GIE for these four procedures is calculated as

$$K_{s2} = (160 + 90 + 136 + 120 + 152 + 100)/6000 = 0.13.$$

Table 4. Pairwise comparisons of different training procedures on the stability of solutions (without GIE)

Variants	Number of consistenciis by "good", n_g	Number of consistencies by "bad", n_b	Number of inconsistencies, n_{gb}
1 2	530	310	160
1 3	565	345	90
2 3	542	322	136
1 4	550	330	120
2 4	534	314	152
3 4	560	340	100

The ratio of the stability indicator for risk LP-models with GIE and to that without GIE is:

$$K_{s2}/K_{s1} = 0.128/0.018 = 7.1.$$

The obtained results can be generalized on the instability of models of risk based on neural networks [5], where a large number of weights of net links is introduced without any restrictions.

8 Conclusion

In this paper we formulated the complex problem of failure risk in business. We developed a theory of LP-estimation and analysis of risk with GIE. This theory utilizes two types of conditional probabilities for events-gradations: probabilities $P1_{jr}$ for GIE and probabilities P_{jr} used in the formula for a B-models of risk. We formulated the proplem of LP-model identification based on statistical data and designed an algorithm for its solution. The risk LP-models showed almost twice more accuracy and seven times more stability in classification of credits, than methods and software prevailing on western market.

We have executed some investigations on training of risk models: the choice of average risk, ways of overcoming of impasses, control of time of training, selection of training and checking sets, providing of structural identification of risk L-model.

We have developed LP-models of risk in banks: credit risk of individuals, credit risk of companies, analysis of bank credit activity, control of state and development of bank by risk criterion. We have considered also failure LP-models for frauds in business: manager, office worker, with investment.

In our research we have described risk LP-models of quality loss. We have received results of research of quality for product because of failure of components, for product because of discrepancy of manufacturing, for human-machine system, for organizational system.

The software for risk simulation, analysis and identification of risk models are executed in Delphi-4, Visual C ++ and Java. The more detailed information about risk LP-models, software and authors is given on Web Site: http: // www.ipme.ru/ipme/labs/iisad/sapr1.htm

References

[1] Solojentsev E.D., Karassev V.V., Solojentsev V.E. (1999) *Risk logic and probabilistic models in banks, business and quality / By edition E.Solojentsev* . St.Petersburg, Nauka, 120 pp.

[2] Ryabinin I.A., Cherkesov G.N. (1981) *Logic-probabilistic methods of research of reliability of structure-complex system* , Moscow, Radio and communication, 238 pp.

[3] Ryabinin I.A. (1994) A suggestion of a new measure of system Components importance by means of Boolean difference. *Microelectronics and Reliability.* Vol.34, N 4, pp.603-613.

[4] Geisser S. (1976) Recognition: Classification and Discrimination. Linear issues. In: Classification and Clusterisation (Ed. by J. Van Ryzin). *Proceedings of Advanced Seminar, Conducted by the Mathematics Research Center* . The University of Wisconsin at Madison, May 3-5).

[5] Seitz J., Stickel E. (1992) Consumer Loan Analysis Using Neural Network / Adaptive Intelligent Systems. *Proceed. Of the Bankai Workshop, Brussels* , 14-19 October, 1992.- p.177-189.

[6] Wasserman Philip D.(1990) *Neural Computing Theory and Practice.* ANSA Research, Inc. VAN NOSTRARD REIHOLD New York, 1990.

[7] Solojentsev E. (1997) New areas and tasks of simulation and analyses of risk by logic-probabilistic method *International Conference on Informatics and Control (ICIandC'97): Proceed.* June 9-13, St.Petersburg, Russia, p.1109-1117.

[8] Solojentsev E.D., Karassev V.V., Solojentsev V.E.(1996) *Logic-probabilistic valuation of bank risks and fraud in business.*- St.Petersburg.- Politekhnika, 60 pp.

[9] Solojentsev E.D., Karassev V.V.(1999) The Logic and probabilistic method of assessment of credit risk and identification of risk model *International ICSC Congress. Computational Intelligence: Method and Applications (CIMA'99) Proceed.*, Rochester, USA, June 22-25, 1999 at the Rochester Institute of Technology, RIT, Rochester N.Y., USA.

Performance analysis of a parallel neural network training code for control of dynamical systems

Javier E. Vitela , Ulf R. Hanebutte[†] and José L. Gordillo
Instituto de Ciencias Nucleares
Universidad Nacional Autónoma de México
04510 México D.F., México

[†]Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore CA 94551, USA

Keywords: Parallel computing, artificial neural networks, nonlinear control, fusion reactor, message passing, MPI, SGI/Cray Origin 2000

Received: January 10, 2000

In this work we present the performance analysis of a parallel neural network controller training code that uses MPI, a standard portable message passing environment. The physical system used in this study is a nonlinear model of a tokamak thermonuclear fusion reactor in which the parameters were taken from the CDA-ITER (Conceptual Design Activity) group. A SGI/Cray Origin 2000 multiprocessor platform was used in the comprehensive performance analysis of the parallel code reported here, which includes a comparison of actual measurements with the results of the theoretical performance of ideal models. Three different assigments schemes for load balance were used in this study.

1 Introduction

The capability of artificial Neural Networks (NN) to approximate most types of nonlinear continuous functions have attracted the attention of the engineering community for the identification and control of nonlinear dynamical systems since over more that a decade ago.[1-4] However, a major obstacle in the use of NNs is the high computing cost of their training procedures in spite of the fact that different types of NNs can be used and several convergence acceleration techniques have been developed to speed up training.[5-7] On the other hand, most of the scientific and engineering laboratories today, have access to multiprocessor computers and/or maintain a small network of workstations or personal computers which provides potential parallel computing platforms. Hence, parallel programming is becoming more accesible to researchers in all scientific fields and in particular it will be increasingly relevant for NN training in the future.[8,9]

Until the appearance of scalable shared memory multiprocessor architectures and the recently developed standard for shared memory programming called OpenMP,[10] the most widely used model for developing scalable parallel programs has been *message passing*;[11] this approach, which assumes a configuration of processors with distributed memory able to work cooperatively, is intended mainly for coarse grained parallelism[8] and provides the two key aspects of parallel programing: syncronization of processes and read/write access to their local memory. Although in principle all message passing enviroments are logically similar, there are differences in the way the above tasks can be implemented. As a consequence, the message passing systems provided by the vendors in the past for their particular machines resulted not only incompatibles, but also highly machine dependent. Thus, a parallel program written using the native directives of one particular machine needs to be significantly modified in order to run on a different parallel computer. In an attempt to avoid the vendor/machine specificity of the native message passing enviroments, some research groups around the world developed different portable message passing platforms. Some examples of these platforms are Linda, P4, PVM and Zipcode among others. Unfortunately except for PVM which was the most succesful, a large number of portable platforms did show some of the same problems of the native enviroments provided by the vendors, and furthermore most of them supported only a subset of the full features of the vendor systems.[11] To overcome this problem, a portable message passing environment for parallel programming, known as MPI (Message Passing Interface) was developed in the last decade by a group of vendors, laboratories and universities worldwide, with the express purpose of creating a standard.[12,13] This platform support all of the best features of the vendors and other portable systems and it is capable to work efficiently in a multiprocessor computer as well as in a cluster of workstations or PCs connected via ethernet or a faster LAN.[14]

Similarly to all message passing enviroments, a parallel computation in MPI lies within the distributed memory SPMD framework (Single Program Multiple Data), and consists of a set of processors concurrently running copies of a single program written in a standard language, such as F77 or C, augmented by library calls to MPI functions for sending and receiving messages between processes as well as for task synchronization.

The purpose of this paper is to present the performance analysis of a parallel NN training code that uses MPI, and demonstrate the advantages of parallel programing by means of a representative control problem: the stabilization at near ignition condition of a nonlinear model of a thermonuclear fusion reactor. The detailed performance analysis presented in this paper compares three different load distributions schemes: two static (block and cyclic distributions) and one quasi-static (a sliding average bin packing also known as best-fit algorithm). The results of the study are presented for the SGI/Cray-Origin 2000 at the Supercomputing Center of the National University of México (UNAM).[15]

The rest of the paper is organized as follows: In section 2 we briefly present the dynamical model of the tokamak reactor used in this study; section 3 discuss the sequential algorithm used previously for NN training; section 4 contains a preliminary performance study based on Amdhal's law; in section 5 it is described the parallel algorithm developed using MPI for training neural networks; the results of the performance analysis of the parallel code based on a theoretical model as well as from actual time measurements are presented in section 6; and finally section 7 contains the concluding remarks.

2 The physical dynamical model

The dynamical system used for this work is a zero-dimensional nonlinear model of a tokamak fusion reactor; although a detailed description of the physical system and the stabilization problem are discussed in References 16-17, here we present a brief summary.

The physical model assumes that the plasma ions and electrons have Maxwellian distributions and share the same temperature at all times; the plasma is solely composed by D-T in equal proportions, n_{DT}, electrons n_e and fully ionized helium ions n_α, which is also referred to as helium ash; in addition, the electron density is determined by the quasi-neutrality condition, $n_e = n_{DT} + 2n_\alpha$. Here, no high-Z impurities are considered, and it is assumed that the alpha particles produced by the fusion reactions are instantaneously thermalized. The control actions considered here include the simultaneous modulation of the D-T refueling rate, the injection of a neutral He-4 beam as well as an auxiliary heating power, which are constrained to take values within a maximum and minimum levels.

The following set of coupled differential equations describe the time evolution of the D-T, helium ash and thermal energy densities, respectively:

$$\frac{d}{dt}n_{DT} = S_f - 2\left(\frac{n_{DT}}{2}\right)^2 < \sigma v > - \ n_{DT}/\tau_P \qquad (1)$$

$$\frac{d}{dt}n_\alpha = S_\alpha + \left(\frac{n_{DT}}{2}\right)^2 < \sigma v > - \ n_\alpha/\tau_\alpha \qquad (2)$$

$$\begin{aligned}\frac{d}{dt}\left[\frac{3}{2}(n_e + n_{DT} + n_\alpha)T\right] &= P_{aux} \ + \ \eta j^2 + \\ Q_\alpha\left(\frac{n_{DT}}{2}\right)^2 &< \sigma v > - A_b Z_{eff}^2 \, n_e T^{1/2} - \\ \frac{3}{2}(n_e + n_{DT} + n_\alpha)\frac{T}{\tau_E}. \end{aligned} \qquad (3)$$

In the above equations, S_f represent the refueling rate, S_α the neutral He-4 injection rate, and P_{aux} the auxiliary heating power density; j is the average plasma current density, $Q_\alpha = 3.5$ Mev is the energy carried by the fusion alpha particles, $< \sigma v >$ is the D-T reactivity, A_b is the coefficient associated to the bremsstrahlung radiation losses, and η is the neoclassical parallel resistivity. Finally τ_E, τ_P, and τ_α stands for the energy, the D-T fuel and the helium ash confinement times, respectively.

Using the quasi-neutrality condition $n_e = n_{DT} + 2n_\alpha$, and the corresponding expressions for the reactivity and the resistivity, the above set of equations are transformed into the following equations,

$$\frac{dn_e}{dt} = \ S_f \ - \ \left(\frac{2f_\alpha}{\tau_\alpha} + \frac{1-2f_\alpha}{\tau_P}\right)n_e \ + \ 2S_\alpha \ , \qquad (4)$$

$$\begin{aligned}\frac{df_\alpha}{dt} &= \tfrac{1}{4}n_e(1-2f_\alpha)^2 < \sigma v > - f_\alpha S_f/n_e + \\ & S_\alpha(1 - 2f_\alpha)/n_e + f_\alpha(1-2f_\alpha)\left(\tfrac{1}{\tau_P} - \tfrac{1}{\tau_\alpha}\right)\end{aligned} \qquad (5)$$

and

$$\begin{aligned}\frac{dT}{dt} &= \frac{2}{3}\ \frac{P_{aux}}{n_e(2-f_\alpha)} \ + \ \frac{2(1-2f_\alpha)}{2-f_\alpha}(T/\tau_P) \ + \\ &\left(\tfrac{1}{6}Q_\alpha + \tfrac{1}{4}T\right)n_e < \sigma v > \frac{(1-2f_\alpha)^2}{2-f_\alpha} \ - \\ &\tfrac{2}{3}A_b\frac{(1+2f_\alpha)}{2-f_\alpha}n_e T^{1/2} \ - T/\tau_E \ + \frac{3f_\alpha}{2-f_\alpha}\frac{T}{\tau_\alpha} \ + \\ &\tfrac{2}{3}A_h\frac{(1+2f_\alpha)^{0.5}}{n_e(2-f_\alpha)T^{3/2}}\left(1 + 1.198(1+2f_\alpha)^{0.5} + \right. \\ &\left. 0.222(1+2f_\alpha)\right)/\left(1 + 2.966(1+2f_\alpha)^{0.5} + \right. \\ &\left. 0.75(1+2f_\alpha)\right) - \frac{2T}{n_e(2-f_\alpha)}S_f - \frac{3T}{n_e(2-f_\alpha)}S_\alpha \ ;\end{aligned} \qquad (6)$$

for the electron density n_e, the relative fraction of helium ions defined by $f_\alpha = n_\alpha/n_e$, and the plasma temperature; the CDA-ITER energy confinement time τ_E, is obtained from the following expression

$$\tau_E = 0.082\, I_P^{1.02}\, R^{1.6}\, B_0^{0.15}\, A_i^{0.5}\, \kappa_x^{-0.19}\, P_{net}^{-0.47} \ ; \qquad (7)$$

where P_{net} represent the net plasma heating given by $V_{core} \quad (P_{aux} + P_\alpha + P_{oh} - P_{br})$; with V_{core} the volume of the plasma, and P_{aux}, P_α, P_{oh} and P_{br} are the auxiliary heating, the alpha particle heating, the ohmic heating and the bremsstrahlung energy radiation densities respectively. It is assumed here that $\tau_P = 3\tau_E$ and $\tau_\alpha = 7\tau_E$. For the reactor parameters specified by the CDA-ITER (Conceptual Design Activity) group,[16] the ignited steady state condition, *i.e.* where $P_{aux} = 0$ and $S_\alpha = 0$, can be obtained for $S_o = 4.16 \times 10^{18}$ m^{-3}sec^{-1} when $n_0 = 9.8 \times 10^{19}$ m^{-3}, $T_0 = 8.28$ Kev and $f_0 = 0.0624$, values that will be assumed to constitute the desired operating point for the CDA-ITER ignited tokamak reactor of interest for this work.

Because both the state and the control variables have different units, it is convenient to define the following normalized variables using the nominal ignited operating point,

$$z_1 \equiv n_e/n_0 \ , \ z_2 \equiv f_\alpha/f_0 \text{ and } z_3 \equiv T/T_0 \ , \tag{8}$$

for the state variables and

$$\hat{S}_f \equiv S_f/n_0 \ , \quad \hat{S}_\alpha \equiv S_\alpha/(f_0 n_0) \text{ and } \\ \hat{S}_{aux} \equiv P_{aux}/(3n_0T_0/2) \ , \tag{9}$$

for the normalized DT refueling rate, the external beam source of neutral helium atoms, and the auxiliary heating power density, respectively. Further details are discussed in Ref. [17]. The resulting normalized Eqs. (4)-(6) represent the dynamical system used in this work. In the following section we briefly describe the functioning of feedforward NN and present the sequential training algorithm.

3 Sequential neural network controller algorithm

The purpose of the NN is to provide a control law of the form $\mathbf{u} = \mathbf{u}(\mathbf{z})$ for time discrete dynamical systems of the form $\mathbf{z}_{k+1} = \mathbf{F}(\mathbf{z}_k, \mathbf{u}_k)$, in order to drive the system from an arbitrary state within a given region, toward a desired target state; where $\mathbf{z}_k$ and $\mathbf{u}_k$ represent the state and the control variables respectively, at time step k ($k = 1, 2, \ldots$). The type of NN used in this work is the well known feedforward multilayer NN with sigmoidal activation functions.

Feedforward neural networks can be considered as nonlinear continuous multivariate mappings composed by nonlinear computational elements or nodes arranged in L layers with $L \geq 3$. The nodes in the first and last layers of the network are fixed by the nature of the problem itself; however, the number of hidden layers and the number of nodes in each of these layers are not known a priori, and although some authors have proposed the use of some techniques based on information theory for NN design,[18] in general they are still determined through a trial and error process. The number of nodes in each layer will be denoted here by $J(l)$ with $l = 1, \ldots, L$. Each one of the $J(1)$ nodes in the first layer are mapped through the identity function, i.e. these nodes receives as input one of the $J(1)$ components of the input vector and transmites them unaltered to serve as the input values to the nodes of the second layer. Communication exists only between neurons in adjacent layers and is implemented through unidirectional feedforward connections known as weights. The nodes in the hidden and output layers are constituted by nonlinear functions, mapping a multidimensional input received from the nodes of the immediately preceding layer into a one dimensional output. The activation of the j-th node in the l-th layer, $\mathcal{O}_j^{(l)}$, is given by

$$\mathcal{O}_j^{(l)} = f_j(\text{input}_j^{(l)}) \quad ; \tag{10}$$

where $\text{input}_j^{(l)}$, the effective input to node j, is the weighted sum of the outputs $\mathcal{O}_i^{(l-1)}$ of the immediately preceding layer, i.e.,

$$\text{input}_j^{(l)} = \sum_{i=1}^{J(l-1)} \omega_{ji}^{(l)} \mathcal{O}_i^{(l-1)} + \theta_j^{(l)} \quad ; \tag{11}$$

with $\omega_{ji}^{(l)}$ denoting the weight that connects the output of node i in layer $l-1$ with the node j in layer l; $\theta_j^{(l)}$ and f_j correspond to the threshold value and the nonlinear activation function of this node, respectively. For short, we will refer as weights $\mathbf{w}$ to both, the parameters $\omega_{ij}^{(l)}$ and the thresholds $\theta_j^{(l)}$. The activation function used in this work is the sigmoid function,

$$f = \frac{1}{1 + e^{-\text{input}_j^{(l)}}} \quad . \tag{12}$$

which is the most widely used activation function in feedforward neural networks.[19] Since the weights and thresholds connecting the nodes can take any real value, the output of this activation function is bounded between 0 and 1.

In this work each element of the input vector corresponds to the normalized values of the electron density, the relative fraction of helium ash and the plasma temperature at a given time step. The elements of the output vector on the other hand, are associated with the normalized D-T refueling rate, neutral helium injection rate, and auxiliary power heating,

$$\hat{S}_f = (S_0/n_0)k_1 u_1 \ , \quad \hat{S}_\alpha = k_2(2u_2 - 1)^2 \quad \text{and} \\ \hat{S}_{aux} = k_3(2u_3 - 1)^2 \ ; \tag{13}$$

where u_1, u_2 and u_3 are the activation levels of the output nodes of the neural network, which are bounded betwen 0 and 1, as will be further discussed later on in this section; and in this work we chosed $k_1 = 4.0$, $k_2 = 0.1s^{-1}$ and $k_3 = 0.1s^{-1}$, defining the maximum values that the refueling rate, the neutral helium injection and the auxiliary power heating can take respectively.[17]

In order to stabilize the system around a given state, the neural network must provide appropriate values for the control variables, according to the current state of the system. In contrast with the training process of NNs for pattern recognition tasks,[19,20] in which a set of input-output teaching patterns to the network are provided, here the correct control values are not known during the training process: only the deviations of the state variables of the reactor with respect to their target values are available. Thus by considering the NN-dynamical system joint configuration as a single unit, see Fig. 1, a set of input-output teaching patterns for training can be generated. It will be denoted by $\mathcal{E}_m$ the error between the target state $\mathbf{z}_t$ and the actual final state $\mathbf{z}_{pk_f}$ of the trajectory reached, after k_f time steps, by the joint NN-dynamical system configuration given initial condition m.

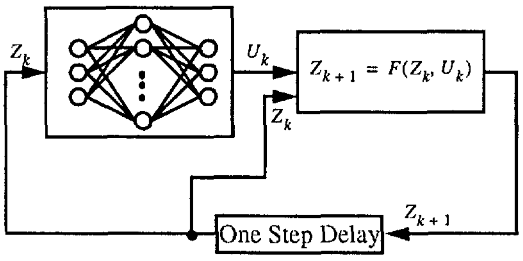


Figure 1: NN-dynamical system joint configuration.

Thus, for a particular set of weights specifying the NN, and a subset of M initial conditions that the system can take, the total error $\mathcal{E}$ is defined then as the sum of these individual errors after the generation of the set of M trajectories, i.e.

$$\mathcal{E} = \sum_{m=1}^{M} \mathcal{E}_m = \frac{1}{2} \sum_{m=1}^{M} |\mathbf{z}_{mk_f} - \mathbf{z}_t|^2 \quad . \qquad (14)$$

A succesful training, given a topology of the feedforward NN, consists of determining a set of weights $\mathbf{w}$ for which the total error $\mathcal{E}$ is a global minimum. For this purpose a formal methodology known as Backpropagation Through Time (BPTT) [21,22] is used to calculate the gradient of the error $\mathcal{E}$ in weight-space, $\nabla\mathcal{E}$, and iteratively reduces this error by updating the weights using a conjugate gradients method.[23,24] This technique is a quadratically convergent gradient method that locates an unconstrained local minimum of a quadratic function in a finite number of steps if round-off errors could be avoided. For non-quadratic functions, which is the case of the training error $\mathcal{E}(\mathbf{w})$ in Eq. (14), the procedure is then iterative. Similarly to all gradient-based optimization methods, this procedure is not guaranteed to find the global minimum, nevertheless the algorithm will lead to the bottom of whatever valley it starts in. Furthermore, this technique has the advantage that it avoids the undesirable phenomenon of premature saturation of the network output units. The ocurrence of such saturation precludes significant improvements in the network weights over a large number of iterations causing an unnecessary increase in the computational cost to train the NN.[25]

The following steps summarize the sequential algorithm used to train a feedforward NN as a dynamic system controller:

- Step 1: Randomly select the initial values of the set of weights that specify the NN.
- Step 2: An admissible region of the phase-space is divided in a number M of cells.
- Step 3: Generate a set of M trajectories by randomly selecting an initial state from each one of the cells in phase space. Then allow the NN-dynamical system to evolve in time until a pre-specified maximum number of time steps is reached, the state of the system has moved out of a certain region or satisfied some other criteria.
- Step 4: Convergence is achieved when the entire set of M trajectories satisfies the convergence criterion $|z_i - z_{it}| \leq \epsilon$, for all i, where ϵ is a pre-specified error range and the subindex i corresponds to the components of the state-space vector $\mathbf{z}$. Otherwise proceed to the next step.
- Step 5: Using BPTT, calculate the gradient of the error, $\nabla\mathcal{E}$, produced by all the different trajectories and update the weights using the method of conjugate gradients; a one dimensional search is implemented here to find the minimum of the error $\mathcal{E}$ along the conjugate direction.
- Step 6: Repeat steps 3-5 until the training of the NN is successfully completed, i.e. when the entire set of P trajectories, each of which starts from a different cell, reaches the target $\mathbf{z}_t$ within the error range ϵ.

Basically the above training algorithm involves two phases. In the first phase the joint NN-dynamical system configuration evolves in time from an initial state $\mathbf{z}(1)$, until a final state $\mathbf{z}(N)$. This final state is compared with its target value, resulting in an error signal for each of the state variables of the system. In the second phase the error signals are propagated recursively backward in time from the final time step N until the initial time step 1, and are used to calculate the adjustments needed in the weights composing the NN using the components of $\nabla\mathcal{E}$ with the method of conjugate gradients.

Using the concept of ordered partial derivatives,[17,21,22] the components of the gradient of the error $\mathcal{E}_m$ with respect to the weights of the NN, associated to trajectory generated from the m-th cell, i.e. $\partial^{\dagger}\mathcal{E}_m/\partial\omega_{pq}^{(l)}$, are obtained from the following expression

$$\frac{\partial^{\dagger}\mathcal{E}_m}{\partial\omega_{pq}^{(l)}} = \sum_{n=1}^{N-1} \sum_{j=1}^{J(L)} \frac{\partial^{\dagger}\mathcal{E}_m}{\partial u_j(N-n)} \frac{\partial u_j(N-n)}{\partial\omega_{pj}^{(l)}} \; ; \qquad (15)$$

the second factor in the RHS of the above equation , i.e. $\partial u_j(N-n)/\partial \omega_{pj}^{(l)}$ is calculated for a feedforward NN using the standard backpropagation technique of Rumelhart et.al.,[19] while the first factor is obtained using the following recursive set of equations,

$$\frac{\partial^\dagger \mathcal{E}_m}{\partial u_j(N-n)} = \sum_{i=1}^{J(1)} \frac{\partial^\dagger \mathcal{E}_m}{\partial z_i(N-n+1)} \times \frac{\partial z_i(N-n+1)}{\partial u_j(N-n)}, \quad \text{for } n=1,\ldots,N-1; \tag{16}$$

with

$$\frac{\partial^\dagger \mathcal{E}_m}{\partial z_i(N-n+1)} = \sum_{j=1}^{J(L)} \frac{\partial^\dagger \mathcal{E}_m}{\partial u_j(N-n+1)} \frac{\partial u_j(N-n+1)}{\partial z_i(N-n+1)} + \sum_{k=1}^{J(1)} \frac{\partial^\dagger \mathcal{E}_m}{\partial z_k(N-n+2)} \frac{\partial z_k(N-n+2)}{\partial z_i(N-n+1)}, \quad \text{for } n = 2,3,\ldots N-1 \tag{17}$$

and the following initialization,

$$\frac{\partial^\dagger \mathcal{E}_m}{\partial z_i(N)} = \frac{\partial \mathcal{E}_m}{\partial z_i(N)} \quad . \tag{18}$$

In the above equations the symbol † is used to denote an ordered partial derivative, and $u_j(N-n)$, the j-th output of the NN, is the value of the corresponding control variable in Eq. 13 at time step $N-n$; $z_i(N-n+1)$ correspond to the value of the i-th state variable at time step $N-n+1$. It can be pointed out that the name *Backpropagation Through Time* is due to the fact that the above set of recursive equations initializes at the last time step of the trajectory and proceeds backwards in time until the first time step is reached.

The sequential code just described has been successfully used to control and stabilize some nonlinear dynamical systems.[17,26,27] For the stabilization problem described in Section 2, the NN used was a three layer feedforward artificial network with sigmoidal activation functions. The first layer contains three nodes corresponding to the electron density, the fraction of the alpha particles and the temperature of the plasma. The second layer contains 16 units and the output layer three nodes corresponding to the control actions associated to the D-T refueling rate, the neutral He-4 beam injection rate, and the auxiliary heating modulation.

4 Preliminary study using Amdahl's law

A preliminary analysis was done in which the computationally demanding and the potentially parallel components of

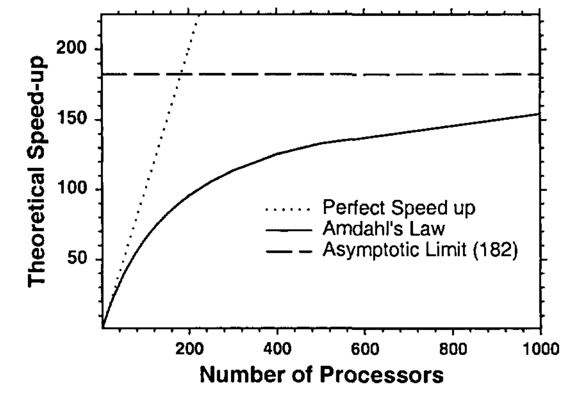


Figure 2: Theoretical ideal speedup obtained from Amdahl's law.

the sequential code were identified. After timing the different sections of the code when restricting the run to only 115 iterations of the algorithm described in the above section, the fraction of the total CPU time spent in the parallelizable parts of the code was estimated as $f \approx 99.45\%$. In general the sequential part of the code would impose a theoretical upper limit to the execution time whenever it runs on multiple processors. The *speedup*, s, defined as the ratio between the execution time in a single processor $\mathcal{T}_1$ and the execution time on multiple processors $\mathcal{T}_n$, is one of the parameters used to measure the performance of a parallel code.[8] Assuming perfectly uniform load balance among the processors, no communication cost between the processors, as well as negligible overhead computational cost due to parallel handling, Amdahl's law gives the following ideal speedup for the parallel code,[8,9]

$$s = 1/[(1-f) + f/p] \quad ; \tag{19}$$

where f is the fraction of the code with parallel content and p is the number of processors involved. Figure 2 shows the behavior of the ideal theoretical speedup given by Eq. (19) as function of the number of processors. According to this law a maximum speedup of approx. 182, could be ideally achievable if an unlimited number of processors would be put to work in this problem. In practice this is of course not possible due to the limitations imposed by the above mentioned factors. In particular, we can point out that the work load generated in each of the cells can not be divided in smaller loads; nevertheless, this give us an estimation of what would we expect under ideal conditions after parallelizing the code.

5 Parallel training algorithm

The sequential training algorithm described above was parallelized using the fact that the gradient of the total error $\nabla\mathcal{E}$, is the sum of the gradients of the individual errors of each of the M independent trajectories,[28]

$$\nabla\mathcal{E} = \sum_{m=1}^{M} \nabla\mathcal{E}_m \quad . \qquad (20)$$

Thus, assuming that we have a set of P processors, labeled $p = 0, 1, \ldots P-1$, which are available to contribute to the calculation of $\nabla\mathcal{E}$ following a given load balance scheme, the following parallel algorithm can be devised,

- Step 1: Task assigned to processors $p = 0$ (also called *root*): Randomly select the initial values of the set of weights that specify the NN, and use MPI library calls to broadcast these values to all the other $P-1$ processors.
- Step 2: Task assigned to processor $p = 0$: Divide an admissible region of the phase-space in a number M of cells.
- Step 3: Task assigned to processor $p = 0$: Select M initial states by random sampling each one of the cells in phase space. Use MPI library calls to broadcast these states to all the other $P-1$ processors.
- Step 4: According to a given load distribution scheme each of the processors is assigned a subset of the M initial states. For each initial state the corresponding trajectories are generated following exactly the same stopping criteria as in Step 3 of the sequential algorithm; a record of the maximum number of time steps in each trajectory is made and will be used by the load balance scheme at the beginning of this Step in future iterations, as will be discussed further below.
- Step 5: Using BPTT each of the P processors calculate and store, the gradient of the individual error $\mathcal{E}_m$ associated to the subset of the M trajectories that were assigned to it by the load distribution scheme.
- Step 6: A test for convergence in each of their corresponding subsets is performed by each processor and the results are sent, using MPI calls, to processor $p = 0$, which determines whether or not global convergence has been achieved. If the global convergence criteria is satisfied one should stop; otherwise proceed to the next step.
- Step 7: Using the MPI global sum operation, the partial gradient of the errors, $\nabla\mathcal{E}_m$, are shared by all P processors; each processor then proceed to determine, by adding these components in the same ordered sequence, the gradient of the total error $\nabla\mathcal{E} = \sum_{m=1}^{M} \nabla\mathcal{E}_m$ avoiding thus the differences due to roundoff errors, ocurring when no care is taken in the order in which the individual terms are added. This ensures us that the results will be independent of the number P of processors involved in the computation.
- Step 8: All processors individually use $\nabla\mathcal{E}$ to determine the new conjugate gradients direction, and cooperate in a way similar to step 4 in the search for the minimum along this direction.

Table 1: MPI library function calls used in the NN parallel training code

mpi_init	Initializes the MPI execution environment.
mpi_comm_size	Determines the size of the group associated with a communicator.
mpi_comm_rank	Determines the rank (label) of the calling processor within the communicator.
mpi_bcast	Broadcasts a message from the processor with rank "root" to all other processors of the group.
mpi_allreduce	Combines values from all processors and distribute the result back to all processors.
mpi_barrier	Blocks until all processors have reached this routine.
mpi_wtime	Returns the elapsed time on the calling processor.
mpi_finalize	Terminates the MPI execution environment.

- Step 9: Updating of the weights is done in each processor.
- Step 10: Repeat steps 3-9 until the training of the NN is successfully completed, i.e. when the entire set of M trajectories, each of which starts from a different cell, reaches the target $\mathbf{z}_t$ within the error range ϵ.

In Figure 3 we show the flow diagram of the parallel code in which the parallel tasks and communications functions are shown explicitly. Table 1 contains the MPI library calls used in the implementations of the parallel code.

Efficient parallel codes distribute as uniform as possible the work load among all the processors involved in the computation. In the algorithm described above the work load to be distributed among the processors is produced by the set of M trajectories generated from each cell in phase space by the NN-dynamical system. Here, work load associated to each trajectory is assumed proportional to its lenght, i.e. to the total number of time steps contained in it. This assumption was sustented by recording data from the serial execution time of the subroutine TRAJEC in charge of generating trajectories and the gradient of the error using BPTT; from this data we concluded that the CPU time needed per time step in the generation of a trajectory was constant for all practical purposes. Although this gives us a measure of the work load, the load balance is complicated by the fact that the number of time steps generated by each trajectory is not constant: it changes from cell to cell in phase space, varies during the one dimensional search for the minimum $\mathcal{E}$, from iteration to iteration of the training algorithm, and furthermore, can not be known a priori. Since the performance of any parallel code depends strongly on an efficient load balance, three task distribution schemes were used for comparative studies of the performance of the parallel NN training code: block distribution, cyclic distribution and a load assigment based on a sliding average together with the bin packing (best-fit) algorithm.[8,29] The first two are static assigment strategies while the last one is a quasi-static scheme, i.e. it is determined at each iteration using the actual work load generated in previous iterations of the algorithm, and remain fixed until the next iteration.

It should be pointed out that if the actual work load information were available in advance, the bin packing scheme would produce the best load balance possible. However, this is not possible since only estimates of the actual work load based on former iterations of the training algorithm are available, and hence, only suboptimal load balances can be achieved in practice. In what follows we briefly describe each of the load distribution schemes used in this study:

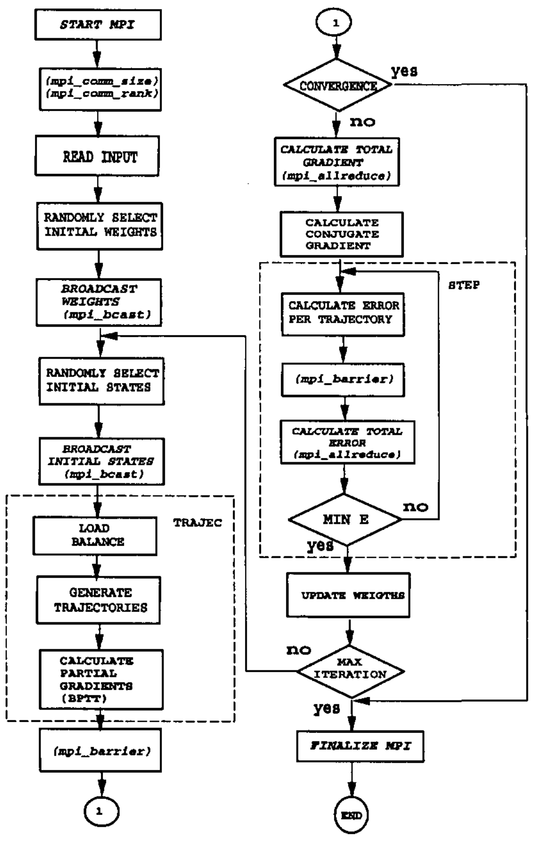

START MPI
(mpi_comm_size)
(mpi_comm_rank)
READ INPUT
RANDOMLY SELECT INITIAL WEIGHTS
BROADCAST WEIGHTS (mpi_bcast)
RANDOMLY SELECT INITIAL STATES
BROADCAST INITIAL STATES (mpi_bcast)
TRAJEC
LOAD BALANCE
GENERATE TRAJECTORIES
CALCULATE PARTIAL GRADIENTS (BPTT)
(mpi_barrier)
1
1
CONVERGENCE
yes
no
CALCULATE TOTAL GRADIENT (mpi_allreduce)
CALCULATE CONJUGATE GRADIENT
STEP
CALCULATE ERROR PER TRAJECTORY
(mpi_barrier)
CALCULATE TOTAL ERROR (mpi_allreduce)
MIN E
no
yes
UPDATE WEIGTHS
no
MAX ITERATION
yes
FINALIZE MPI
END

Figure 3: Flow diagram of the parallel NN training code in which the main MPI functions calls used in the code are shown explicitly.

- Block Distribution: In this scheme the work load is distributed as follows: We divide the total number of cells by the number of processors *npro* involved in the computation, by rounding off this number we obtain the number of tasks *ntra* to be assigned to each processor. The first processor is in charge of generating the trajectories and error gradients of cells numbered $1, 2, \ldots ntra$, the second processor is in charge of those calculations involving cells numbered $ntra + 1, \ldots 2ntra$, and so on until the *ncells* have been distributed.

- Cyclic Distribution: After finding the number of cells per processor *ntra* to be assigned as described in the above item, the *ncells* are distributed as follows: The processor with label *id* is in charge of the cells numbered $id + 1, npro + id + 1, 2 * npro + id + 1, \ldots, (ntra - 1) * npro + id + 1$ for $id = 0, \ldots, npro - 1$.

- Predicted Bin Packing: In this scheme the total number of time steps generated by the trajectories in each of the cells during the five previous iterations are used to obtain an average value and predict the work load for each cell at the current iteration. The bin-packing algorithm is then used to distribute the different cells among all the *npro* processors as follows: the cells are ordered according to their predicted work load; the cell with the highest work load is assigned to the processor with $id = 0$; the cell with the second highest value is assigned to the processor with $id = 1$, and so on, until the last processor, i.e. that with $id = npro - 1$ has been reached. The rest of the unassigned cells are then distributed according to the following rule: the cell with the highest work load is assigned to the processor with the currently lowest accumulated work load; this procedure is repeated until all cells have been distributed.

- Actual Work Load Balance: In order to compare the performance of the above load distribution schemes, we perform a preliminary run in order to calculate the actual number of time steps generated by the different trajectories in each cell at Step 4 of the algorithm. With this information we runned the code, using at each iteration the bin packing algorithm described above, in order to distribute the actual work load (obtained by the preliminary run), thereby obtaining the optimal load distribution at each iteration.

It should be pointed out that if the work load were equal regardless of the cell, any scheme would lead to similar load balances. However, such equal distribution is not observed for the problem we are concerned with in this work.

6 Performance analysis

The parallel NN training code was initially developed and tested on a multiprocessor SUN workstation and then ported with ease to the high performance multiprocessor platform for which performance data is presented. Specific compiler options were chosen to optimize the single processor performance of the code.

The SGI/CRAY-ORIGIN 2000 at UNAM is configured as two independent modules, one with 8 and the other with 32 processors for a total of 40 processors. All processors in the system are 195 MHz MIPS R10000 processors, each of which has a 4 MB secondary cache and 512 MB of memory per node (two processors).[15] The presented results were obtained during dedicated user operation, where access to the entire system by other users was disabled.

In this specific application, the phase-space was divided into $M = 27$ cells around the nominal operating point, thereby restricting the parallel granularity to 27 processors. For this study, the code was restricted to perform only 115 iterations. The complete calculation performed required approximately 72.08 seconds on a single processor of the SGI/Cray Origin 2000; the parallel subroutines TRAJEC and PATTERRROR, which perform Steps 4, 5 as well as the one dimensional search for the minimum at Step 8 of the algorithm, required 30.81 and 41.15 seconds, respectively.

The performance of the resulting code was measured by two parameters: the speedup and the parallel efficiency. As discussed earlier the speed up is defined by the ratio between the execution times in one processor and in p processors. The execution time of a parallel program running in p processors, $\mathcal{T}_p$, is the interval of time measured from the instant the first processor starts to work until the time the last processor finishes working, This time can be divided in three components:

$$\mathcal{T}_p = t_{comp}(i) + t_{comm}(i) + t_{iddle}(i) \quad , \qquad (21)$$

where $t_{comp}(i)$ is the time spent by the i-th processor performing computation, including the overhead cost due to parallel handling; $t_{comm}(i)$ is the actual time spent by this processor communicating with other processors, i.e. sending and receiving messages; and $t_{iddle}(i)$ is the total time spent by the i-th processor waiting for data from the other processors. The time $\mathcal{T}_1$ is the execution time in a single processor and is given by $\sum_{i=1}^{p} t_{comp}(i)$. The quantity $\mathcal{T}_p$ is the same regardless the processor in which the timing data is obtained. The speedup is thus given by,

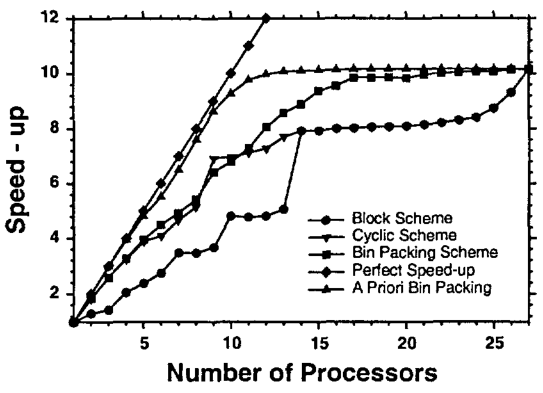
$$s \equiv \mathcal{T}_1/\mathcal{T}_p \quad . \qquad (22)$$

On the other hand the efficiency of the parallel code measure the fraction of the execution time that all the processors spent doing useful work, i.e. performing computations,

$$e \equiv \sum_{i=1}^{p} t_{comp}(i)/(p\,\mathcal{T}_p) = s/p \quad . \qquad (23)$$

Efficiencies of less than 50% means, therefore, that the code is spending more time iddling and communicating between processors than doing useful work.

In Fig. 4 are shown the predicted speedup's (left) and efficiencies (right) obtained using the trajectory length as a measure of the work load, as discussed above; results are shown for the different load balance schemes discussed in Section 4. We should point out here, that these figures show an almost linearly perfect speed up that the bin packing scheme would yield for up to 10 processors if the work load were known a priori.



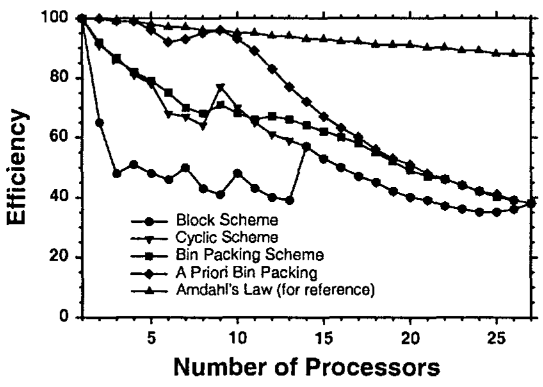


Figure 4: Predicted behavior of the speed-up (top) and efficiency (bottom), as a function of the number of processors, for the different load balance schemes when the trajectorie's lenght are used as a measure of the work load.

The actual speedup's and efficiencies obtained from timing the subroutine TRAJEC and the entire code are shown in Figures 5 and 6, respectively. The results shown in Fig. 5 exclude the cost of the load balance which is performed inside this subroutine as can be seen in Fig. 3; while those of Fig. 6 include all communications and parallel handling costs. The measured speedup and efficiency match the model predictions quite well; the small deviations observed for larger number of processors are expected since the performance model neglects some higher order effects in the BPTT routine (step 5) and the synchronization step at the end of the TRAJEC routine implemented by a call to the *mpi_barrier* function; the higher order effects are due to the fact that the total number of arithmetic operations to calculate $\nabla\mathcal{E}_m$ does not scale linearly with the trajectory length, as can be seen in Eqs. 15-18. In addition these figures contain the predicted speedup's and efficiencies (solid lines) and, for reference, they also include

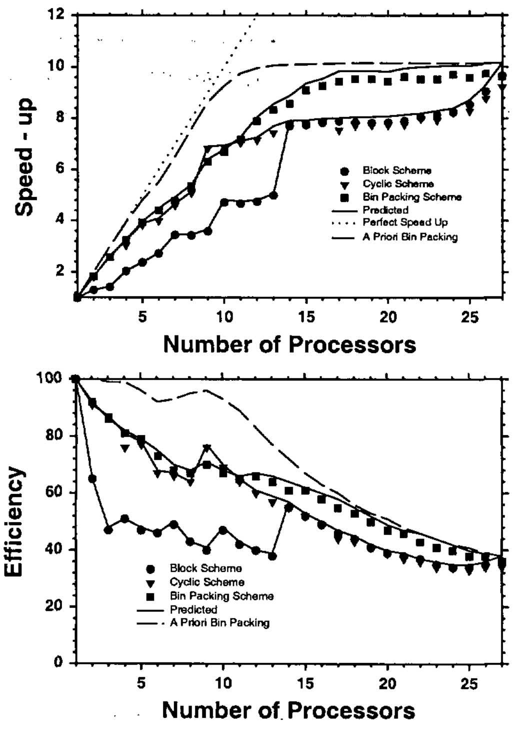


Figure 5: Speed-up (top) and efficiency (bottom) obtained from timing measurements of the subroutine TRAJEC with the different load balance schemes.

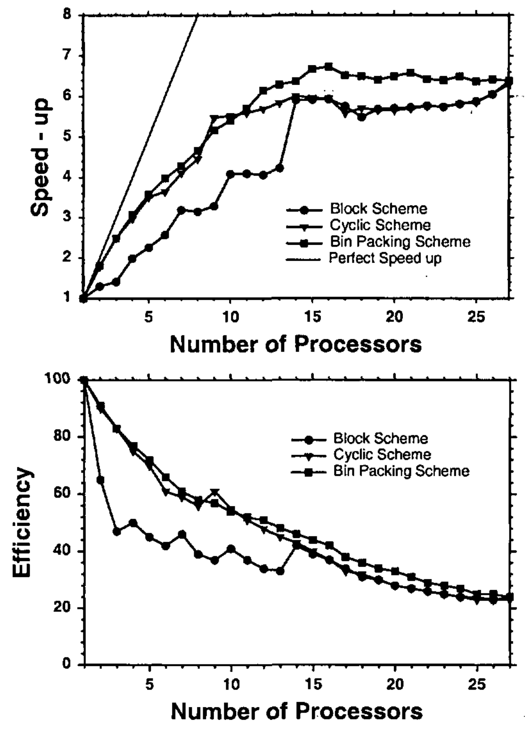


Figure 6: Speed-up (top) and efficiency (bottom) obtained from actual timing measurements of the entire code for the different load balance schemes used.

the limits yielded by the bin packing scheme if the knowledge of the actual work load of every cell at each iteration were available a priori. As observed, in general the predicted bin packing load distribution scheme outperformes the other two. This is expected because the work load for each cell is not equally distributed in phase space, resulting in a highly unbalance load distribution. When comparing the cyclic and the bin packing schemes in Fig. 5, it is observed that for up to 11 processors, the performances are very similar. However, for the cyclic scheme the slope of the speedup flattens sharply at 11 processors, while the speedup with the bin packing scheme grows steadily up to 17 processors reaching a plateau afterwards. Figure 5 shows that for 18 processors the speedup of the block and cyclic schemes is approx. 8 versus 9.6 for the bin packing scheme. These numbers correspond to a parallel efficiency of 40% and 50% respectively. As seen in the figures in general the block distribution yields low efficiencies, i.e. less than 50%, when more than 2 processors are used. An efficiency higher than 50% is obtained when the number of processors involved is less that 19 for the bin packing load balance scheme and when less than 15 processors are used for the cyclic scheme.

Although the resulting speedup's and efficiencies obtained by timing the total code, see Fig. 6, are smaller than for the subroutine TRAJEC, it is observed a similar behaviour. The culprit of the performance degradation is not the serial fraction of the code, as can be deduced from the discussions in Sec. 4, but the communications overhead. The serial fraction of the parallel NN training algorithm, does not noticeably affect the performance of the code, which is limited by the load balance of the parallel tasks. The communications overhead, however, significantly hinder the performance mainly due to a MPI library call for a global sum operation (mpi_allreduce), whose function is to collect the gradient of the partial error associated to the each of the trajectories. As with the subroutine TRAJEC, the total code with either the bin packing or the cyclic schemes shows a similar performance; the speedup increases up to 11 processors, and flattens afterwards for the cyclic scheme at a value around 5.8, while the speedup of the bin packing scheme grows steadily, reaching a plateau after 15 processors at a value around 6.6.

As observed in Figure 6 an efficiency of more 50% is obtained when used less than 12 processors with the bin packing algorithm and less than 11 for the cyclic scheme; the block distribution has a low efficiency compared with the other two schemes. A maximum speedup of 6.8 is reached when using 16 processors, with an efficiency of aprox. 42%. Figure 7 shows the measured time spent in the global MPI communication call all_reduce used to determine the total gradient of the error during the 115 iterations of this calculation, as a function of the number of processors. It is apparent that this function consumes an important fraction of the execution time hindering significatively the performance, as discussed above. The slight but noticeable

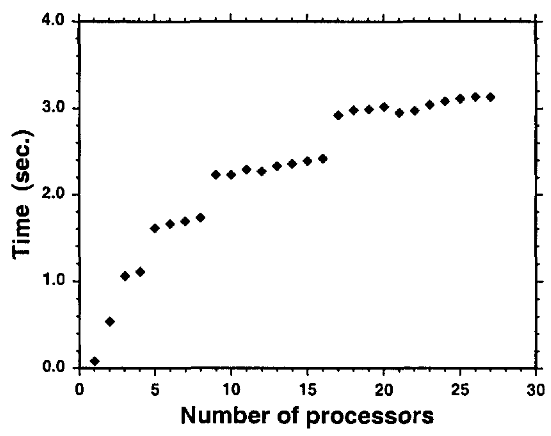


Figure 7: Measured mpi_allreduce communication function implementing a global sum of 28,458 mpi_real8 numbers stored in an array in the memory of each processor.

decrease in the speedup of Fig. 6 when going from 16 to 17 processors is explained by the behavior of this particular communication requirement, which shows jumps each time a multiple of 2 in the number of processors is reached.

7 Conclusions

In this work we have presented the performance analysis of a parallel NN training code for control of discrete dynamical systems that was developed using MPI, a standard message passing enviroment for parallel programing. The results of the study shows that an efficient portable parallel NN training code was built, and the use of only few MPI library functions calls demostrates the simplicity of the code. The study compared three different load balance schemes: the block distribution, the cyclic distribution and a predictive bin-packing scheme. The first two are static while the last one is quasi-static, i.e. the cells assigned to each processor are determined at each iteration during the training process and remain fixed during the entire iteration. It is concluded that the predictive bin-packing algorithm, utilizing an estimate of the work load based on a sliding average over the last five training iterations, yields higher speedups and efficiencies than the block and cyclic load distributions for the entire range of processing cluster sizes; this is due to the fact that not only the total work load is not uniformrly distributed among the cells, but also it is not know a priori. With this load distribution scheme the code is expected to to make efficient utilization of higher number of processors for larger problems, i.e. when larger number of cells are required in more complex problems. Parallel efficiencies of more than 50% are achieved, for the computation shown here, with the sliding average bin packing scheme, when up to 12 processors are used (speedup of approx. 6) in the SGI Origin 2000 multiprocessor platform.

Acknowledgments

The authors J.V. and J.L.G., gratefully acknowledge partial financial support from CONACYT 27974-E, DGAPA-UNAM IN116200 and CRAY/UNAM SC010399 projects. The work by the second author was performed under the auspices of the U.S. Department of Energy under contract No. W-31-109-ENG-38 as a member of the Reactor Analysis Division at Argonne National Laboratory.

References

[1] Vemuri, V.R. (Ed.) (1992) Artificial Neural Networks: Concepts and Control Applications. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[2] Miller III,T.W., Sutton, R.S. and Werbos, P.J., (Eds.) (1990) Neural Networks for Control. MIT Press, Cambridge Massachusetts.

[3] Narendra, K.S. and Parthasarathy, K.(1990) Identification and Control of Dynamical Systems using Neural Networks. *IEEE Trans. Neural Networks*, Vol 1, No. 1

[4] Sadeh, N. (1993) A Perceptron Network for Functional Identification and Control. *IEEE Trans. Neural Networks*, Vol 4, No. 6

[5] Minai, A.A. and Williams, R.D. (1990) Acceleration of BackPropagation Through Learning Rate and Momentum Adaptation. *Proc. Int. Joint Conf. Neural Networks*, Washington, D.C., January 15-19, 1990, Vol. I, p. 627 M. Caudill, Ed., IEEE Neural Networks Council.

[6] Battiti, R. (1990) Optimization Methods for BackPropagation : Automatic Parameter Tuning and Faster Convergence. *Proc. Int. Joint Conf. Neural Networks*, Washington, D.C., January 15-19, 1990, Vol. I, p. 593 M. Caudill, Ed., IEEE Neural Networks Council.

[7] Baba, N. (1992) A Stochastic Optimization Approach for Training the Parameters in Neural Networks", *Proc. Int. Workshop Computationally Intensive Methods, Simulation and Optimization* Laxenburg, Austria August 23-25, 1990, p. 55 G.Pflug and U. Dieter Eds., Springer-Verlag, N.Y.

[8] Foster, I. (1994) Designing and Building Parallel Programs. Addison Wesley, Massachussets.

[9] Pancake, C.M. (1996) Is parallelism for You? *Computational Science and Engineering* Vol. 3, No. 2 pp. 18-37

[10] Dagum, L. and Menom, R., (1998) OpenMP: An Industry-Standard API for Shared-Memory Programming. *IEEE Computational Science and Engineering*, January-March pp. 46-55

[11] McBryan, O.A. (1994) An Overview of Message Passing Enviroments. *Parallel Computing* Vol. 20, pp. 417-444

[12] The MPI Committe (1994) The Message Passing Interface (MPI) Standard.
http://www.mcs.anl.gov/mpi/index.html

[13] Groop, W., Lusk E. and Skjellum A. (1994) Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press.

[14] Bruck, J., Dolev D., Ho C.T., Rosu M.C. and Strong R. (1997) Efficient Message Passing Interface (MPI) for Parallel Computing on Clusters of Workstations. *J. Parallel and Distributed Computing* Vol 40, pp. 19-34

[15] Departamento de Supercómputo DGSCA-UNAM (1998) World Wide Web Document SGI/Cray-Origin 2000 at UNAM. http://www.super.unam.mx.

[16] Hui, W., Bamieh, B.A. and Miley, G.H. (1994) Robust Burn Control of a Fusion Reactor by Modulation of the Refueling Rate, *Fusion Technol.* Vol. 25, 318; and, Robust Burn Control of a Fusion Reactor with Modulation of Refueling Rate, *Fusion Technol.* Vol. 26, 1151.

[17] Vitela, J.E. and Martinell, J.J. (1998) Stabilization of Burn Conditions in a Thermonuclear Reactor Using Artificial Neural Networks. *Plasma Phys. Contr. Fusion*, Vol. 40 pp. 295-318

[18] Fogel, D.B. (1991) An Information Criterion for Optimal Neural Network Selection. *IEEE Trans. on Neural Networks* Vol.2 No.5, 490.

[19] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) Learning Internal Representations by Error Propagation. Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. I, p.319, MIT Press.

[20] Lippman, R.P. (1987) Introduction to Computing with Neural Networks. *IEEE ASSP*, Vol. 4, No. 2

[21] Werbos, P.J. (1990) Backpropagation Through Time: What It Does and How to Do It. *Proc. of the IEEE*, Vol.78, No.10, 1550

[22] Piché, S.W. (1994) Steepest Descent Algorithms for Neural Networks Controllers and Filters. *IEEE Trans. on Neural Networks*, Vol.5 No.2, 198

[23] Fletcher, R. and Reeves, C.M. (1964) Function Minimization by Conjugate Gradients. *Computer J.* Vol. 7, 149

[24] Reifman, J. and Vitela, J.E. (1994) Accelerating Learning of Neural Networks with Conjugate Gradients for Nuclear Power Plant Applications. *Nuclear Technology* Vol.106 No. 2, 225

[25] Vitela, J.E. and Reifman, J. (1997) Premature Saturation in Backpropagation Networks: Mechanism and Necessary Conditions. *Neural Networks* Vol 16 No. 4, 721

[26] Vitela, J.E., Hanebutte, U.R. and Reifman, J. (1996) An Artificial Neural Network Controller for Intelligent Transportation Systems Applications, *Proc. Neural Network Applications in Highway and Vehicle Engineeering Conference*, April 8-10, Ashburn, VA (1996)

[27] Hanebutte, U.R., Doss, E. Ewing, T. and Tentner A., Simulation of Vehicle Traffic on an Automated Highway System. *Mathl. Comput. Modelling* Vol. 27 pp. 129-141

[28] Hanebutte, U.R., Vitela, J.E. and Gordillo, J.L. (1998) A Parallel Neural Network Training Algorithm for Control of Discrete Dynamical Systems. *Proc. High Performance Computing HPC'98* (A. Tentner Ed.) Boston MA April 5-9, pp 81-87

[29] Graham, R.L. (1969) Bounds on Multiprocessing Timing Anomalies. *SIAM J. Appl. Math.*, Vol. 17 No. 2, 416.

Fault-tolerant ATM switching architectures based on MINs: A survey

Mohsen Guizani
Computer Science Department
University of West Florida
Email: mguizani@cs.uwf.edu

Muhammad Anan
Broadband Network Management Development
Sprint Telecommunications Company
Email: muhammad.anan@mail.sprint.com

Keywords: MIN, fault-tolerance, ATM switch, banyan network, SEROS, self-routing, reliability, availability.

Received: December 21, 1998

Multistage Interconnection Networks (MINs) offer an attractive way of implementing fast packet switching communication networks. Due to providing a very large capacity, MINs play an important role in building ATM switches. With the throughput requirements of the packet switches exceeding several Gigabits/sec, it is important to improve the fault-tolerance of these switches. The specific fault-tolerance methods used in the system plays an important role in the availability, reliability, and performance of the overall system. In this paper, a survey of fault-tolerant ATM switch fabric architectures is presented. The intention is to give a descriptive overview of the major techniques used to achieve fault-tolerant ATM architecture. The survey covers many important techniques that have been proposed for fault tolerant MINs. It discusses the fault-tolerace approach for computing systems reliability. In addition, it goes through some quantitative techniques to evaluate fault-tolerant systems.

1 Background

The field of fault-tolerant computing has evolved over the past four decades and addressed issues affecting the design of reliable systems. A *fault-tolerant system* is one that can continue to correctly perform its specified operations after the occurrence of specified faults. The amount of fault-tolerance needed depends on the nature of the application and the potential consequences of system failure. Therefore, the field of fault-tolerance is an important concern to many designers.

To achieve fault tolerance, some form of redundancy is required. Redundancy is the addition of information, resources, or time beyond what is needed for normal system operation. There are four possible kinds of redundancy: hardware redundancy, software redundancy, information redundancy, and time redundancy. To build up a fault-tolerant system, normally more than one kind of redundancy is required. The use of one or more of the redundancy techniques in Multistage Interconnection Networks (MINs) is the focus of this work. In particular we discuss in details applying the hardware redundancy and time redundancy techniques in MINs.

MINs are usually constructed from two or more stages of switches. They are typically designed so that data can be sent to the desired destination through the network. MINs offer an attractive way of implementing fast packet switches in communication networks. Therefore, MINs play an important role in building ATM switches which provide considerable capacity [4],[13]. Interconnection networks which continue to provide service even when they contain faulty components are known as *fault-tolerant* networks. With the throughput requirement of the packet switches exceeding several Gbps, it becomes imperative to make them fault tolerant [23],[40],[41].

A MIN is composed of identical basic switching building blocks, called switching elements (SEs), that are interconnected in a specific topology. Thus, a MIN is defined when its topology and the architecture of its SEs are determined. MINs with $logN$ stages have two important properties: a unique path that exists from any input to any output and distinct input-output paths may have common links. This leads to two serious disadvantages. First, the failure of even a single link or switch in the network disconnects several input-output paths that can lead to a lack of fault tolerance and low reliability. Second, an input-output connection may be blocked by a previously established connection (even if the outputs involved are distinct) that can cause poor performance in a random access environment.

To solve the problems of unique path MINs, several multiple path MINs have been proposed [1],[2],[3],[15],[19],[21],[22],[30],[31],[36],[37],[40],[41]. In setting up a connection, multiple path MINs allow an alternate path to be chosen if conflicts arise with other connections or if faults develop in the network. Thus, multiple path MINs provide both better performance and higher reliability than unique path ones. Multiple path MINs have a higher hardware cost than unique path MINs in terms of the number of stages, the number of switches per stage, and/or the size of the switching elements.

There are many examples of such fault tolerant MINs: the Extra-Stage Cube and the F-Network [1],[2], the Modified Omega Network [25],[33], the Augmented Shuffle-Exchange Network [1],[22], the Augmented C-Network and the Merged Delta Network [1], the Chained Baseline Network [37], Itoh's network [15], Tagle and Sharma's network [36], to name few. We briefly describe these networks in the following sections.

2 Design of fault tolerant MINs

As the network size grows, reliability becomes crucial. Any single component failure may crash the network if there is no redundancy. Much of the fault-tolerance work reported in the multistage interconnection networks literature is at the fabric topology level. They all concentrate on the use of multiple paths to reroute around failures. System reliability can be increased by offering multiple alternative paths between each input and output. The system could be over dimensioned initially, so that failures will not affect the quality of service, or it could be gracefully degrading.

There are different techniques to improve the fault tolerance of MINs. In general, fault tolerance in MINs can be achieved by space redundancy or time redundancy. When space redundancy is used, multiple paths are created between inputs and outputs using redundant hardware. This can be achieved by adding an extra stage of switches, varying the switch size, adding extra links, and/or adding extra ports and fault tolerant switching elements. These techniques preserve the *full access* property for the MINs in the presence of some faults. When time redundancy is used, multiple passes are made through the MIN to reach the desired destination. This technique is called *Dynamic Full Access* (DFA) [21]. In the following sections we discuss some examples of switches in which some of these techniques have been applied.

2.1 Addition of extra links

The unique-path property in multistage networks enables routing to be performed in a simple manner. This property, however, presents a problem: the failure of a single component in the network destroys the full access capability of the network. The probability of such a failure can be significant in large networks even when the individual components are designed to be highly reliable. A number of multistage network designs that provide multiple paths between source-destination pairs have been proposed to remedy this problem. Most of the multiple path networks described in the literature are derived from some unique-path networks by introducing additional switches and links. Examples of fault-tolerant MINs that add extra links include Augmented Shuffle-Exchange Network (ASEN), Tzeng's network, data manipulator family like augmented data manipulator and Gamma network. This scheme requires a simple routing algorithm and allows a low level of fault tolerance with reasonable cost. However, the switching throughput of these networks degrades rapidly when the number of faulty components increases.

2.1.1 Augmented Shuffle-Exchange Network (ASEN):

The ASEN was presented by Kumar and Reddy [22]. It is constructed by adding interstage links among switches within each stage in the shuffle-exchange interconnection network. This forms several loops of switches. The advantage of this network is that the input and the output stages do not comprise a hard core of reliability. An individual component failure reduces ASEN performance. Therefore, the switching throughput of the network degrades rapidly when the number of faulty components increases. It does not cause a total network failure by tolerating multiple switch faults while maintaining the ability to connect any source to any destination. There are several versions of the ASEN, depending upon the number of switches included in each loop.

An ASEN with maximum size loops (ASEN-Max) with size 16 is shown in Figure 1. It also shows an arrangement of multiplexers at the input side of the network which makes it possible to tolerate switch faults in stage 0. Switches i in stage $n-1$ are replaced by demultiplexers which provide fault tolerance in stages $n-2$ and $n-1$. Another version of ASEN network is ASEN-2 in which the loops contain exactly two switches.

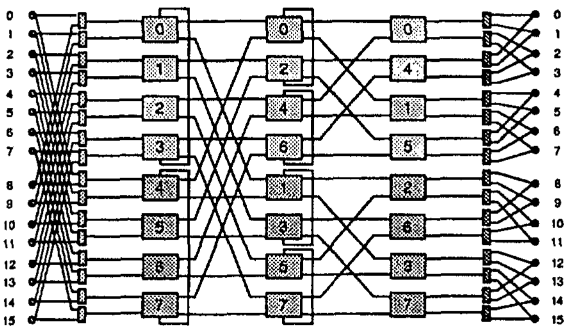


Figure 1: 16 × 16 ASEN with maximum size loops (ASEM-Max).

2.2 Addition of extra stages

Some of the multipath networks are based on the "extra-stage approach" where one or more switching stages are added to a unique-path MIN to create additional paths. Examples of this type of network are the Extra-stage Cube and the Multipath Omega Network.

2.2.1 Extra stage cube

The Extra Stage Cube (ESC) network [1][2] was derived from the generalized cube MIN. It adds an extra stage to the input side of the network together with multiplexers and demultiplexers at the input and output stages, respectively. In addition, dual I/O links to and from the devices using the network are required. Stage n is connected like stage 0. Figure 2 illustrates the ESC for $N = 8$.

Stage n and stage 0 can be enabled or disabled (bypassed). A stage is enabled when its switches are being used to provide interconnection. It is disabled when its switches are being bypassed. Enabling both stages n and 0 provides two distinct paths between any source and destination, at least one of which must be fault-free given any single fault in the network. The ESC is single-fault tolerant and robust in the presence of multiple faults. Extra logic and extra computation time are required to generate the new tags.

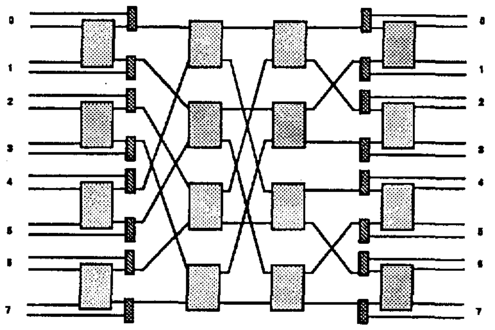


Figure 2: 8 × 8 Extra Stage Cube.

2.2.2 Multipath omega network

The multipath omega network was derived from the omega MIN [31]. This network consists of $\lceil \log_B N \rceil$ identical stages of $B * B$ switch elements, where $N = 2^n$ and $B = 2^b$. Each stage consists of N/B switches, and stages are interconnected by $B * N/B$ shuffle. The multipath omega network has multiple paths because it has redundant switching stages with λ distinct paths between source/destination pair $(1 - \lambda)$ faults can be tolerated.

Routing in the multipath omega network is controlled by routing tags like the procedure used for omega network and generalized shuffle network. Figure 3 shows one possible multipath omega network for $N = 16$ [1].

2.3 Varying the switch size

2.3.1 Augmented C-Network (ACN):

The ACN involves doubling the number of switching input and output ports, while maintaining the same type of switch functions and then adding links to connect selected pairs of switches in adjacent stages. The ACN is derived

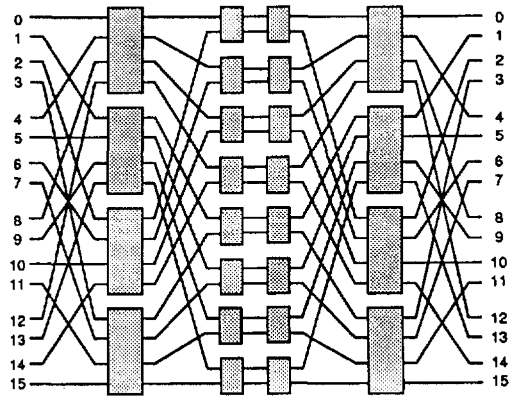


Figure 3: One possible multipath Omega Network for 16 × 16.

from the C-Network by replacing each 2 × 2 switch with a 4 × 4 cross-point switch [33]. Multiple paths between any source-destination pair are obtained using this redundancy. Thus, the ACN is single-fault tolerant to both switch and link failures. The ACN provides 2^n distinct paths between any source and destination (at each stage either switch i or switch $conjugate(i)$ can be used), but most of these paths are disjoint. However, this redundancy will increase the complexity of switches and the routing algorithm. Moreover, extra computation is required to decide routing paths and thus the advantages of self-routing are lost. Different connections in the ACN are shown in Figure 4.

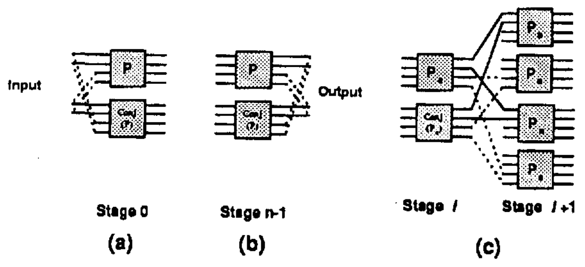


Figure 4: (a) Connections to stage 0 switches in an ACN (b) Connections from stage $n - 1$ in an ACN (c) Connections from a conjugate switch pair in an ACN.

2.4 Addition of extra switches (subswitches)

Providing multiple paths also has been implemented by adding subswitches between switching stages of MIN networks. Itoh's network is a good example; but it has a large number of redundant paths. In addition, substantial amount of redundant switching elements are required with four types (3 × 3, 3 × 2, 2 × 3, 2 × 2), see Figure 11. This leads to a high cost effectiveness and it becomes large as network size increases. Itoh's network is more survivable due to the large additional links/subswitches. The extra subswitches can also be used to avoid contention internally. Itoh's network will be revisited in the next section but more details are found in [15].

2.5 Redundancy in time

Redundancy in time can be applied to MIN with end-around connections as shown in Figure 5.

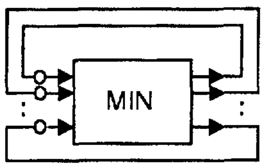


Figure 5: MIN with End-Around Connections.

In this figure, multiple passes through the network can be allowed to route data from the input ports to the output ports. For example, if input port x can not route directly to output port y in one pass through the network, it might be possible to route from x to z in one pass, and then from z to y in the second pass. The ability to route from any input to any output in a finite number of passes is called the dynamic full access (DFA) property [21].

The DFA property was suggested as a criterion for fault-tolerance of network based on 2×2 switching elements. The DFA method can not work if the switches in the first or in the last stage compromise faults that can not be tolerated. Thus, even MINs with multiple connections from input to output can lose the full access property in the presence of multiple faults. A third approach that applies both space and time redundancies to achieve fault-tolerance in MINs was suggested [20]. This approach was applied to four multiple path MINs. These are the Augmented Shuffle-Exchange Network (ASEN-2), MD-Omega Network, Augmented C-Network (ACN), and INDRA Network.

2.6 Cost-effective hardware redundancy

Almost all of the implementations of fault-tolerant MINs introduce redundancy in the network. Most of these solutions are expensive in terms of number of extra switches per stage, and/or the size of the switching elements. Moreover, these solutions have a high hardware complexity that need complex routing algorithms.

A fault-tolerant high-performance self routing optical ATM switch (SEROS) was proposed [12]. SEROS can be used with any multistage interconnection network that uses 2×2 switches with cost-effective hardware redundancy.

The proposed 2×2 switch architecture is an extension of existing switches that achieve straight and exchange connections. Fault-tolerance is increased by introducing buffers in the switch that act as queues and store the packets that cause contention, hence preventing their loss. Additional fault-tolerance is provided by augmenting the switch with two circuits. These are responsible for detecting faults in the network and correcting them. They can also route data to other switches in case the main module of the switch fails.

The top level breakdown of the complete switch is shown in Figure 6. It consists of three circuits. The first is the *bypass circuit* which passes the data to the next stage when the current switch fails. The second is the error detection and correction circuit responsible for detecting errors in the data and removing them before being processed by the next routing switch. The routing switch examines the bits in the address field of the data and routes it to the appropriate output. In cases of contention, it may loop the data within the same switch.

The first two circuits are used only for handling faults in the network. If the routing switch of stage i fails to respond correctly, its bypass circuit routes the data to the next stage $(i + 1)$ where routing decisions are taken.

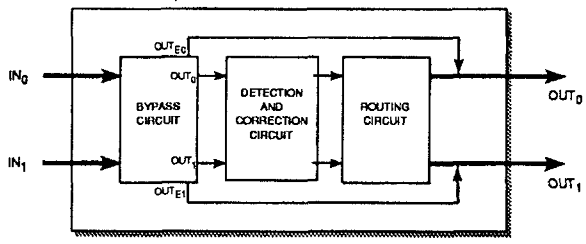


Figure 6: Block Diagram of the 2×2 Switch.

An example for comparing the output of a faulty switch to a properly working switch is shown in Figure 7. In part (a) of this figure, the data are routed to the lower output link based on the routing bits (destination address). Note that there are no bits in the error field and the most significant bit (1) is dropped from the routing bits. Part (b) of Figure 7 illustrates the operation of a faulty switch. Note that the input is replicated to both high and low outputs of the switch with the error field containing 0 and 1, respectively. The routing bits remain unchanged.

3 Fault tolerant ATM fabrics based on MINs

Due to their attractiveness, banyan networks are popular in the network community as the need to design ATM switches arise. This type of switch is practical for large network size. Also, they are not as expensive as those of the N^2 disjoint path types. In this section, several banyan-based multistage interconnected examples are presented to show the evolution of this switch fabric type. The switch architectures are different from each other with an ultimate objective to increase the network fault tolerance, resolve

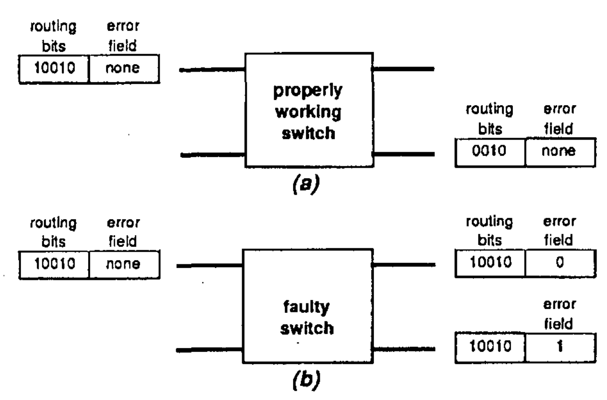


Figure 7: Comparison between Traffic Generated by a Faulty and a Properly Working Switch.

the internal blocking problem, and provide high performance switch fabric. Few of the commonly used architectures that support fault-tolerance are also discussed.

3.1 Baseline Banyan network

This is the simplest banyan-based network. There is at most one path connecting any input to any output lines. Figure 8 shows an 8 × 8 baseline network that consists of three stages. Each stage has four 2 × 2 switch elements (SE). To route a cell in a SE using the self routing technique, the SE checks the bit in the destination address that corresponds to the stage number (starting with the most significant bit). If the bit is equal to 0, the SE routes the cell using the top output port. Otherwise, it uses the lower output port [26]. The dashed line in Figure 8 shows the exact route of an ATM cell from input port 1 to output port 5.

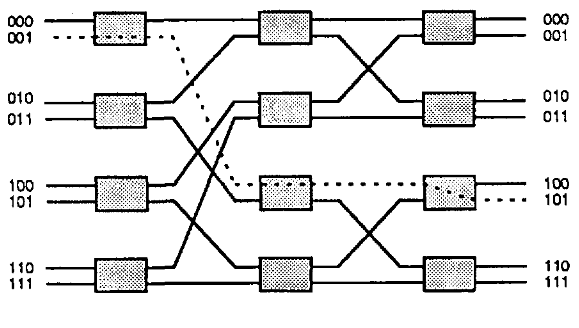


Figure 8: 8× 8 Baseline Banyan Network.

The baseline network is fast and efficient since it uses self routing technique. However, it is a blocking network since it is possible that some cells require the same switch element (link) to go through. Then, one of the cells passes to the destination and the other cell is either lost or deflected to the wrong destination. In addition, if one switch element is faulty, all cells routed through that switch are lost since there is no alternative path to go through. This means that all switch elements have to function correctly in order to guarantee the proper function of the whole network.

3.2 Parallel Banyan network

This architecture consists of two parallel baseline networks connected using input and output routers as shown in Figure 9. Each baseline network is called a plane. Depending on the function of the routers, this architecture could double the throughput of a single baseline by routing two cells concurrently in two planes, or could make the switch fabric more fault tolerant by broadcasting the cell to two planes. Each plane is still a blocking network. The number of disjoint paths is doubled because of doubling the hardware of the baseline network and adding the input and output routers. The routing procedure is the same as that of a single baseline network. In Figure 9, the dashed lines show the only two possible routes from input port 1 to output port 5.

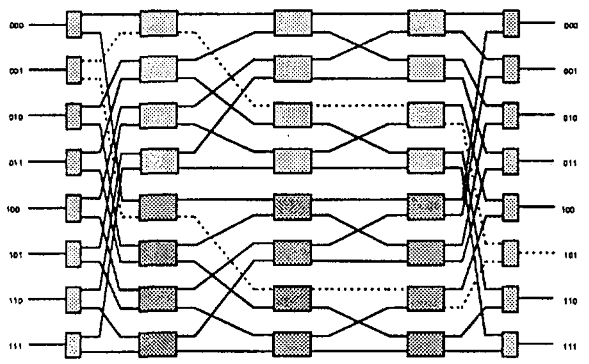


Figure 9: 8 × 8 Parallel Banyan Network.

3.3 Benes' network

This architecture consists of two baseline networks mirrored to each other sharing the middle stage as shown in Figure 10. This network has four possible routes from any input to any output line(s). The blocking problem and the fault tolerance of switch elements are solved in the first $(\log_2 N) - 1$ stages. However, the remaining $\log_2 N$ stages still have the blocking problem and if there is a fault in any of the switch elements, cells will be lost. Each cell has to be routed through $2(\log_2 N) - 1$ stages. It is almost double the time it takes through the previous networks. The routing procedure through the network is divided into two sections. The first $(\log_2 N) - 1$ stages use the baseline routing. If there is an internal contention through a switch element, it deflects one cell to the wrong output port of the SE. The remaining stages use the shuffle-exchange routing algorithm . If there is an internal contention through a SE, it either discards one cell or deflects it to the wrong output port of the SE resulting in a wrong delivery [32].

Generally, the shuffle-exchange routing algorithm is as follows: the source sends a binary pattern representing the destination number into the network. As the pattern moves through the network, each stage examines a different bit to determine switch settings. Stage 1 uses the most significant bit, stage 2 the middle bit, and stage 3 the least significant bit. When a request arrives on either input of a switch, it is routed to the upper output if the controlling bit is a 0 and to the lower output if the bit is a 1.

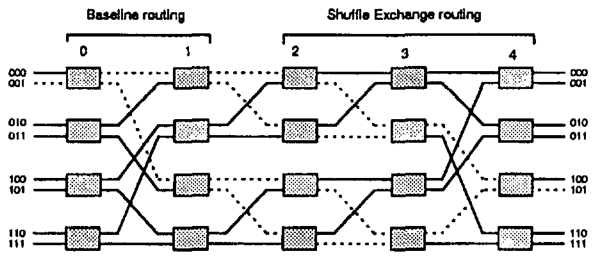


Figure 10: 8 × 8 Benes' Network.

The dashed lines in Figure 10 show the four possible routes from input port 1 (001) to output port 5 (101). The following steps describe the normal route from input port 1 ($S = s_2s_1s_0 = 001$) to output port 5 ($D = d_2d_1d_0 = 101$) if there is no internal contention or faulty switch elements in the path:

1. The cell in SE(1) of stage(0) has two routes to the next stage (stage (1)). Either it goes to SE(1) or SE(3) depending on the status of those swich elements and the internal contention. Following the baseline routing algorithm, SE(1) routes the cell to SE(3) of stage(1) since the most significant bit of the destination address (d_2) is equal to 1.

2. The cell in SE(3) of stage(1) has two routes to the next stage (stage(2)). Either it goes to SE(3) or SE(4). Normally, the cell goes to SE(3) since the second bit of the destination address (d_1) is equal to 0.

3. The cell in SE(3) of stage(2) has only one route to the next stage (stage(3)) using the shuffle-exchange routing algorithm. The cell goes to SE(4) since the most significant bit of the destination address (d_2) is equal to 1.

4. The cell in SE(4) of stage(3) has only one route to SE(3) of the next stage (stage(4)) since the second bit of the destination address (d_1) is equal to 0.

5. Finally, SE(3) of stage(4) routes the cell to the output port 5 since the least significant bit of the destination address (d_2) is equal to 1.

The other possible routes are as follows:

1. If SE(3) of stage(1) is faulty, the cell will be routed using the following path: SE(1) of stage(0), SE(1) of stage(1), SE(1) of stage(2), SE(2) of stage(3), SE(3) of stage(4), then to the output port 5.

2. If SE(3) of stage(2) is faulty, the cell will be routed using the following path: SE(1) of stage(0), SE(3) of stage(1), SE(4) of stage(2), SE(4) of stage(3), SE(3) of stage(4), then to the output port 5.

3. If SE(3) of stage(1) and SE(1) of stage(2) are both faulty, the cell will be routed using the following path: SE(1) of stage(0), SE(1) of stage(1), SE(2) of stage(2), SE(2) of stage(3), SE(3) of stage(4), then to the output port 5.

Note that all four paths start from the same switch element and merge to the same last switch element. Hence, the first and the last stages have to work correctly in order for the network to route the cells to the proper destinations.

3.4 Itoh's network

This architecture consists of a modified version of the baseline network with added subswitches between stages in order to increase the number of paths from any input to any output line(s) as shown in Figure 11. In an 8 × 8 Itoh's network, there are five possible paths from any input to any output line(s). This is a non-blocking network with one exception at the last stage of the network when there is more than one cell routed to the same destination.

The routing algorithm in Itoh's network is essentially the same as the one used in the baseline network [15]. In this routing scheme, assume that the network inputs are labeled as $S = s_{n-i}s_{n-2}\cdots s_0$ and outputs as $D = d_{n-i}d_{n-2}\cdots d_0$. At stage i, bit d_{n-i} of the destination address is used to route ATM cells. If $d_{n-i} = 0$, the cell will be routed to the upper port of the SE; If $d_{n-i} = 1$, it will be routed to the lower port. When the destined port is blocked due to an internal contention, link failure, or SE failure in the next stage, the cell will be routed through the redundant path to the subswitch of rank-1 ($R1$ subswitch). The same d_{n-i} bit will be used for the routing in the subswitch. If the destined port of the subswitch $R1$ is also blocked, the cell will be routed to the subswitch of rank-2 ($R2$ subswitch). The cell can be routed through as many subswitches within the maximum rank of the stage as needed [15].

Figure 11 shows that in the worst case, the cell would be routed through 5 links in an 8 × 8 network to go from input port 1 (001) to output port 5 (101).

The first path is the normal route based on the baseline network routing. The cell traverses through the normal route in the following order: SE(1) of stage(1), SE(3) of stage(2), SE(3) of stage(3), and then to the output port 5.

The other four routes are as follows:

1. If the link between SE(1) of stage(1) and SE(3) of stage(2) is faulty, then SE(1) of stage(1) routes the

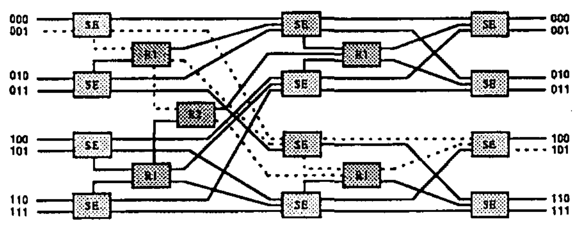


Figure 11: 8 × 8 Itoh's Network.

cell to the upper $R1$ subswitch of stage(1), SE(3) of stage(2), SE(3) of stage(3), then to the output port 5.

2. If SE(3) of stage(2) is faulty, the cell will be routed to the upper $R1$ subswitch of stage(1), then $R2$ subswitch of stage (1), the lower $R1$ subswitch of stage(2), SE(3) of stage(3), then to the output port 5.

3. If the link between SE(3) of stage(2) and SE(3) of stage(3) is faulty, then SE(1) of stage(1) routes the cell to SE(3) of stage(2), then to the lower $R1$ subswitch of the same stage, SE(3) of stage(3), then to the output port 5.

4. If the link between SE(1) of stage(1) and SE(3) of stage(2) is faulty and the link between SE(3) of stage(2) and SE(3) of stage(3) is faulty, then SE(1) of stage(1) routes the cell to the upper $R1$ subswitch of stage(1), SE(3) of stage(2), then to the lower $R1$ subswitch of the same stage, SE(3) of stage(3), then to the output port 5.

Notice that all five paths start from the same switch element and merge to the same last switch element. The first and last stages have to work correctly at all times in order for the network to route the cells to the proper destinations. The intermediate stages can have five routes for any input to any output lines. The problems of internal contention and the single faulty link or faulty switch element are resolved at the expense of adding subswitches. There is an inherent problem with this architecture. Cells might arrive out of sequence at the destination port, which is prohibited in ATM. Another basic problem with this architecture is the use of different switch elements' size and the non-modularity. In a sense, the rank subswitches have different sizes at each stage.

3.5 Tagle & Sharma's network

This architecture consists of two baseline networks similar to that of the parallel baseline network. The switch elements are modified (4×4 switch elements) to allow routing from one baseline network (plane) to the other, if required [36]. In an 8 × 8 network, there are 8 possible paths from any input to any output ports as shown by the dotted lines in Figure 12. Notice that the gray and the dark switches represent the switches of plane 0 and plane 1 of the banyan networks, respectively.

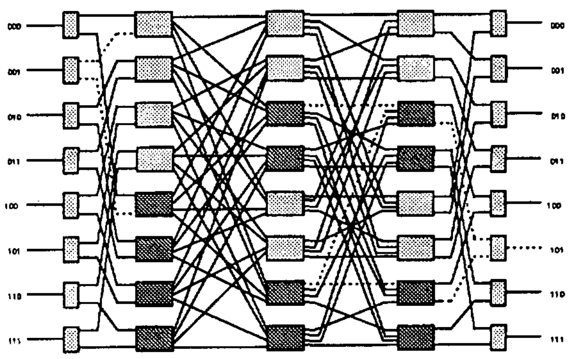


Figure 12: 8 × 8 Tagle & Sharma's Network.

This network is simple and resolves the internal blocking problem within the network stages and the problem of faulty links and/or switch elements within a plane. The last stage can route only two cells to one output line. So that when cells are going to the same destination, only two would reach the destination line and the remaining cells are lost. This switch fabric is fast and it could double the throughput of a single baseline by routing two cells concurrently. But if there is a faulty switch or congestion in one of the planes, one cell would be lost.

The routing procedure is the same as that of the baseline network with the exception that if there is an internal contention in one of the SE's in the first $(\log_2 N) - 1$ stages, it routes the cell to the next stage in the other plane. Take the simplest case of a point to point routing with no fault encountered. Then the cell stays in plane 0 and the destination D is reached as in the case of a baseline network. If a fault is encountered, then the cell will be sent to the next stage in the other plane (plane 1). The routing tag (RT) is simply the destination output D and cells can be sent to planes 0 and 1 by appending one bit to the front of the RT ($a = 0$ or $a = 1$). Therefore, cells can be moved to either plane to bypass faults where plane 1 would be initially on stand-by mode. The advantage of using alternate plane is to free up the other plane which could double the throughput. Plane 1 also can be used to pass the high priority traffic using the appended bit (a) to designate priority for cells.

3.6 Baseline-tree (B-tree) network

This architecture is based on multiple interconnected baseline networks to provide multiple paths from any input to any output line(s) with minimum cell loss. Figure 13 shows an 8 × 8 network connection and the possible eight paths from input port 1 to output port 5 (dashed lines). This switch fabric is fast since cells have to go through only $\log_2 N$ stages to reach the output line. It basically provides N different routes and $2(\log_2 N)$ access to each output port. In other words, if eight cells are routed to the same destination, only two cells will be lost and the remaining cells will reach the destination port concentrator [24]. Figure 13 shows the connection of the network stages, (six stages: each stage consists of four 4×4 switch elements), and output port 5 only. For simplicity, the remaining output ports are not shown.

Each switch element has four inputs, two formal outputs, and two redundant outputs. The switch element is capable of routing four different cells to four different routes if there is no fault(s) in any of the switch elements of the next stages. In addition, the routing destination bit is 0 for two cells and 1 for the other two cells since each SE has two normal output links and two redundant output links.

The following steps show an example of the normal route from input port 1 (001) to output port 5 (101) (see Figure 13):

1. The cell in SE(0) of the stage on the first row and the first column has two possible routes: horizontal and/or vertical. Normally, SE(0) routes the cell to the next horizontal stage. In this case, SE(0) routes the cell to SE(2) of the next horizontal switch since the most significant bit of the destination address is equal to 1.
2. SE(2) routes the cell to SE(2) of the next horizontal stage since the second destination bit is equal to 0.
3. SE(2) routes the cell to output port 5 through the horizontal link since the first destination bit is equal to 1.

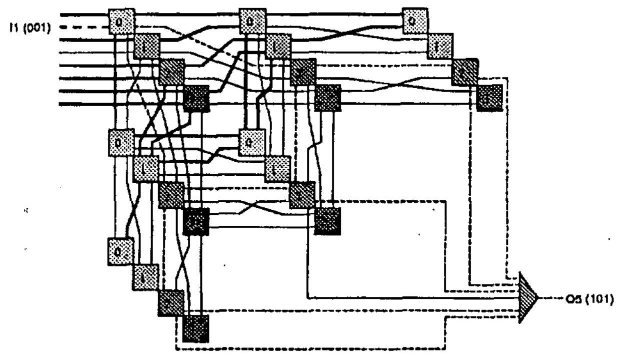


Figure 13: 8×8 Baseline-Tree Network.

The other seven routes from input port 1 (001) to output port 5 (101) are:

1. If there is a fault in the horizontal link of SE(2) of the stage on the first row and the third column, then the route would be the same as the normal route with the exception of the last routing link. SE(2) routes the cell to output port 5 through the vertical link.
2. If there is a fault in SE(2) of the stage (row 1, column 3), then SE(2) in the previous stage (row 1, column 2) routes the cell to SE(2) of the stage in (row2, column 2) using the vertical link. Finally, SE(2) routes the cell to output port 5 through the horizontal link.
3. If there is a fault, in addition to the fault in the previous case, in the horizontal output link of SE(2) of the stage located in (row 2, column 2), then the route would be the same as the previous case with the exception of the last routing link. SE(2) routes the cell to output port 5 through the vertical link.
4. If there is a fault in SE(2) of the stage located in (row 1, column 2), then SE(0) routes the cell to SE(2) of the stage in (row 2, column 1) using its vertical output link. After that, SE(2) routes the cell to SE(2) of the stage in (row 2,column 2). Finally, SE(2) routes the cell to output port 5 through the horizontal link.
5. If there is another fault, in addition to the previous case, in the horizontal output link of SE(2) in (row 2, column 2), then the route would be the same as the previous route with the exception of the last routing link. SE(2) routes the cell to output port 5 through the vertical link.
6. If there is another fault, in addition to the first case, in SE(2) of the stage in (row 2, column 2), then SE(0) routes the cell to SE(2) of the stage in (row 2,column 1). After that, SE(2) routes the cell to SE(2) of the stage in (row 3, column 1). Finally, SE(2) routes the cell to output port 5 through the horizontal link.
7. If there is another fault, in addition to the previous case, in the horizontal link of SE(2) of the stage in (row 3, column 1), then the route would be the same as the previous route with the exception on the last routing link. SE(2) routes the cell to output port 5 through the vertival link.

This capability of having eight routes from any input port to any output port at the same time is the main advantage of the B-Tree architecture. This switch fabric is fast since cells have to go through only $\log_2 N$ stages to reach the output port. In addition, it could double the throughput easily by routing two cells concurrently in two disjoint paths. The other six routes are to handle congestion and faulty switch elements. The routing procedure is the same as that of the baseline network with the exception that if there is an internal contention in one of the SEs, it routes the cell to the next stage using the redundant output links.

Table 1 summarizes the performance and the fault tolerance of the six banyan-based switch fabrics described in this paper. All of the switch fabrics are 8×8 networks. To come up with the values for the number of switching elements (SEs) for the six banyan-based networks, the following formulas were used:

$$SE_{Baseline} = \frac{N}{2} \cdot n$$

$$SE_{Benes} = \frac{N}{2} \cdot (2n - 1)$$

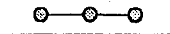
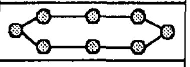
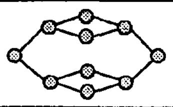
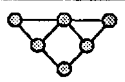
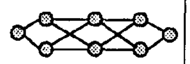
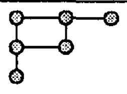
Switch Fabric Name	Min. # of Stages	Max. # of Stages	# of SE	# of routes	Routing Scheme	Channel graph
Baseline Banyan Network	3	3	12	1	Baseline routing	
Parallel Banyan Network	3	3	24	2	Baseline routing	
Benes' Network	5	5	20	4	Baseline and Shuffle-Exchange routing	
Itoh's Network	3	5	17	5	Baseline routing	
Tagle & Sharma's Network	3	3	24	8	Baseline routing	
Baseline-Tree Network	3	3	24	8	Baseline routing	

Table 1: Summary of the performance and the fault tolerance of six Banyan-Based Networks.

$$SE_{Itoh} = N \cdot (n-1) + 1$$

$$SE_{Tagle\&Sharma} = SE_{parallel} = SE_{B-Tree} = N \cdot n$$

where N is the network size and n is the stage number in the network.

4 Reliability analysis of MIN

The fault-tolerance of network topologies are sometimes compared by evaluating reliability measures. Three reliability measures are of particular interest in the context of MIN: *terminal reliability*, *broadcast reliability*, and *network reliability*. Basic terminal reliability, also called two-terminal reliability, addresses the probability that a given source-destination pair has at least one fault-free path between them. This considers the smallest subnet of the pairings between sources and destinations. Broadcast reliability expresses the ability to perform a broadcast of data from a given input terminal to all the output terminals of the network. This reliability measure is sometimes referred to as the Source-to-Multiple Terminal (SMT) reliability. Whereas network reliability, also called all-terminal reliability, addresses the probability that at least one path exists between each source and every destination and properly establishes the reliability of the MIN.

For the evaluation of a MIN reliability, several algorithms exist for the efficient computation terminal reliability expressions. Varma and Reghavendra [39] derived expressions for terminal reliability and broadcast reliability for several multipath MINs. Cheng and Ibe [9] also used similar measures to evaluate several MINs. Mohapatra, Yu and Das [28] analyze MIN reliability considering the cubic allocation algorithm. Several others have evaluated and compared multipath MINs using various reliability measures [8],[10],[14],[16],[27],[36].

A powerful tool for analyzing complex probabilistic systems is the Markov process model. For more details and examples in understanding how the Markov models are used to derive the reliability of a system, refer to [17] and [35].

Reliability analysis of MINs usually relies on approximate techniques since exact dependability computation of these MINs with imperfect coverage is known to be an NP-complete problem. Software packages like HARP or SHARPE are powerful tools useful for reliability analysis. These packages are primarily based on enumeration of Markov chains [7],[34].

Some examples that illustrate obtaining the exact reliability of a MIN as well as extra-stage MIN are presented. For parallel banyan network, the reliability can be obtained from the the network's channel graph which is shown in Table 1. There are only two redundant paths between any source and destination. Therefore, the terminal reliability of the parallel banyan network is given by:

$$TR_{Pbanyan} = 1 - (1 - R_{path})(1 - R_{path})$$

Hence,

$$TR_{Pbanyan} = 2R_{path} - (R_{path})^2$$

There are n 2×2 switches connected in series on any path in the parallel banyan network, so

$$R_{path} = R_{2x2}^n$$

From the above equations,

$$TR_{Pbanyan} = (2R_{2x2})^n - (R_{2x2})^{2n}$$

Since we have the multiplexers and the demultiplexers in series with every path in parallel banyan network, the terminal reliability of the parallel banyan network will be:

$$TR_{Pbanyan} = R_{demux}[(2R_{2x2})^n - (R_{2x2})^{2n}]R_{mux}$$

To calculate the terminal reliability of Tagle's network, we can observe from the channel graph in Table 1 that a cell always has two switching elements to go to any stage in the network. Hence, we have switching element pairs in series. The reliability for the switching element pairs is simply the product of the reliability of each pair of 4×4 switching element.

The terminal reliability of Tagle network is therefore as follows:

$$TR_{Tagle} = (TR_{stage})^n$$

where,

$$TR_{stage} = 2R_{4x4} - (R_{4x4})^2$$

Since we have the multiplexers and the demultiplexers in series with every path, the terminal reliability of Tagle's network is:

$$TR_{Tagle} = R_{demux}(2R_{4x4} - (R_{4x4})^2)^n R_{mux}$$

Blake and Trivedi [5] analyzed the shuffle-exchange multistage interconnection network (SEN) which is just one network in a large class of topologically equivalent MINs. Also, they have obtained bounds on the reliability for the Augmented Shuffle-Exchange Network (ASEN). Figure 14 shows an 8 × 8 ASEN which can be achieved by adding additional stage to the 8 × 8 SEN.

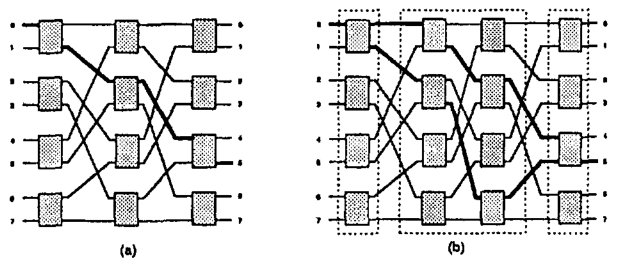


Figure 14: (a) 8 × 8 SEN (b) 8 × 8 ASEN.

The exact reliability analysis was obtained under two assumptions. First, the network is considered to be operational as long as every source can communicate with each destination. Second, the switching elements are identical and they are either working or failed.
Since the SEN is a unique-path MIN, the failure of any switch will cause system failure, and from the reliability point of view, $(N/2)(\log_2 N)$ switching elements are in series. Therefore, the reliability of an $N \times N$ SEN can be written as:

$$R_{SEN}(t) = [r_{SE}(t)]^{N/2 \log_2 N},$$

where $r_{SE}(t)$ is the reliability of the basic switching element.

For the ASEN network, increasing the number of stages will increase the possible number of configurations to achieve the full connectivity. Since the network complexity of the ASEN is $(N/2)(\log_2 N+1)$, presenting the configurations of a ASEN as a continuous-time Markov chain will produce the states of chain to be as $[(N/2)(\log_2 N + 1)]$-tuples, where each position of the tuple is either a 1 or 0 corresponding to the up or down state of the respective SE.

Modeling the system's time-to-failure behavior using the continuous-time Markov chain approach has a major problem which is the exponential growth of the state space as the network size increases. This exponential growth occurs because of the need to consider the operational status of each SE in each state. For example, for an 8 × 8 ASEN, which has 16 SEs, 2^{16} possible states have to be considered. Since the first and the last stages must function properly for the network to function, only 2^8 possible configurations can be considered. The reliability of a SE can be given by:

$$R_{SE}(t) = e^{-\int_0^t \lambda(t)dt}.$$

The reliability of the 8 × 8 ASEN can be written as:

$$\begin{aligned} R_{ASEN}(t) &= 2e^{-12\int_0^t \lambda(t)dt} + 4e^{-14\int_0^t \lambda(t)dt} \\ &\quad -8e^{-15\int_0^t \lambda(t)dt} + 3e^{-16\int_0^t \lambda(t)dt} \\ &= 2[r_{SE}(t)]^{12}+4[r_{SE}(t)]^{14}-8[r_{SE}(t)]^{15}+3[r_{SE}(t)]^{16}. \end{aligned}$$

As network size increases, modeling the reliability of the MIN networks using Markov chains will become more complex. Therefore, approximation techniques can be used for determining the reliability of larger MINs.

Blake and Trivedi considered the approximation approach for determining the reliability of larger ASEN networks. To obtain a lower bound on ASEN, it was observed that the switch elements in the intermediate stages of a ASEN can be divided into two subnetworks. Therefore, a lower bound on $R(t)$ was obtained under two assumptions to guarantee the full access connections:

1. none of the switches in the first and last stages had failed, and
2. at least one of the two complete subnetworks which decompose the ASEN and are connected by the two extreme stages was fault-free.

The intermediate stages can be modeled as a system consisting of a parallel arrangement of two series subsystems each with $(N/4)(\log_2 N-1)$ switches. The lower bound of reliability can be easily obtained using reliability block diagram. As shown in Figure 14 (b), the system is composed of three series subsystems, the first and the last are series subsystems. The middle subsystem is a parallel-series subsystem. The lower-bound can be written as:

$$R > (r_{SE})^N(1 - (1 - (r_{SE})^{N'})^2)$$

where $N = 2^n$ is the number of input terminals, n is the number of stages and $N' = (N/4)(\log N - 1)$ is the number of switches in each subnet. The upper bound on $R(t)$ can be obtained by observing that full access is lost if a switch in the first or last stage fails or if a pair of switches that occupy corresponding positions in the two subnetworks fail. Since there are many combinations of failed switches which will cause the network to fail other than such pairs, this consideration alone overestimates $R(t)$. Hence,

$$R < (r_{SE})^N[(1 - (1 - r_{SE})^2]^{N'}.$$

These bounds are tight for small networks but diverge for large networks. Thus, Menezes and Bakhru [27] motivated this attempt to obtain better bounds on estimating $R(t)$ for the SEN with wrap-around connections as well as Kumar and Wang [20]. Kumar and Wang studied the upper bounds for the reliability of four multiple path MIN under DFA [21]. These MINs are the ASEN-2, MD-Omega Network, CAN, and INDRA Network. They found that the MD-Omega network which uses a minimum amount of hardware redundancy to provide two connections from each source to the MIN and to each destination from the MIN, shows the most gain in reliability when time redundancy is used. The MD-Omega network has 2×2 switch as its basic element, but is almost as reliable as another fault tolerant MIN, the ASEN, which uses a 3×3 element, when multiple pass routing is used.

5 Conclusion

Due to the need of increased network bandwidth, different switching techniques have been adopted. Each of these switching techniques serves a particular application. ATM switching technology came to unify all switching techniques with a flexible environment. Therefore, ATM has been the development focus of many research groups around the world.

In this paper, an overview of the basic fault tolerant ATM switch fabric architectures is presented. We concentrated on ATM switches that use multistage interconnection networks (MINs). In particular, we investigated the different existing types of architectures based on the banyan network. Most of the improvements in the ATM switch fabric are to increase the switch performance, improve network fault tolerance and decrease the rate of cell loss.

For the evaluation of a MIN reliability, several algorithms exist for the efficient computation terminal reliability expressions. This paper covers an example that illustrates obtaining the exact reliability of a MIN as well as extra-stage MIN.

In the appendix, a summary of the basic evaluation techniques used in fault tolerance is presented.

6 Appendix: Fault-tolerance evaluation techniques

Two major methods can be used for evaluating fault-tolerant systems: qualitative and quantitative . Qualitative evaluation techniques are typically subjective in nature and describe the benefits of one system over another. Flexibility of a system, its technology dependence and the degree to which the fault tolerance techniques are transparent to the user are all examples of qualitative evaluation techniques. Quantitative measures produce numbers that can be used to compare different approaches. Reliability modeling, availability measurement, failure rate, mean time to failure and mean time to repair are all quantitative techniques. Cost and weight are also examples of quantitative measures that can be used to evaluate systems. The purpose of this appendix is to summerize some quantitative techniques that are used in the evaluation of fault-tolerant systems. [18][29][35]

6.1 Quantitative measures

The quantitative measures of interest are:

6.1.1 Reliability function and failure rate

The *reliability* $R(t)$ of a system is the probability that a system will not fail within a given time t provided that the system was performing correctly at time 0. The reliability function can be written as:

$$R(t)= e^{-\lambda t} \tag{1}$$

where λ is the constant *failure rate*. The *failure rate* (λ) is defined as the expected number of failures of a type of devices or system per a given time period. For more details refer to [17][14].

6.1.2 Mean time to failure ($MTTF$)

The $MTTF$ is the expected time that a system will operate before the first failure occurs given successful startup of the system at time zero. It is defined in terms of the reliability function as

$$MTTF = \int_0^{\infty} R(t)dt = \int_0^{\infty} e^{-\lambda t}dt = \frac{1}{\lambda} \tag{2}$$

6.1.3 Mean time to repair ($MTTR$)

$MTTR$ is the expected time to repair a failed system or subsystem. $MTTR$ is related to *repair rate* μ which is the average number of repairs that occur per time period.

$$MTTR = \frac{1}{\mu}. \tag{3}$$

The units of the repair rate are normally numbers of repairs per hour.

6.1.4 Mean time between failures ($MTBF$)

$MTBF$ is the mean time between failures in a system with repair, and is derived from a combination of repair and failure processes.

$$MTBF = MTTF + MTTR. \tag{4}$$

6.1.5 Availability $A(t)$

The *availability* $A(t)$, is the amount of time the system is working when compared to the measured lifetime of the system. Availability is given by:

$$A = \frac{MTTF}{MTTF + MTTR} = \frac{1}{1 + \frac{\lambda}{\mu}} \tag{5}$$

where, $MTTF = \frac{1}{\lambda}$ and $MTTR = \frac{1}{\mu}$.

6.1.6 Maintainability $M(t)$

The *maintainability* $M(t)$ is defined as the probability that a failed system will be restored to operation within time t, given that it was in a failed state at time 0. The maintainability can be measured using the following expression:

$$M(t) = 1 - e^{-\lambda t} \tag{6}$$

6.2 Reliability modeling

The most popular reliability analysis techniques are the analytical approaches. The two most commonly used approaches are the combinatorial modeling and Markov modeling. Most frequently, reliability evaluation involves a series or parallel combination of independent systems. The reliability of a system is generally desired in terms of the reliability of the individual components of the systems. [17][35]

6.2.1 Series systems

If $R_i(t)$ is the reliability of module i and if the modules are assumed independent, then the overall reliability of a series system is:

$$R_{series}(t) = \prod_{i=1}^{n} R_i(t) \tag{7}$$

Consider that each individual component satisfies the exponential failure law with a constant failure rate λ_i then,

$$R_{series}(t) = e^{-\sum_{i=1}^{n} \lambda_i t} = e^{-\lambda_{system} t} \cdot e^{-\lambda_n t}$$

6.2.2 Parallel systems

The parallel system reliability can be found as follows:

$$R_{parallel}(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)). \tag{8}$$

6.2.3 Series and parallel combinations

In general, systems are typically combinations of series and parallel systems. A reliability block diagram that contains both series and parallel structures can be reduced to a single diagram by replacing each of the parallel portions of the system with an equivalent, single element that has the same reliability as the parallel structure.

6.2.4 M-of-N systems

M-of-N systems are a generalization of the parallel system. In the ideal parallel system, only one of N modules is required to work for the system to function. In the M-of-N system, M of the total identical modules are required to function for the system to function. The reliability can be written as:

$$R_{M-of-N}(t) = \sum_{i=0}^{N-M} \binom{N}{i} R^{N-i}(t) \cdot (1 - R(t))^i, \tag{9}$$

$$\text{where} \binom{N}{i} = \frac{N!}{(N-i)!\, i!}.$$

References

[1] D. P. Agrawal, G. B. Adams, and H. J. Siegel, "A Survey and Comparison of Fault-Tolerant Multistage Interconnection Networks", *Computer,* pp.14-27, Jun. 1987.

[2] G. B. Adams and H. J. Siegel, "Modifications to improve the fault tolerance of extra stage cube interconnection network", *Proc. of the International Conf. on Parallel Processing,* pp.169-173, Aug. 1984.

[3] G. B. Adams and H. J. Siegel, "A Survey of Interconnection Networks," *Computer,* Globecom 84, pp. 105-113, 1984.

[4] H. Ahmadi and W. Denzel, "A Survey of Modern High Performance Switching Techniques ", *IEEE JSAC,* vol. 7, no.7, Sept. 1989.

[5] J. T. Blake and K. S. Trivedi, "Multistage Interconnection Network Reliability," *IEEE Trans. on Computers,* vol. C-38, pp. 1600-1604, Nov. 1989.

[6] C. Botting, S. Rai, and D. P. Agrawal, "Reliability computation of Multistage Interconnection Networks," *IEEE trans. on Computers,* pp. 138-145, April 1989.

[7] J. B. Dugan, K. S. Trivedi, "The Hybrid Automated Reliability Predictor," *AIAAJ. Guidance, Control and Dynamics,* vol.9, no.3, pp.319-331, May-June 1986.

[8] J. Chin and J. B Dugan, "Analysis of a Multistage Interconnection Network using Binary Decision Diagrams (BDD)," *IEEE 15th symposium on Reliable Distributed Systems,* 1996.

[9] X. Cheng and O. C. Ibe, "Reliability of a class of Multistage Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 3, no. 2, pp. 241-246, March 1992.

[10] C. J. Colbourn, J. Devitt, D. Harms, and M. Kraetzl, "Assessing Reliability of Multistage Interconnection Networks," *IEEE Trans. on Computers,* vol. 42, no. 10, pp. 1207-1221, Oct. 1993.

[11] U. Grag and Y. Huang, "Decomposing Banyan networks for performance analysis", *IEEE Trans. on Computers,* vol. c-37, no. 3, pp. 371-376, March 1988.

[12] M. Guizani and A. Memon, "SEROS : A Self Routing Optical ATM Switch", *International Journal of Communication Systems,*vol. 9, no. 2, pp.115-125, March/April 1996.

[13] M. Guizani and A. Rayes, *ATM Switching and Networking*, McGraw-Hill, 1998.

[14] C. L. Hwang, F. Tillman, and M. H. Lee, "System Reliability Evaluation Techniques for Complex/Large Systems- A review," *IEEE Trans. on Reliability,* vol. R-30, no.5, pp. 416-425, Dec. 1981.

[15] A. Itoh, "A Fault-Tolerant Switching Network for B-ISDN," *IEEE Journal on Selected Areas in Comm.*, vol. 9, no. 8, pp. 1218-1226, Oct. 1991.

[16] M. Jeng and H. Siegel, "Design and Analysis of Dynamic Redundancy Networks," *IEEE Trans. on Computers,* vol. 37, no. 9, pp. 1019-1029, Sept. 1989.

[17] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley, Reading MA, 1989.

[18] K. Kant, *Introduction to Computer System Performance Evaluation*, McGraw-Hill, Inc., 1992.

[19] V. P. Kumar and A. L. Reibman, "Failure dependent performance analysis of a fault tolerant multistage interconnection network", *IEEE Trans. on Computers,* vol. 38, no. 12, pp. 1703-1713, Dec. 1989.

[20] V. P. Kumar and S. J. Wang, "Reliability Enhancement by Time and Space Redundancy in Multistage Interconnection Networks", *IEEE Trans. on Reliability,* vol. 40, no. 4, Oct. 1991.

[21] V. P. Kumar and S. J. Wang, "Dynamic full access in fault tolerant multistage interconnection networks," *Proc. Int. Conf. Parallel Process.*, pp. I-621-I-630, 1990.

[22] V. P. Kumar, S. M. Reddy, "Augmented Shuffle-Exchange Multistage Networks," *Computer,* pp. 30-40, Jun. 1987.

[23] V. P. Kumar, J. Kneuer, D. Pal and B. Brunnr, "PHOENIX: A Building Block for Fault-tolerant Broadband Packet Switches," *Computer,* pp. 228-232, 1991.

[24] J. J. Li and C. M. Weng, "B-Tree: a High-Performance fault-Tolerant ATM switch," *IEE Proceedings Comm.*, vol. 141, no.1, Feb. 1994.

[25] S. Lin and F. Lin, "On Multiple-Fault Diagnosis of Baseline Interconnection Networks", *Computer,* 1992

[26] Martin de Prycker, "Asynchronous Transfer Mode Solution for Broadband ISDN," Second Edition, 1993.

[27] B. Menezes and U. Bakhrn, "New Bounds on Reliability of Augmented Shuffle-Exchange Networks," *IEEE Trans. on Computers,* vol. 44, no. 1, pp. 123-129, Jan. 1995.

[28] P. Mohapatra, C. Yu, and C. R. Das, "Allocation and Mapping Based Reliability Analysis of Multistage Interconnection Networks," *IEEE Trans. on Computers,* vol. 45, no. 5, May 1996.

[29] V. P. Nelson, "Fault-Tolerant Computing: Fundamental Concepts," *IEEE Computer,* vol. 23, no. 7, pp.19-25, July 1990.

[30] K. Padmanabhan, "An Efficient Architecture foe Fault-Tolerant ATM Switches", *IEEE ACM Trans. on Networking,* vol.3, no.5, Oct. 1995.

[31] K. Padmanabhan and D. H. lawrie, "A Class of Redundant Path Multistage Interconnection Networks", *IEEE Trans. on Computers,* vol. C-32, no. 12, pp. 1099-1108, Dec. 1983.

[32] C. S. Raghavendra and R. V. Boppana,"On Self-Routing in Benes and Shuffle-Exchange Networks," *IEEE Trans. on Computers,* vol. 40, no. 9, Sept. 1991.

[33] S. M. Reddy and V.P. Kumar, "On Fault-Tolerant Multistage Interconnection Networks,"*Proc. Of the Intl. Conf. Parallel Processing,* pp. 155-164, August 1984.

[34] R. Sahner and K. S. Trivedi, "Reliability Modeling using SHARPE," *IEEE Trans. Reliability,* pp.186-193, June 1987.

[35] D. Siewiorek and R. Shwarz, *The Theory and Practice of Reliable System Design*, Digital Press, U.S.A., 1982.

[36] P. U. Tagle and N. K. Sharma, "A High-Performance Fault-Tolerant Switching Network for B-ISDN", *IEEE Conf.*, pp. 599-606, 1995.

[37] N. Tzeng, P. Yew and C. Zhu, "A fault-tolerant scheme for multistage interconnection networks",*Proc. 12th Int'l Symp. Computer Architecture,* pp. 368-375, Jun. 1985.

[38] S. Urushidani, "Routing Network: A High Performance Self-Routing Switch for B-ISDN ", *IEEE JSAC,* vol. 9, no. 8, Oct. 1991.

[39] A. Varma and C. S. Raghavendra, "Reliability Analysis of Redundant-Path Interconnection Networks," *IEEE Transactions on Reliability,* vol. R-38, pp.130-137, April 1989.

[40] S. Yang and J. Silvester, "A Reconfigurable ATM Switch Fabric for Fault Tolerance and Traffic Balancing", *IEEE JSAC,* vol. 9,no. 8, Oct. 1991.

[41] S. Yang and J. Silvester, "Reconfigurable Fault Tolerant Networks for Fast Packet Switching", *IEEE Trans. on Reliability,* vol. 40, no. 4. Oct. 1991

Soft computing on small data sets

Bojan Novak
University of Maribor, Faculty of Electrical Engineering, Computer Science and Informatics,
Smetanova 17, 2000 Maribor, novakb@uni-mb.si

Keywords: soft computing, machine learning, neural networks

Received: October 24, 2000

The fusion of artificial neural networks (ANN) with soft computing enables to construct learning machines that are superior compared to classical ANN because knowledge can be extracted and explained in the form of simple rules. If the data sets are small it is hard to find the optimal structure of ANN because classical statistical laws do not apply. One possible remedy is the structural risk minimization method applied together with a VC dimension estimation technique. The construction of the optimal ANN structure is done in the higher dimensional space. The distortion of an image in this transformation can happen and the widely used expression for VC estimations based on minimal input data enclosing hypersphere and margin is not precise. An improvement of VC dimension estimation is presented. It enables better actual error estimation and is particularly suitable for the small data sets. Tests on some real life data sets have confirmed the theoretical expectations.

1 Introduction

One of the important steps in improving the usefulness of artificial neural networks (ANN) was fusion with soft computing [Zadeh 1994]. The construction of learning machines that are able to extract knowledge and explain it in the form of simple rules is now possible. The problem of an optimal number of neurons and weights of connections and their values is in a fact global optimization problem because of the nonlinear relations between the input and output data.

When data sets are small classical statistical laws do not apply and the construction of an optimal ANN is even harder. This problem is handled with the application of the structural risk estimation method together with the VC dimension estimation technique. Mapping of nonlinear relations between the input and the output is done by a transformation into the higher dimensional Hilbert kernel space where the problem is linearized. A distortion of image in this transformation can happen and widely used VC estimations based on minimal enclosing hypersphere and margin are not precise anymore.

In this paper a different approach that enables better VC estimation is presented. It is integrated into the structural risk minimization technique. An efficient strategy for constructing fuzzy artificial neural network (FANN) with the minimal actual error has been developed that can be easily implemented as a small addition to the existing FANN learning algorithm.
The performances of the proposed method were tested on some small data sets from the UC Irvine machine learning repository. The obtained results have confirmed theoretical expectations.

2 Support vector fuzzy modeling

In this chapter some basic definitions and modeling procedures are set. For given k observations, each consisting of a pair: x_i, y_i, where $x_i \in R^n$, $i=1,\ldots,k$ is the input vector and y_i is the associated output having values -1 or 1. Learning a machine is actually building up a mapping ability $x \rightarrow f(x,\alpha)$ where the functions $f(x,\alpha)$ themselves are labeled by adjustable parameters α. For the ANN α represents weights and biases. The expectation of test error for the trained machine is

$$R(\alpha) = \int \frac{1}{2}\left|y - f(\mathbf{x},\alpha)dP(\mathbf{x},\alpha)\right| \qquad (2.1)$$

where R(α) is the expected risk. The measured mean error rate on the finite number of observations is "empirical risk"

$$R_{emp}(\alpha) = \frac{1}{2k}\sum_{i=1}^{k}\left|y - f(\mathbf{x}_i,\alpha)\right| \qquad (2.2)$$

$R_{emp}(\alpha)$ is fixed for a particular choice of α and for a particular training set $\{x_i, \alpha\}$. The probability is not included in the equation. The quantity $\frac{1}{2}\,|\,y_i - f(x_i, \alpha)|$ is loss function. Empirical risk minimization does not imply a small error on a test set if the number of training data is limited. The structural risk minimization is one of new techniques for handling efficiently a limited amount of data. For a chosen η: $0 \le \eta \le 1$ the bound holds

$$R(\alpha) \le R_{emp}(\alpha) + \Phi(\frac{h}{k}, \frac{\log(\eta)}{k}) \quad (2.3)$$

where Φ is defined as :

$$\Phi(\frac{h}{k}, \frac{\log(\eta)}{k}) = \sqrt{\frac{h(\log\frac{2k}{h}+1) - \log(\frac{\eta}{4})}{k}} \quad (2.4)$$

The parameter h is the Vapnik Chervonenkis (VC) dimension [Vapnik 1998]. It describes the capacity of a set of functions implemented on the learning machine.

According to eq. (2.3), the risk could be controlled by two quantities: $R_{emp}(\alpha)$ and $h(f(x,\alpha): \alpha \in k_{sub})$, where k_{sub} is some subset of index set k. The empirical risk R_{emp} depends on the choice of the optimal function (α) applied in the learning machine. The VC dimension h depends on the set of functions $\{f(x,\alpha) : \alpha \in k_{sub}\}$. The parameter h is controlled by introducing the structure of nested subsets $S_n := \{f(x,\alpha) : \alpha \in k_n\}$

$$S_1 \subset S_2 \subset S_3 \subset .... \subset S_n \subset .... \quad (2.5)$$

with adequate VC dimensions satisfying

$$h_1 \le h_2 \le ..... \le h_n \le ......$$

The structural minimization principle chooses the function $f(x,\alpha^*)$ in the subset $\{f(x,\alpha): \alpha \in k_{sub}\}$ with the minimal right hand side of (2.3). The guaranteed risk bound is minimal. For a given set of data

$$(\mathbf{x}_1, y_1),.....,(\mathbf{x}_k, y_k), \ \mathbf{x} \in R^n, y \in \{-1, 1\} \quad (2.6)$$

a separation of two classes can be performed with an optimal hyperplane

$$(\mathbf{w}_0 \cdot \mathbf{x}) + b_0 = 0 \ . \quad (2.7)$$

The margin

$$\rho = \frac{1}{\|\mathbf{w}\|}$$

can be maximized by the following quadratic programming model

$$\min \Phi(\mathbf{x}, \xi) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C\sum_{i=1}^{k} \xi_i \quad (2.8)$$

subject to constraints

$$y_i(\mathbf{w}_0 \mathbf{x}_i + b_0) \ge 1 - \xi_i$$

and

$$\xi_i \ge 0$$

where ζ are slack variables introduced in the case when the problem is not separable, and C is the pre-specified value.

For the nonlinear cases a non-linear support vector approach is applied. A non-linear mapping is applied to map the data in a higher dimension feature space where a linear classification is applied. This is possible with the kernel functions. These functions originate from the theory of Reproducing Kernel Hilbert Spaces [Aronszajn 1950]. An inner product in the feature space has an equivalent kernel input space

$$K(\mathbf{x}, \mathbf{y}) = k(\mathbf{x}) \cdot k(\mathbf{y}) \ . \quad (2.9)$$

If K is a positive definite function, satisfying Mercer's conditions

$$K(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^{\infty} \alpha_m \psi(\mathbf{x})\psi(\mathbf{y}), \quad \alpha_m \ge 0 \quad (2.10)$$

$$\iint K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} > 0, \quad \int g^2(\mathbf{x}) d\mathbf{x} < \infty \quad (2.11)$$

then the kernel is a legitime product in the feature space.

There are different functions satisfying Mercer's condition: polynomial, splines, B-splines, radial basis functions, etc. In the present work the Gaussian radial basis function is used:

$$K(\mathbf{x}, \mathbf{y}) = \exp[-\frac{(\mathbf{x} - \mathbf{y})^2}{2\sigma^2}] \ . \quad (2.12)$$

The support vector technique places one local Gaussian function in each support vector. This means that there is no need for a clustering method. The basis width σ is selected using structural minimization principle (2.3) and (2.5). The non-linear classification support vector solution using kernels can be solved by

$$\min f(\mathbf{x}, \alpha) = sign(\sum_{i=1}^{k} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \quad (2.13)$$

subject to constraints

$$\alpha_i \ge 0 \ . \quad (2.14)$$

The coefficients α_i can be found by the following quadratic optimization problem

$$\max W(\alpha) = -\frac{1}{2}\sum_{i,j=1}^{k}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{y}_j) + \sum_{i=1}^{k}\alpha_i \tag{2.15}$$

subject to constraints

$$\sum_{i=1}^{k}\alpha_i y_i = 0$$
$$0 \le \alpha_i \le C, \quad i = 1,\dots,k \tag{2.16}$$

In the solution of (2.15) only some coefficients α_i differ from zero. The corresponding vectors are support vectors.

The model described by (2.15) has only one optimum (that is also global) which is a great advance against the backpropagation based learning algorithm in ANN.

The following support vector FANN architecture can be defined as presented in the fig. 2.1

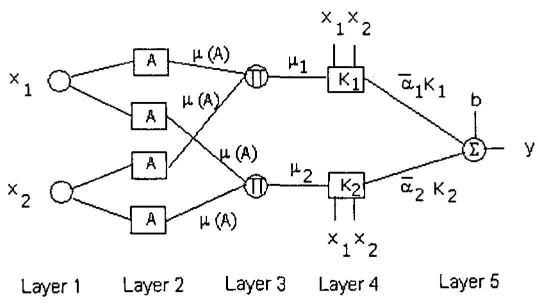


Figure 2.1 Support vector FANN architecture

Layer 1 calculates membership values. Layer 2 performs T norm operator (multiplication). Layer 3 derives the product of each rule's output. Layer 4 performs the kernel Gaussian radial basis operation (2.12). Layer 5 sums its inputs as the overall output where $\acute{\alpha}_i = \alpha_i$. The non-linear classification function is

$$f(x) = sign(\sum_{i=1}^{SV}\alpha_i K(x_i,x) + b) \tag{2.17}$$

where SV is the number of support vectors.

3 Optimization of parameters

Described are the conditions for optimizing the parameters of FANN, considering that some asymptotical laws from statistics are not valid for small data sets.

The empirical risk $R_{emp}(\alpha)$ in (2.2) is fixed for a particular choice of α and for a particular training set $\{x_i, \alpha\}$, and the probability is not included in the equation. The risk functional (2.1) depends on the conditional distribution function (c.d.f.) $P(x, \alpha)$, which is not known in advance. The only available information is from the finite independent and identically distributed (i.i.d.) training data sets. There are two possible approaches. The first one is to estimate the unknown c.d.f. $P(x, \alpha)$, or also unknown probability distribution function (p.d.f.) in the form of $p(x)$, and then compute the optimal estimate $f(x, \alpha_0)$. The second approach is to find a minimum of the empirical risk, which is calculated with the empirical risk minimization procedure (ERM). The second approach is preferable on the small data sets. It works if the asymptotic consistency is fulfilled

$$R(\alpha^*|\ k) \quad \rightarrow \quad R(\alpha 0) \quad when\ k \rightarrow \infty \tag{3.1}$$

$$R_{emp}(\alpha^*|\ k) \quad \rightarrow R(\alpha 0) \quad when\ k \rightarrow \infty, \tag{3.2}$$

where $R_{emp}(\alpha^*|\ k)$ is the optimal value that minimizes the empirical risk in the loss function (2.2). $R(\alpha^*|\ k)$ is the unknown value of the true risk for the same loss function. The ERM is consistent if the true risk $R(\alpha^*|k)$ and the empirical risk $R_{emp}(\alpha^*|\ k)$ converge to the same limit $R(\alpha_0) = min_\alpha\ R(\alpha)$ as the number of samples k grows toward the infinite value (fig. 3.1).

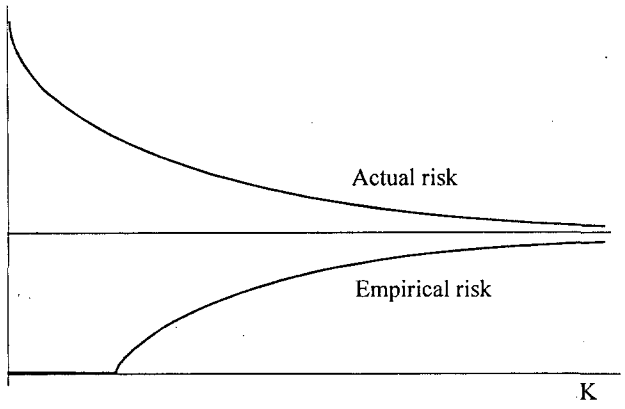


Fig. 3.1 Convergence of the empirical risk toward the actual risk

The condition that a small actual risk will be guaranteed at a small empirical error is given by the following limit [Vapnik et al. 1989]

$$\lim_{k\to\infty} P[\sup_{\alpha} | R(\alpha) - R_{emp}(\alpha) | > \varepsilon] = 0, \quad \forall \varepsilon > 0 \tag{3.3}$$

The consistency is determined for the worst case function.

The structural risk minimization principle is applied to find the optimal value for the true risk estimation

R(α|k)*. The penalization method is used to find the optimal function *f(x, α₀)* from the set of functions *f(x, α)*. Then the true risk *R(α)* is related to the empirical risk

$$R(\alpha) = R_{emp} r(p) \tag{3.4}$$

where *r(p)* is the penalization factor. In the support vector approach the penalization factor is of the form (2.4) and is proportional to the ratio of the VC dimension divided by the number of samples *k*.

The structural risk minimization principle (2.5) can be realized by estimating the VC dimension as a product of the radius of the minimal sphere that encloses data in the feature space divided by the margin (the distance between the hyperplane and the closest training vector in the feature space)

$$h = \frac{D^2}{\rho^2} \tag{3.5}$$

The actual risk prediction can be estimated very efficiently by the leave-one-out procedure. It consists of removing sequentially one sample from the training data and constructing *k* learning machines and testing all *k* elements for an error *L(xᵢ,yᵢ)*. The leave-one-out estimator is almost unbiased, that is

$$E\frac{L(\mathbf{x}_1 y_1, \dots, \mathbf{x}_k y_k)}{k} = ER(\alpha_{k-1}) \quad . \tag{3.6}$$

For optimal hyperspheres passing through the origin the equivalent of the expression (3.6) is

$$E\left[p_{error}^{k-1}\right] \le \frac{E\left[D^2 w^2\right]}{k} \tag{3.7}$$

where p^{k-1}_{error} is the probability of error on the test set. The expectation on the left is over all training sets of size *k-1*. The expectation on the right is over all training sets of size *k*.

This bound is tight when data fill almost the whole area of the sphere (fig 2.2 left) enclosing the training data. The consequence of data transformation into the feature space is that the sphere is often transformed into a flat ellipsoid (fig. 2.2 right). The bound (3.7) is not tight anymore.

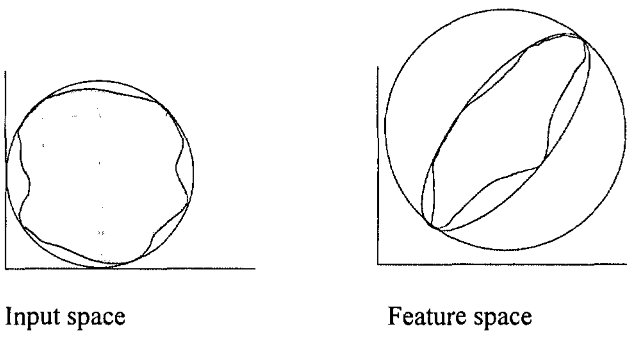


Fig. 3.2 the shape of the feature space

It is possible to achieve a better upper bound on the estimate of the expected error rate. The upper bound is constructed from the leave-one-out bound [Luntz 1969], Opper-Winther bound [Opper et al.], Khun Tucker optimality conditions [Karush 1939, Kuhn et al. 1951] and properties of the essential support vectors [Vapnik 1998].

The optimal supporting plane is unique, but can be expressed with different expansions of support vectors. Essential support vectors are those support vectors that appear in all possible expansion of an optimal hyperplane. Their number is presented by kesv and they have the following properties

$$k_{esv} \le n \tag{3.8}$$

where *n* is the dimensionality of the transformed input data in the feature space.

Let *ER(α)* is the expectation of the probability of an error for optimal hyperplanes constructed on the basis of training samples of size *k*, then the following inequality holds

$$ER(\alpha_{k-1}) \le \frac{EK_{esv(k)}}{k} . \tag{3.9}$$

In the case of a learning machine without threshold under assumption that the set of support vector does not change after removing example p (essential support vector), Opper-Winther equality applies

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \frac{\alpha_p^0}{(K_{SV}^{-1})_{pp}} , \tag{3.10}$$

K_{SV} is the kernel matrix (2.12). The f_0 is decision function (2.13) trained on the whole training set and f_p is decision function after one point x_p has been removed. It follows from (3.9) that the number of errors in the leave-out procedure is proportional to the number of essential support vectors. Instead of computing them we can use (3.10) and count the number of cases L_E when

$$y_p f^0(\mathbf{x}_p) \le \frac{\alpha_p^0}{(K_{SV}^{-1})_{pp}} \tag{3.11}$$

is true. Then expectation is

$$ER(\alpha_{k-1}) = \frac{L_E}{k} \tag{3.12}$$

4 Experimental results

The program developed on the described principle for constructing optimal fuzzy learning machine on small data set was tested on some real data sets from practice. The data are form the UC Irvine machine learning repository_(www.ics.uci.edu/~mlearn/MLRrepository). The first data set is a sonar data set (originally from www.boltz.cs.cmu.edu/ benchmarks/ sonar.html). The input consists of 104 samples of dimension 60, and the test data set is of the same size.

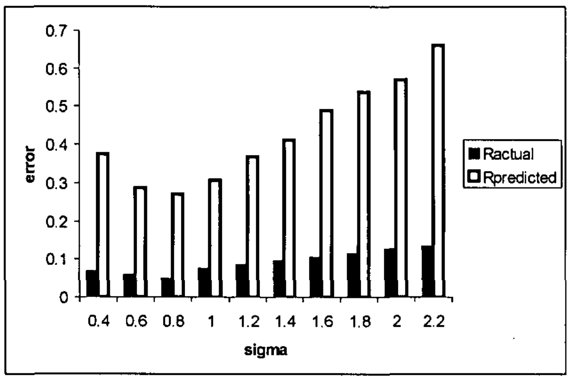


Fig. 4.1 The actual and the predicted risk for the sonar data

The actual error $R(\alpha)$ and its prediction (3.12) for different σ (sigma) values is presented in the fig 4.1. In our case a different set of functions $f(x,\alpha)$ is generated by varying σ in (2.12) for the radial basis function (α is actually σ in our case). The results are average values of a 100 fold repetition for each sigma value. Errors are given in the relative value (as in (2.2)).

Next example is the ionosphere data set with the input dimension n=33 and 200 learning samples and 151 test examples randomly generated from the complete set of 351 samples. The actual error $R(\alpha)$ and its prediction (3.12) for different σ (sigma) values is presented in the fig 4.2.

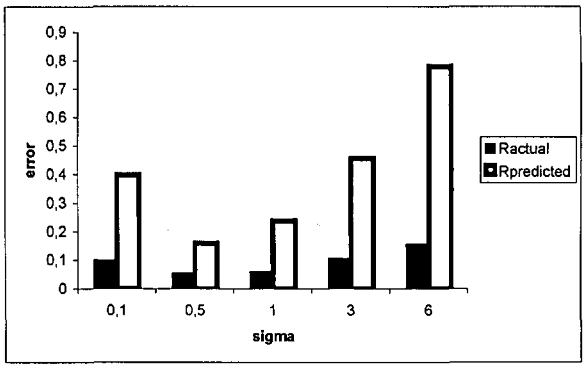


Fig. 4.2 The actual and the predicted risk for the ionosphere data

The last example is the Pima Indians diabetes data set with the input dimension n = 8 and 200 learning samples and 200 test examples randomly generated from the complete set of 768 samples.

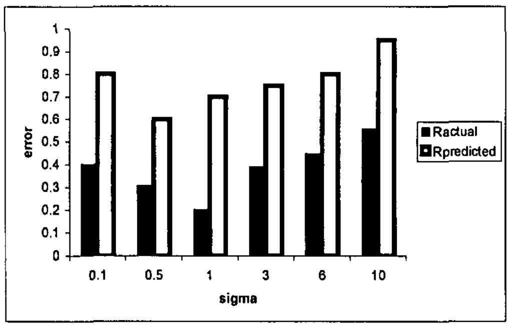


Fig. 4.3 The actual and the predicted risk for the Pima data

The actual error $R(\alpha)$ and its prediction (3.12) for different σ (sigma) values is presented in the fig 4.3.

5 Conclusion

In practice it often happens that the amount of data is limited. Such cases appear in engineering where data are collected through expensive experiments or in medicine where records on certain diseases are rare. When a data set is small a significant discrepancy between the empirical error achieved on the learning data set and the actual error on the testing data set appears. The actual error can be minimized with the structural risk minimization principle. It is calculated with the application of the VC dimension, which has to be estimated precisely to achieve good results.

The estimation based purely on the margin and the minimal diameter of sphere including input data can be inadequate due to possible flattening of the sphere caused by mapping data into the feature space.

In this paper a different approach that enables better VC estimation is presented. It is integrated into the structural risk minimization technique. An efficient strategy for constructing FANN with the minimal actual error has been developed that can be easily implemented as a small addition to the existing FANN learning algorithm.

The performances of the proposed method were tested on some small data sets from the UC Irvine machine learning repository. The obtained results have confirmed theoretical expectations.

6 References

[1] N. Aronszajn, Theory of Reproducing Kernels, Trans. Amer. Math. Soc. 68, 337-404 , (1950).

[2] W. Karush, Minima of functions of several variables with inequalities as side constraints. Master's thesis, Dep. Of Mathematics, Univ. of Chicago, 481-492, (1939).

[3] W. Kuhn. & A.W. Tucker, Nonlinear Programming in (J. NEYMANN ed.), Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, CA, 481-492, (1951),

[4] A. Luntz & V. Brailovsky, On estimation of characters obtained in the statistical procedure of recognition (in Russian), Techniceskaya Kibernetica, 3, (1969).

[5] M. Opper & O. Winther, Gaussian processes and SVM: Mean field and leave one out, in (A. J. SMOLA, P. L. BARTLETT, B. SCHÖLKOPF AND D. SCHUURMANS, editors), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 311-326, (2000).

[6] V. N. Vapnik, Statistical Learning Theory, John Wiley and Sons, (1998).

[7] V. Vapnik & J. Chervonenkis: The necessary and sufficient conditions for the consistency of the method of empirical risk minimization (in Russian), Yearbook of the Academy of Science of the USSR on Recognition, Classification, and Forecasting, 2, Moscow, 217-249, (1989). (English translation, The necessary and sufficient conditions for the consistency of the method of empirical risk minimization, Pattern Recog. Image Anal. 1, 284-305, (1991).

[8] L. A. Zadeh, Fuzzy logic, neural networks, and soft computing, Commun. ACM, 37/3, 77-84, (1994).

Computerized logistics information systems — a key to competitiveness

Anton Čižman
Univesity of Maribor, Faculty of Organizational Sciences
4000 Kranj, Kidričeva 55/a, Slovenia
E-mail: anton.cizman@fov.uni-mb.si

Keywords: information systems, logistics management, order processing, decision support systems, information tecnology, manufacturing, services, order processing

Received: November 17, 2000

Part of an organization's ability to use logistics as a competitive weapon is based on its ability to assess and adjust actual logistics performance real time. This means the ability to monitor customer demands and inventory levels as they occur, to act in timely manner to prevent stockouts, and communicate potential problems to customers. This requires excellent, integrated logistics systems which impact all of the logistics activities. In this paper we examined how computer and information technology can be used to support logistics management. Customer order cycle and order processing systems are pointed out first. Then advanced information technologies such as decision support systems, artificial intelligence, and expert systems which are being used directly to support decision making in logistics, are examined.

1 Introduction

Logistics activity is literally thousand of years old, dating back to the earliest forms of organized trade. As the area of study however, it first began to gain attention in early 1900s in the distribution and farms products, as a way to support organization's business strategy, and a away of providing time and place utility. Since the second World War, logistics began to receive increased recognition and emphasis.

The first dedicated logistics text began to appear in the early 1960s, which also is the time that Peter Drucker, a noted business expert, author, and consultant stated that logistics was one of the last real frontiers of opportunity for organizations wishing to improve their corporate efficiency. These factors combined to increase the interest of logistics.

Computer and information technology has been utilized to support logistics for many years. It grew rapidly with the introduction of microcomputers in the early 1980s. About this time, information technology really began to explode, which gave the organizations the ability to better monitor transaction intensive activities such as the ordering, movement, and storage of goods and materials. Combined with the availability of computerized quantitative models, this information increased the ability to manage flows and to optimize the inventory levels and movements. Transactional systems such as materials requirement planning (MRP, MRP II), distribution resource planning (DRP, DRP II), and just-in-time (JIT) allow organizations to link many materials management activities, from order processing to inventory management, ordering from a supplier, forecasting and production scheduling. Information technology is seen as the key factor that will affect the growth and development of logistics [13].
The order processing system is the nerve center of the logistics system. A customer order serves as the communications message that sets the logistics process in motion. The speed and quality of the information flows have a direct impact on the cost and efficiency of the entire operation. Slow and erratic communications can lead to lost customers or excessive transportation, inventory, and warehousing costs, as well as possible manufacturing inefficiencies caused by frequent production line changes. The order processing and information system forms the foundation for the logistics and corporate management information systems. It is an area that offers considerable potential for improving logistics performance.

Organizations of all types are utilizing computers to support logistics activities. This is especially true for companies thought to be on the "leading edge," that is, leaders in their industry. Such firms are heavy users of computers in order entry, order processing, finished goods inventory control, performance measurement, freight audit/payment, and warehousing. A recent study of world-class logistics practices cited logistics information systems as a key to competitiveness [11].

Going beyond "transaction processing and tracking," decision support systems (DSSs) are computer-based and support the executive decision-making process. The DSS is an integrative system of subsystems that has the purpose of providing information to aid a decision maker in making better choices than would otherwise be possible.

To support time-based competition, organizations are increasingly using information technologies as a source of competitive advantage. Systems such as quick response (QR), just-in-time (JIT), and efficient consumer response (ECR) are integrating a number of information-based technologies in an effort to reduce order cycle times, speed responsiveness, and lower supply chain inventory.

In addition, more sophisticated applications of information technology such as decision support systems, artificial intelligence, and expert systems are being used directly to support decision making in logistics.

Despite the accessibility of sophisticated information technology, the successful implementation of computerized logistics information systems into companies remains a complex and elusive issue. This situation is still more expressed in smaller manufacturing organizations, particularly those operating in Slovenia and in the other Central European countries facing intense transition processes for incorporation into the European Union and the global information society. This case is characterized by a lack of related methodological and management knowledge and skills [8, 15] in companies, which is the major weakness rather than limited funding for investments in advanced information technology.

Therefore the purpose of this contribution is to introduce the readers with the ways that computers and IT can be used to aid the logistics mangement which plays a key role in the economy. The paper begins with the customer order cycle which is the heart of logistics information systems. After that some significant order processing systems are pointed out and the use of typical logistics management informations systems to support decision making is examined.

2 Customer order cycle

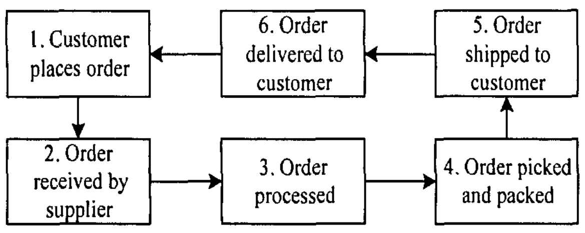
The *customer order cycle* includes all of the elapsed time from the customer's placement of the order to the receipt of the product in an acceptable condition and its placement in customer's inventory. The typical order cycle consists of the following components: (1) order preparation and transmission, (2) order receipt and order entry, (3) order processing, (4) warehousing picking and packing, (5) order transportation, and (6) customer delivery and unloading.

Figure 1 illustrates the flow associated with the order cycle. In this model taken from the customer's point of view, the total order cycle is 13 days [14]. However, many manufacturers make the mistake of measuring and controlling only the portion of the order cycle that is *internal* to their firm. That is, they monitor only the elapsed time from receipt of the customer order until it is shipped. The shortcomings of this approach are obvious.

In the example presented in Figure 1, the portion of the total order cycle that is internal to the manufacturer (steps 2, 3, and 4) amounts to only 7 of the 13 days. This ratio is usual for companies that *do not have* an *automated order entry* and *processing system*.

Improving the efficiency of the seven-day portion of the order cycle that is "controlled" by the manufacturer may be costly compared to eliminating a day from the six days not directly under the manufacturer's control. For example, it may be possible to reduce transit time by as much as one day by monitoring carrier performance and switching business to carriers with faster and more consistent transit times.

A change in the method of order placement and order entry may have the potential for the most significant reduction in order cycle time. An *advanced order processing system* could reduce the total order cycle by as much as two days. In addition, the improved information flows could enable management to execute the warehousing and transportation more efficiently, reducing the order cycle by another one or two days.


1. Customer places order
6. Order delivered to customer
5. Order shipped to customer
2. Order received by supplier
3. Order processed
4. Order picked and packed

Key:

1.	Order preparation and transmital	2 days
2.	Order received and entered into system	1 day
3.	Order processing	1 day
4.	Order picking/production and packing	5 days
5.	Transit time	3 days
6.	Customer receiving and placing into storage	1 day
	Total order cycle time	13 days

Figure 1: Total order cycle

3 Advanced order processing systems

The Order entry

A customer may place an order in many ways. Historically, customers handwrote orders and gave them to salespeople, mailed them to the supplier, or telephoned them to the manufacturer's order clerk, who then wrote it up. Today, it is more common for a customer to telephone orders to a supplier's customer service representative, who is equipped with a computer terminal networked to the supplier's database.

This type of system allows the customer service representative to determine if the ordered products are available in inventory, and to deduct orders automatically from inventory so that items are not promised to another customer. This improves customer service because if there is a stockout on the item, the representative can inform the customer of product availability and perhaps arrange product substitution while the customer is still on the telephone. In addition, this type of system almost completely eliminates the first two days of the order cycle described in Figure 1.

Electronic methods, such as an electronic terminal with information transmitted by telephone lines, and computer-to-computer hookups such as electronic data interchange (EDI), are commonplace today. These methods support the maximum speed and accuracy in order transmittal and order entry. Generally, rapid forms of order transmittal require an initial investment in equipment and software. However, management can use the time saved in order transmittal to reduce inventories and realize opportunities in transportation consolidation, offsetting the investment.

There is a direct trade-off between inventory carrying costs and communications costs. In many channels of distribution, significant potential exists for *using advanced order processing to improve logistics performance.*

The Order Processing

Once the order enters the order processing system, various checks are made to determine if (1) the desired product is available in inventory in the quantities ordered, (2) the customer's credit is satisfactory to accept the order, and (3) the product is scheduled for production if not currently in inventory. If these activities are *performed manually*, a great amount of time may be required, which can *slow down* (i.e., lengthen) *the order cycle*. The norm is that these activities are performed by computer in a minimal amount of time; often these activities can be performed simultaneously with other order cycle activities. The inventory file is then updated, product is back-ordered if necessary, and production is issued a report showing the inventory balance.

Management also can use the information on daily sales as an input to its sales forecasting package. *Order processing* next provides information to accounting for invoicing, acknowledgment of the order to send to the customer, picking and packing instructions to enable warehouse withdrawal of the product, and shipping documentation. When the product has been pulled from warehouse inventory and transportation has been scheduled, accounting is notified so that invoicing may proceed. All of these processes can be automated seamlessly to reduce additional input of data, and avoid the errors, paper shuffling, and nonvalue added of manual effort.

The primary function of the *order processing system* is to provide a communication network that links the customer and the manufacturer. In general, greater inconsistency is associated with slower methods of order transmission. Manual methods of order transmission require more handling by individuals; consequently, there is greater chance of a communication error. Management can evaluate methods of order transmission on the basis of speed, cost, consistency, and accuracy. As shown in Table 1, order transmission should be as direct as possible; orders transmitted electronically instead of manually minimize the risk of human error.

Table 1: Characteristics of Various Order Processing Systems

Level	Type of Systems	Speed	Cost to implement/ maintain	Consistency	Accuracy
1	Manual	Slow	Low	Poor	Low
2	Phone in to customer service rep with CRT	Intermediate	Intermediate	Good	Intermediate
3	Direct electronic linkage (EDI)	Rapid	Investment high; operating cost low	Excellent	High

The order processing system can communicate useful sales information to marketing (for market analysis and forecasting), to finance (for cash-flow planning), and to logistics or production. Finally, the order processing system provides information to those employees who assign orders to warehouses, clear customer credit, update inventory files, prepare warehouse picking instructions, and prepare shipping instructions and the associated documentation. In advanced systems, many of these activities are computerized.

No component of the logistics function has benefited more from electronic and computer technology than *order entry* and *processing*. Some advanced systems are so sophisticated that the orders are automatically generated when stock reaches the reorder point. Advanced order processing systems are shown as the second and third level in Table1.

Inside sales/telemarketing is an extension of the advanced order processing systems. It enables the firm to maintain contact with existing customers who are not large enough to justify frequent sales visits; increase contact with large, profitable customers; and efficiently explore new market opportunities.

Electronic data interchange (EDI) is the electronic, computer-to-computer transfer of standard business documents between organizations [14]. EDI transmissions allow a document to be directly processed and acted upon by the receiving organization. Depending on the sophistication of the system, there may be no human intervention at the receiving end. EDI specifically replaces more traditional transmission of documents, such as mail, telephone, and even fax, and may go well beyond simple replacement, providing a great deal of additional information.

Several types and variations of EDI systems are in use today. The main types of systems are proprietary systems, value-added networks (VANs), and industry associations.

Proprietary systems, also known as *one-to-many systems,* are aptly named, because they involve an EDI system which is owned, managed, and maintained by a single company. Value-added networks, also known as VANs, third-party networks, or *many-to-many systems,* appear to be the most popular choice for EDI systems. Under VANs, all of the EDI transmissions go through a third-party firm, which acts as a central clearinghouse.

Using EDI over the Internet is rapidly becoming a reality. After initial software purchase and systems setup, EDI over the Internet is virtually "free," versus VAN transmission. There is an Internet Engineering Task Force made of prominent companies such as Compaq, Hewlett-Packard, Digital Corporation, Microsoft, Oracle, SAS System, SAP/R3 etc. that is working to ensure the capability of EDI products on the Internet [10]. EDI has many potential benefits. The reduction of clerical work is a major benefit, reducing paper work, increasing accuracy and speed, and allowing purchasing to shift its attention to more strategic issues. One expert estimates that EDI can reduce the cost of processing a purchase order by 80 percent [3].

Of course, it will be necessary to justify an advanced order processing system in terms of cost-benefit analysis. The costs of developing the system, *start-up costs,* can be justified by comparing the present value of improvement in cash flows associated with the new system to the initial investment.

4 The company's logistics management information system

The order processing system sets many logistics activities in motion, such as:

- Determining the transportation mode, carrier, and loading sequence.
- Inventory assignment and preparation of picking and packing lists.
- Warehouse picking and packing.
- Updating the inventory file; subtracting actual products picked.
- Automatically printing replenishment lists.
- Preparing shipping documents (a bill of loading if using a common carrier).
- Shipping the product to the customer.

Other computerized order processing applications include maintaining inventory levels and preparing productivity reports, financial reports, and special management reports.

Processing an order requires the flow of information from one department to another, as well as the referencing or accessing of several files or databases, such as customer credit status, inventory availability, and transportation schedules. The information system may be fully automated or manual; most are somewhere in between.

Depending on the sophistication of the order processing system and the corporate management information system (MIS), the quality and speed of the information flow will vary, affecting the manufacturer's ability to provide fast and consistent order cycle times and to achieve transportation consolidations and the lowest possible inventory levels.

Generally, manual systems are very slow, inconsistent, and error prone. Information delays occur frequently. A manual system seriously restricts a company's ability to implement integrated logistics management, specifically, to reduce total costs while maintaining or improving customer service. Some common problems include the inability to detect pricing errors, access timely credit information, or determine inventory availability. Lost sales and higher costs combine to reduce the manufacturer's profitability.

Indeed, timely and accurate information is valuable. Information delays lengthen the order cycle. Automating and integrating the order process frees time and reduces the likelihood of information delays. Automation helps managers integrate the logistics system and allows them to reduce costs through reductions in inventory and freight rates. The communications network is clearly a key factor in achieving least total cost logistics.

Basic Need for Information

A logistics management information system is necessary to provide management with the ability to perform a variety of tasks, including the following:

- Penetrate new markets.
- Make changes in packaging design.
- Choose between common, contract, or private carriage.
- Increase or decrease inventories.
- Determine the profitability of customers.
- Establish profitable customer service levels,
- Choose between public and private warehousing.
- Determine the number of field warehouses and the extent to which the order processing system should be automated.

To make these *strategic decisions,* management must know how costs and revenues will change given the alternatives being considered.

Once management has made a decision, it must evaluate performance on a routine basis to determine (1) if the system is operating under control and at a level consistent with original profit expectations, and (2) if current operating costs justify an examination of alternative systems. This is referred to as *operational decision making.* The *order processing system* can be a *primary source of information* for both strategic and operational decision making.

An *advanced order processing system* is capable of providing a wealth of information to various departments within the organization. Terminals for data access can be made available to logistics, production, and sales/marketing. The system can provide a wide variety of reports on a regularly scheduled basis and status reports on request. It also can accommodate requests for a variety of data including customer order history, order status, and market and inventory position.

Designing the Information System

The design of a logistics management information system should begin with a survey of the needs of both customers, and a determination of standards of performance for meeting these needs. Next, customer needs must be matched with the current abilities of the firm, and current operations must be surveyed to identify areas that will require monitoring and improvement.

It is important at this stage to interview various levels of management. In this way, the organization can determine what strategic and operational decisions are made, and what information is needed for decision making and in what form. Table 2 illustrates the various types of strategic and operational decisions that management must make within each of the functions of logistics.

Table 2: Typical Strategic and Operational decisions by Logistics Function

Decision Type	**Customer Service**	**Transportation**	**Warehousing**	**Order Processing**	**Inventory**
Strategic	Setting customer service levels	Selecting transportation models	Determination of number of warehouses and locations	Extent of mechanization	Replenishment systems
		Freight consolidation programs	Public vs. private warehousing		
		Common carriers vs. private trucking	Public vs. private warehousing		
Operational	Service level measurements	Rate freight bills	Picking	Order tracking	Forecasting
		Freight bill auditing	Packing	Order validation	Inventory tracking
		Claims administration	Stores measurement	Credit checking	Carrying- cost measurement

	Vehicle scheduling	Warehouse stock transfer	Invoice reconciliation	Inventory turns
	Rate negotiation	Staffing	Performance measurement	
	Shipment Planning	Warehousing layout and design		
	Railcar Management	Selection of materials-handling equipment		
	Shipment routing and scheduling	Performance measurement		
	Carrier selection			
	Performance Measurement			

Table 2 (continuation)
Source: American Telephone and Telegraph company, Business Marketing, Market Management Division.

The next stage is to survey current data processing capabilities to determine what changes must be made. Finally, a common database must be created and management reports designed, considering the costs and benefits of each. A good system design must support the management uses previously described and must have the capability of moving information from locations where it is collected to the appropriate levels of management [2, 5].

Data for a logistics information system can come from many sources. The most significant sources of data for the common database are (1) the order processing system, (2) company records, (3) industry/external data, (4) management data, and (5) operating data. The type of information most commonly provided by each of these sources is shown in Figure 2.

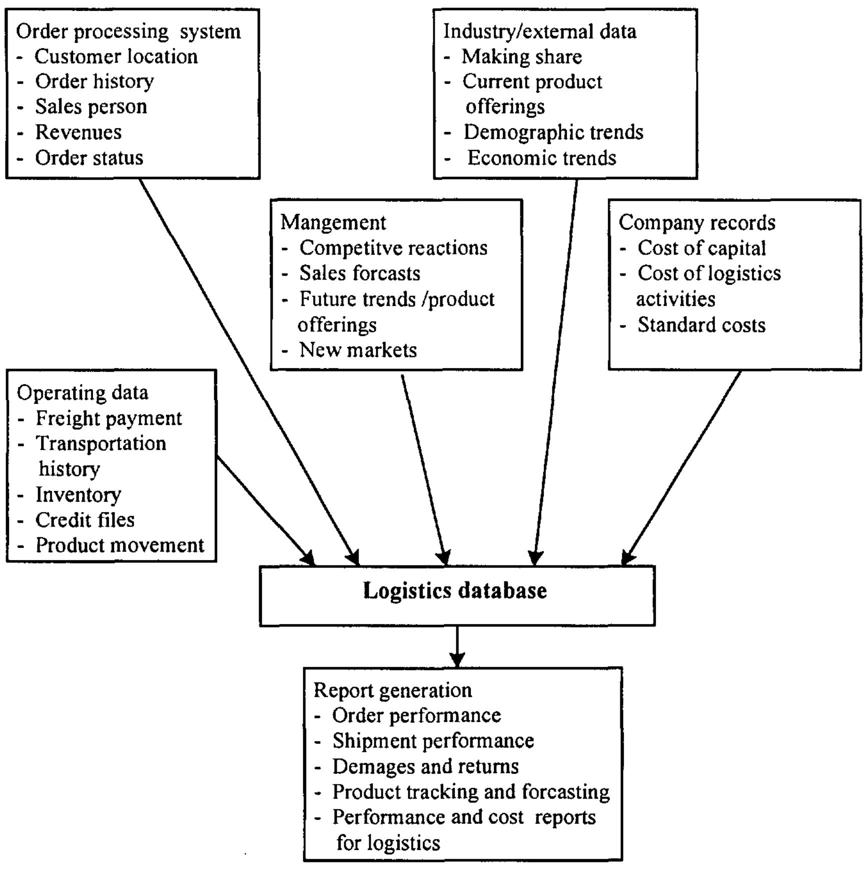


Figure 2: Key sources of information for the logistics database

Usually, the database contains computerized data files, such as the freight payment system, transportation history, inventory status, open orders, deleted orders, and standard costs for various logistics, marketing, and manufacturing activities. The *computerized information system* must be capable of (1) data retrieval, (2) data processing, (3) data analysis, and (4) report generation [1].

Data retrieval is simply the capability of recalling data such as freight rates (in their raw form) rapidly and conveniently. **Data processing** is the capability to transform the data to a more useful form (information) by relatively simple and straightforward conversion. Examples of data processing capability include preparation of warehousing picking instructions, preparation of bills of lading, and printing purchase orders.

Data analysis refers to taking the data from orders and providing management with information for strategic and operational decision making. A number of mathematical and
statistical models are available to aid a firm's management, including linear programming and simulation models. **Linear programming** [6, 7, 9] is probably the most widely used strategic and operational planning tool in logistics management. It is an optimization technique that subjects various possible solutions to constraints that are identified by management.

Simulation is a technique used to provide a model of a situation so that management can determine how the system is likely to change through the use of alternative strategies. The model is tested using known facts. Although simulation does not provide an optimal solution, the technique allows management to determine satisfactory solutions from a range of alternatives.

The last feature of an information system is **report generation**. Typical reports that can be generated from a logistics management information system include order performance reports; inventory management reports; shipment performance reports; damage reports; transportation administration reports; system configuration reports, which may contain the results of data analysis from mathematical and statistical models; and cost reports for logistics.

Management Information Systems (MIS), Decision Support Systems (DSS), and Executive Information Systems (EIS) represent a natural progression in information systems development. On-Line Analytical Processing (OLAP) is a recent advance in the field of Information Systems (IS) for decision support. OLAP not only integrates the MIS, DSS, EIS, functionality of the earlier generations of IS, but goes further and introduces spreadsheet-like multidimensional data views and graphical presentation capabilities [12].

4.1 Database Management

A database management system (DBMS) allows application programs to retrieve required data stored in the computer system. The types of data stored were shown in Figure 2. A DBMS must store data in some logical way, showing how different pieces of data are related, in order for retrieval to be efficient. This is a critical issue in logistics because of the large volume of data generated which may require analysis at a later date. For example, a buyer may want to see a history of transportation carriers with which it has placed orders for a particular item in the past six months.

The DBMS must be able to use the item number to reference the order and "pull up" the pertinent data. If the buyer sees that two suppliers have been used, the buyer may want the system to provide a transaction history with those suppliers over a given time period for all purchased items. The DBMS which must have the flexibility to sort data in a variety of ways that are meaningful to the user is the on-line transaction processing system (OLTP), also known as transaction database [12].

Relational database structures are popular today because they allow access to and sorting of data by relating the data to other data in many ways. This allows a great deal of flexibility. Increasingly, companies are using what is known as a local area network (LAN). This consists of a minicomputer linked to a number of microcomputers or terminals which allow access to a common database, software, and other systems features. LANs give microcomputers the power of mainframe systems.

Regardless of the sophistication of the software and hardware, a system cannot provide good results if the data in the system are not accurate and timely. Thus, systems integrity is vital. If people do not use the system consistently (i.e., do not scan each bar-coded item individually) the system will quickly be inaccurate. Once a system has data accuracy problems, it is very difficult, costly, and time consuming to correct.

4.2 Decision Support Systems

Decision support systems (DSSs) encompass a wide variety of models, simulations, and applications that are designed to ease and improve decision making [1, 5, 6]. These systems incorporate information from the organization's database into an analytical framework that represents relationships among data, simulates different operating environments (e.g., vehicle routing and scheduling), may incorporate uncertainty and "what-if" analysis, and uses algorithms or heuristics. DSSs actually present an analysis and, based upon the analysis, recommend a decision.

The artificial intelligence tools can be incorporated into DSSs, which may contain decision analysis frameworks, forecasting models, simulation models, and linear programming models. They can be used to assist in a wide variety of logistics decisions, such as evaluating alternative transportation options, determining warehouse location, and setting levels of inventory.

While the use of DSSs is not currently widespread, it appears to be growing as the potential contribution becomes more understood, and computing costs continue to decline. Figure 3 shows the components of a DSS.

A DSS is applications oriented. More specifically, a DSS has the following objectives:

- To assist logistics executives in their decision processes.
- To support, but not replace, managerial judgment.
- To improve the effectiveness of logistics decisions.

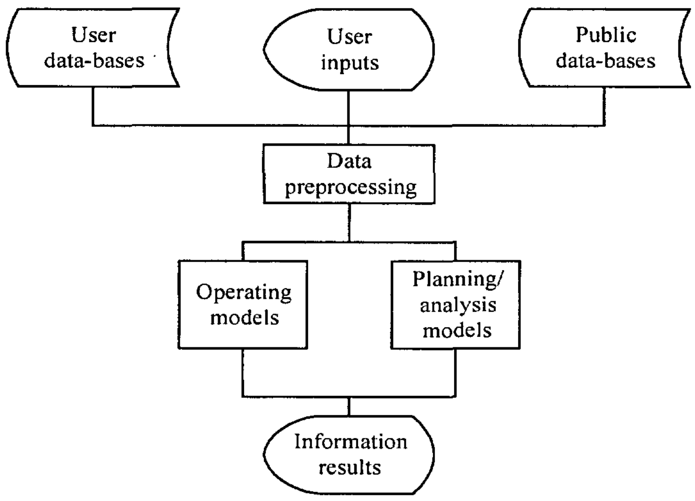


Figure 3: Decision Support System

Perhaps the most critical element of a DSS is the quality of the data used as input into the system. DSSs require information about the environment that is both internal and external to the organization. Thus, an important first step in DSS planning, implementation, and control is to have good external information.

Models are also needed to provide data analysis. Modeling can be defined as the process of developing a symbolic representation of a total system. A model must accurately represent the "real world" and be managerially useful. By using a model, we are able to establish a current situation and then play "what-if" games. This "what-if" ability is significant. It allows us to quickly consider many different alternatives and test to outcome [9, 14].

4.3 Artificial intelligence and expert systems

Developed out of the field of computer science, **artificial intelligence** (AI) is concerned with the concepts and methods of inference by a computer and the symbolic representation of the knowledge used in making inferences. The term *intelligence* covers many cognitive skills, including the ability to solve problems, to learn, to understand language, and in general, to behave in a way that would be considered intelligent if observed in a human.

AI is a comprehensive term encompassing a number of areas, including computer-aided instruction, voice synthesis and recognition, game-playing systems, natural language translators, robotics, and expert systems. While the number of AI applications is limited, the potential in logistics is staggering. AI has been used to model response time requirements for customer delivery; model transportation costs and times for various transportation modes, locations, and routings; determine which warehouses should serve which plants, with which products, and what inventory levels; model customer service response with various levels of reliability; and perform sensitivity analysis to determine how much inputs can vary without affecting the structure of the optimal solution [1].

Of specific interest to logistics executives are the subareas of AI known as *expert systems* (ES), *natural language recognition*, and *neural networks* [14]. An ES is defined as a computer program that uses knowledge and reasoning techniques to solve problems normally requiring the abilities of human experts. An expert system is an artificial intelligence (AI) program that achieves competence in performing a specialized task by reasoning with a body of knowledge about the task and the task domain.

Expert systems are capable of being applied to a variety of problems in marketing and logistics, including interpretation, monitoring, debugging, repair, instruction, and control. Examples of ES applications can be found in many industries.

Five criteria aid decision makers in determining whether expert systems should be used to solve a particular logistics problem. If any of the criteria are met, an ES may be appropriate:

1. The task or problem solution requires the use of human knowledge, judgment, and experience.
2. The task requires the use of heuristic (e.g.. rules of thumb) or decisions based on incomplete or uncertain information.
3. The task primarily requires symbolic reasoning instead of numerical computation.
4. The task is neither too easy (taking a human expert less than a few minutes) nor too difficult (requiring more than a few hours for an expert to perform).
5. Substantial variability exists in people's ability to perform the task. Novices gain competence with experience. Experts are better than novices at performing the task.

If an ES is appropriate, the next decision facing the logistics executive is whether the system can be economically justified and if EDI can be combined with other systems such as AI. Natural language capabilities of AI are using to make data stored within companies computer much more accessible. This contributes tremendously to the purchasing function as well as other areas of the firm.

Neural networks are still in the development stages. They can be considered an offshoot of ESs because they aid in decision making through the use of logic and rules. A key difference is that neural networks actually create their own rules based on past decisions and outcomes, rather than relying on an "expert." Once developed, these systems will be excellent for any repetitive activity that requires analysis of large amounts of data, more than a human could process effectively. As such, neural networks could be used to alert management to potential problems in supplier performance patterns, quality, delivery, invoicing, and similar issues.

5 Conclusions

This paper demonstrates how order processing system can directly influence the performance of the logistics function. Order processing systems can be used to improve customer communications and total order cycle time, or lead to substantial inventory reductions and transportation efficiencies.

Information is vital for the planning and control of logistics systems. Today's computer technology and communication systems make it possible for management to have the information required for strategic and operational planning of the logistics function. The order processing system can form the basis of a logistics information system and can significantly improve the quality and quantity of information for decision making.

Computers have become an invaluable aid to the logistics executive in making various operational and strategic decisions. Decision support systems, which are computer based, provide information for the decision-making process. The DSS has three components: data acquisition, data processing, and data presentation.

Computers are widely employed in many areas of logistics, including transportation, inventory control, warehousing, order processing, material handling, and so forth. Some of the most exciting areas of computerization are modeling, artificial intelligence (AI), and expert systems (ES). Improved database management contributes to the support of logistics decision making.

Future challenges for logistics performance improvement include the following significant areas: greater participation in setting organizational strategy and strategic planning process; total quality management (TQM); identification of opportunities for using logistics as a competitive weapon/marketing strength; just-in-time (JIT) logistics; the use of quick response (QR) and efficient consumer response (ECR) techniques, better understanding of global logistics and improved logistics information systems; greater participation of logistics professional work teams; appropriate understanding and use of outsourcing, partnership and strategic alliances; greater understanding and appropriate application of technology; green marketing.

It is our belief that this representation will make computerized logistics information systems more comprehensible and acceptable for implementation to a wider audience (managers), enhancing their decision-making effectiveness.

6 References

[1] Bender, Paul S., Using Expert Systems and Optimization Techniques to Design Logistics Strategies, Proceedings of the annual Conference of the Council of Logistics Management, Cincinnati, OH: Oct. 16-19, 1994.

[2] Bullers, I. William, Jr. and Reid A. Richard, Toward a Comprehensive Conceptual Framework for Computer Integrated Manufacturing, Information & Management 18 (1990) pp. 57- 67.

[3] Carbone James, “Make Way for EDI,” *Electronics Purchasing,* Sept. 1992, pp. 20—24.

[4] Cassidy Mike, “The Catalyst to Electronic Commerce,” *EDI World,* Apr. 1996, pp.14-16

[5] Caudhry, S. Sohail, Salhenberger, Linda and Beheshstain, Mehdi, 1996. A small business inventory DSS: design, development and implementation issues, Computres & Operations Research 23 (1) pp. 63-72.

[6] Čižman, Anton, Cerc, Samo, Pajenk, Andrej, 1999, Improving productivity using standard mathematical programming software, Proceedings SEUGI 99, Den Haag, Netherland, SAS Institute Inc.

[7] Čižman, Anton and Černetič, Janko: Improving Manufacturing Efficiency with Business Intelligence Software, Preprints of 6th IFAC Symposium on Automated Systems Based on Human Skill - Joint Design of Technology and Organization, Kranjska gora, Slovenia, 1997 (pp. 230-234).

[8] Dekleva, Saša, Zupancic Joze, Key issues in information systems management: a Delphi study in Slovenia, Information & Management 31 (1996) pp.1-11.

[9] Dilworth, B., James (1996): Operations management, McGraw-Hill.

[10] Drummond Rik, “EDI over the Internet Inter-operability,” EDI World", Apr. 1996, p. 8

[11] Global Logistics Research Team, Michigan State University, World Class Logistics: The Challenge of Managing Continuous Change, Oak Brook, IL: Council of Logistics Management, 1995, pp. 137—64.

[12] Koutsoukis, Nikitas-Spiros, Mitra, Gautam, Lucas, Cormac, 1999, Adapting on-line analytical processing for decision modelling: the interaction of information and decision technologies, Decision Support Systems 26 pp. 1–30.

[13] La Londe, J. Bernard and Masters, M. James, “The 1996 Ohio State University Survey of Career Patterns in Logistics,” Proceedings of the Annual Conference of the Council of Logistics Management, Oct. 20—23, 1996, pp. 115-38.

[14] Lambert, M., Douglas, Stock, R., James, Ellram, M., Lisa (1998): Fundamentals of Logistics Management, McGraw-Hill.

[15] Rupnik-Miklic, Erna, Zupancic, Joze, Experiences and expectations with CASE technology -- an example from Slovenia, 1995, Information & Management Vol. 28 (6) pp. 377-391.

[16] International Journal of Physical distribution and Logistics Management, special issue in "Supporting supply chain management through an IT/IS infrastructure", Part II, Vol. 30, No.7/8, (2000), pp. 640-660. Editors: Zahir Irani D.D.Love And Heng Li.

An agent that understands job descriptions

Andraž Bežek and Matjaž Gams
Jožef Stefan Institue, Jamova 39, 1000 Ljubljana, Slovenia
Phone: +386 61 1773 900, Fax: +386 61 1251 038
andraz.bezek@ijs.si, matjaz.gams@ijs.si

Keywords: intelligent agents, understanding, intelligent services

Received: July 6, 2000

Abstract: *An important property of intelligent agents is semi-understanding of desired tasks. We have designed an agent module referred to as Professional Description Assistant that is able to identify the corresponding job definition from a job description in textual form. Thus it plays a significant role in the advanced services of our EMA employment agent. By means of some standard and modified learning methods the findings proved to be quite encouraging. The new function aids the user performing some standard tasks, which normally demand human decision and experience, to a great extent.*

1 Introduction

"If software agents are such a promising technology, why can't they do something simple for me, like finding a movie? This comment is proffered as a sort of litmus test for agent capability."[10]

The above question refers to the task in which a certain amount of information is available, e.g. a movie description. The needed inference is the relation between a text description and one of possible choices.

We are basically dealing with a similar problem, only that it refers to the domain of employment. The aim of the task is accordingly to find the appropriate job definition from the description of a job. The problem, for example, might be as follows:

Job description: Construction worker
Job definition: auxiliary manual labour, constructing, heavy machinery.

The major problem we need to overcome is how to create and as well incorporate knowledge into an employment agent to enable such kind of understanding. These kinds of services are too difficult for classical systems to perform, and are rather new in the field of agents. These types of services are in general related to advanced Internet services, which apply at least some kind of knowledge or intelligence.

While the Internet on one hand is becoming more and more sophisticated, simple services on the other hand are only of rare occurence. They evolved and acquired the characteristics of intelligent systems and agents. With the latter we refer to software programs that, with some degree of autonomy, perform operations on behalf of the human user or another program. They help to automate a variety of activities, mostly the time-consuming ones. Software agents differ from "traditional" software in their ability to get personalized and social [3]. Agents are also semi-autonomous and can run continuously. We can distinguish four categories of agent functionality [1]:

- Problem-solving (typical research agents; intelligent agents, expert systems)
- User-centric (interaction with user; "intelligent" filtering, user guidance)
- Control (exclusive in multiagent systems, control services for other agents)
- Translation agents (bridge between systems with different data standards).

Here we shall limit ourselves to user-centric agents. They help users to achieve their goals in a more natural and usually in a more flexible way. Agents are in principle more intelligent, flexible and robust than the classical systems. Another important advantage of agents is their potential to struggle against the information overload[2]. Internet agents filter huge amounts of data, discover data relevant to user and present it in a structured way (e.g. a group with profession, location, salary etc.).

We have implemented a major national employment Internet system six years ago[8]. Recently we developed as well as implemented a couple of agent-related modules including the one presented here that is capable to find the appropriate job profession from a context-free job description.

2 Employment tasks

Online employment databases can be regarded as typical Internet services. They help users to accomplish job-related tasks. Instead of walking to the employment agency users simply surf WWW at home thus saving time, money and human resources. Currently, several hundreds of employment-related sites are available on the Internet [5]. They have attention-catching user interfaces combined with big databases of job seekers and job providers. However, current employment sites are limited to browsing and simple searching through a database. When a large amount of data is presented to user – this especially holds true for extensive employment databases – people find themselves lost in heap of information.
Employment databases have two major types of users: job seekers and job providers.
Job-seekers-related tasks:

- Predisposition: frequently provide job offers
- Browse/search through available jobs
- Enter job-searcher's data and match it with those of job providers
- Alert job searcher about new job offers (by means of an intelligent filter).

Job seekers are looking for an interesting job. As users they need to enter job descriptive words into the search query. The system offers the user corresponding, entry-related data in order that he may browse through. However, this kind of approach is appropriate only for small databases or special jobs. Many pieces of information in the system's reply render searching for the best job too time-consuming. Advanced search is therefore required. An agent can retrieve more descriptive data directly from the user, from his/her user profile, from the history of user actions. User profiles consist of job description, location of work, amount of salary, working conditions, working experience and schooling. With all this information an agent can provide more relevant data. Any additional facts of similar content also help to create information-rich output. Items can either be sorted by similarity score and/or grouped by several categories.

The tasks for job providers are similar to those of job seekers. Instead of searching for job providers the system searches for corresponding persons looking for a job. Thus also the problems regarding searching and the huge amount of available data are the same. The main quality for employment-related agents are therefore better understanding of users intention which help to extract relevant information from a heap of potential data.

The following employment-related sites can be found on WWW:

- **http://careers.altavista.com/**
  Provides simple searching with keywords combined with city- and state-limit criteria.
- **http://www.ajb.dni.us/**
  America's Job Bank contains more than one million available jobs. Job searching is limited with only one job title per search. It allows users to limit the distance of wanted job location by requesting zip code and radius.
- **http://www.hotjobs.com/**
  Hotjobs offers advanced search options. One can limit the search with a keyword, location, or the type of job. Results can be browsed, sorted out alphabetically by industry, location or company name.
- **http://www.occ.com/**
  Apart from the standard keyword search, location and job category selections, @Monster Jobs provides sub-search within the extracted data. One can as well limit the output by the date of job offer.
- **http://www.espan.com/esp/plsql/espan_enter.espan_home**
  Job Options uses typical job searching strategies and also provides the search for the following categories as employers, posting of résumés, notification of new jobs and a set of helpful career-dependant information (articles, surveys, links).

Slovenian job-related sites:

- **http://www.ef.uni-lj.si/jobprovider/**
  A job provider can search among available jobs and possible applicants. Job seekers must specify their profession and the area of activity. Simple and detailed view is available. Job providers can search for résumés and can give more search-narrowing conditions including schooling, the type and location of work as well as other skills.
- **http://www-ai.ijs.si/~ema/**
  EMA, an employment agent, is described in the next section.

3 EMA

EMA is an Employment Agent (http://www-ai.ijs.si/~ema/) developed by Intelligent Systems Department, Jozef Stefan Institute, Slovenia[8]. EMA's basic task is to help those who are looking for an employment or those providing it, and also to provide information on scholarships. It consist of several modules:

- Automatically updated database of available jobs
- Text matching search for available jobs
- Database of job seekers
- Automatic notification of job seekers for new jobs
- English and Slovenian speaking speech agent for browsing results.

The input

The input database of available jobs is automatically updated from the Web pages of the Employment Service of Slovenia (ESS, http://www.ess.gov.si/). EMA's communication agent daily transfers all employment data available on ESS's Web pages and incorporates it into EMA's database. This results in a daily updated database without user's interaction. The communicating agent can in principle parse other employment sites. The process demands adding a new Web address and parsing rules. In this way EMA has become a social agent since it can get data from any employment database on the Internet. Another major source of input data are Internet users who directly enter data.

3.1 Search

The search method matches text over the name of a job or the whole job description. In addition, users can limit their search by geographical areas of available jobs. Results are grouped by job names so that users can search the document for a wanted job.

3.1.1 Job seekers

EMA also handles job-seekers' database. Each job seeker must fill in a form with corresponding data (personal data, education, working experience, wanted job, expected salary). Job providers can search the job-seekers' database and browse it in a simple or more detailed way.

3.1.2 Notification

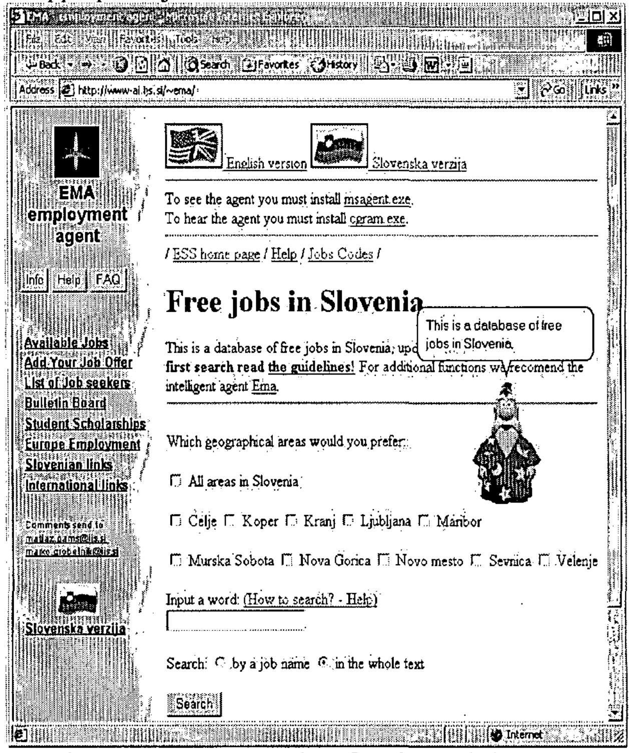
For registered users EMA can track the available employment information and notify users interested of any relevant new offers. When changes in any database occur, an agent, using interesting strings provided by the user, searches for appropriate jobs and mails them to registered users. In this way users do not have to visit employment web pages on daily basis; instead they are informed when relevant new information appears. In our opinion this feature significantly helps users to save time and computer resources.

3.1.3 Speech agents

EMA incorporates an English and Slovene speech agent. Results are "read" when a user clicks on a specific job on the screen. Microsoft Speech Agent was used for English language while the Slovenian one was entirely developed by the Intelligent Systems Department at Jozef Stefan Institute [6].

4 Profession description assistant

In order to improve EMA we have developed a profession-description auxiliary module. The idea is to help users searching for a job without having to know the exact definition of that job. In everyday life this is quite of frequent occurrence. When users register as job seekers they have to provide a lot of job-related data. Human employment agents find relevant information based on their experience. The whole process is called job matching since the systems tries to match user's data with an appropriate job.


Address http://www-ai.ijs.si/~ema/

EMA employment agent

English version Slovenska verzija

To see the agent you must install msagent.exe.
To hear the agent you must install cgram.exe.

/ ESS home page / Help / Jobs Codes /

Free jobs in Slovenia

This is a database of free jobs in Slovenia; upd... first search read the guidelines! For additional functions we recomend the intelligent agent Ema.

This is a database of free jobs in Slovenia.

Which geographical areas would you prefer:

☐ All areas in Slovenia

☐ Celje ☐ Koper ☐ Kranj ☐ Ljubljana ☐ Maribor

☐ Murska Sobota ☐ Nova Gorica ☐ Novo mesto ☐ Sevnica ☐ Velenje

Input a word: (How to search? - Help)

Search: ○ by a job name [✓ in the whole text]

Figure 1: EMA's agent interface in English.

For EMA the main difficulty is how to recognize which job is appropriate for a specific text description. Matching can be done in the following way. All available data are transferred into text for a specific description. This text is matched with a database of available jobs. Jobs that have the best matching scores are displayed to the user. A similar option is also to recognize job descriptions on the basis of previous examples. We applied this approach to develop a system, which is able to suggest appropriate jobs from a job descriptive text.

4.1 The algorithm

Text classification process can be presented in the following steps[7]:

A preprocessing step for determining the values of the features or attributes that will be used for representing the individual documents within a collection. This is essentially the dictionary creation process.

A *representation step for mapping each individual* document into a training sample using the above

dictionary and associating it with a label that identifies its category.
An induction step for finding patterns that distinguish categories from each other.
An evaluation step for choosing the best solution based on minimizing the classification error or cost.

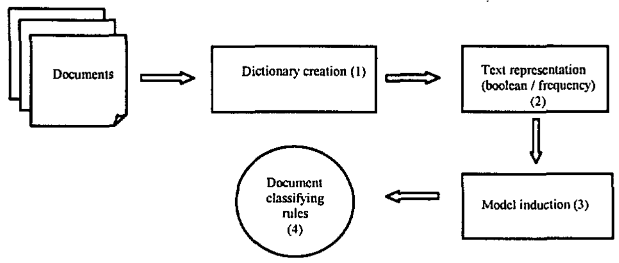

Documents
Dictionary creation (1)
Text representation (boolean / frequency) (2)
Model induction (3)
Document classifying rules (4)

Figure 2: A model for automatic classification based on previous examples.

In Figure 2, the test classification process is represented. Numbers in parentheses in Figure 2 correspond to the above four items. The initial task is to produce a list of attributes from samples of text of labeled documents, i.e. the dictionary. The attributes are single words or word phrases. Given an attribute list, sample cases can be described in terms of the words or phrases found in the documents. Each case consists of the values of the attributes for a single article, where the values could be either Boolean, i.e. indicating whether the attribute appears in the text or not, or numerical, i.e. frequency of occurrence in the text being processed. In addition, each case is labeled to indicate the classification or topic of the article it represents.

4.2 Learning dataset

We collected employment data for 23 weeks. The whole dataset has 15125 records categorized into 590 classes - job names. For further processing all records were cleared of irrelevant attributes. Only a job description and a job name remained in the record. As irrelevant attributes, the location of an employee and working conditions are regarded. Each word in a job description was considered an attribute. There were 5390 different attributes. An average record had 3.5 attributes; the maximum being 18 and the minimum 1. A class, on average, had 25,64 records, the smallest having one and the biggest 651 of them.

A dictionary of all words was created. Each word in the dictionary has its unique identification number. Each record was converted from the text into numeric presentation using word id-s combined with frequency statistics. This conversion was a necessary step in order to achieve reasonable short learning times.

An example of a learning dataset:

```
CAR MECHANIC
CAR MECHANIC - HEAD OF WORKROOM
CONSTRUCTION WORKER
ASSISTANT CONSTRUCTION WORKER
MACHINE MECHANIC
MACHINE LOCKSMITH
```

4.3 Classifying methods

We tested several classifying methods, namely:

- k-nearest neighbors (kNN)
- Naive Bayes method
- the method described here
- Linear combination of methods

kNN

kNN stands for k-nearest neighbor classification, a well-known statistical approach that has been intensively studied[11]. kNN has been applied to text categorization since the early stages of the research [12][17][15]. It is one of the top-performing methods on the benchmark Reuters corpus (the 21450 version, Apte set).

The kNN algorithm is quite simple: given a test document, the system finds k nearest neighbors among the training documents, and uses the categories of the k neighbors to weight the category candidates. The similarity score between test and all neighbor documents is used as a weight for the categories. If several of the k nearest neighbors share a category, the per-neighbor weights of that category are added together, and the resulting weighted sum is used as the likelihood score of that category with respect to the test document. With the process of sorting the scores of candidate categories we get a ranked list for the test document.

The algorithm is as follows:

```
∀ e ∈ E
  similar[e] := similar(e, $W_{test}$)
sort(similar[])
score[] := 0
∀ i ∈ 1..K
  c := class[example[i]]
  score[c] := score[c] + similar[i]
sort(score[])
```

Naive Bayes' method

This is one of the most successful algorithms for learning how to classify text documents[8]. Naive Bayes' classifier computes conditional probabilities of the classes given the instance and picks the class with the highest posterior probability. Attributes are assumed to be independent, an assumption that is unlikely to be true, but the algorithm is nonetheless very robust to violations of this assumption. Naive Bayes' formula is

$$P(c \mid V) = P(c) * \prod_{i=1}^{|V|} \frac{P(v_i \mid c)}{P(v_i)}$$

where P(c | V) represents the conditional probability of class c (in our case specific job) in vector of attributes (job description).
Because of

$$P(v_i \mid c) = P(v_i) * \frac{P(c \mid v_i)}{P(c)}$$

we can show that:

$$P(c \mid V) = P(c) * \prod_{i=1}^{|V|} \frac{P(c \mid v_i)}{P(c)}$$

The classifying algorithm therefore was as follows:

```
∀ c ∈ C
  score[c] := P(c)
  ∀ w  ∈ W_test
    score[c]:= score[c]*P(c|w)/P(c)
sort(score[])
```

For each class we compute P(c | V). A class with the biggest conditional probability is the most probable one. If we want to have a list of most probable classes, we can propose a couple of most probable classes. This is useful since users are often looking for several relevant jobs. Another reason is that the learning process is never 100 % accurate and the suggestion of only one post might be misleading. Therefore it is up to user to choose the corresponding job name from a couple of good candidates. In this way we give the final decision to the user. The agent is thus a true auxiliary – it only helps users to achieve their task by sensibly proposing a reasonable amount of relevant information.

Our method

We developed a statistical method based on conditional probability of attributes. The idea is that a conditional probability of attribute correctly evaluates the class given the attribute. P(w | C)=0, meaning that the word w does not belong to class C which is therefore not probable. A similar case holds true for P(w | C)=1. The attribute is contained only in class C which makes it the most probable. The conditional probabilities can be computed by using study examples.

Because of:

$$\forall w \notin c : P(w \mid c) = 0$$

we can take into account only classes which contain compared attributes.

$\forall w \in W_{test}$:

$$\sum_{c \in C} \sum_{w \in c} P(w \mid c) = 1$$

Therefore:

$$\sum_{w \in Wtest} \sum_{c \in C} \sum_{w \in c} P(w \mid c) = |W_{test}|$$

Because of the above rule we have to normalize class score by dividing it with the number of attributes in the test example. The result is probability distribution of classes.

$$P(c \mid V) = \frac{\sum_{i=1}^{|V|} P(v_i \mid c)}{|V|}$$

The algorithm being:

```
Score[] := 0
∀ w ∈ W_test
  ∀ c : (w ∈ c)
    Score[c] := Score[c]+P(w|c)
∀ c ∈ C
  Score[c] := Score[c]/|W_test|
sort(score[])
```

4.4 Linear combination of methods

The last tested method was linear combination of two classifying methods. The concept is based on the assumption that linear combination would have a superposition effect. Combination of two different methods can result in better classifying accuracy due to elimination of each method's bias.

score[c] := α*scoreX[c] + (1-α)*scoreY[c]

Our assumption can be shown with the following example:

method X score	method Y score	50%X + 50%Y
A 30%	D 35%	B 27,5%
B 25%	B 30%	A 25%
C 10 %	A 20%	D 17%
...	...	C 5%

Table 1: An example of the effect of linear combination.

By means of both methods class B scored high but not the best; with linear combination, however, it ranked the highest, which is correct.

Time complexity is a sum of time complexity for each method.

$$O_{\text{lin.combination}} = O_{\text{methodX}} + O_{\text{methodY}}$$

In this case we chose the Naive Bayes' and our own method. The application of both methods proved to be good, and consequently we made an assumption that two good methods can produce even better results.

$$\textbf{score[c]} := \alpha{*}\textbf{scoreNaiveBayes[c]} + (1-\alpha){*}\textbf{scoreOurMethod [c]}$$

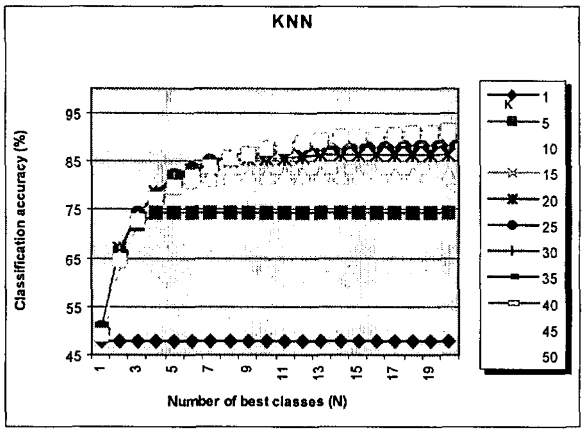
The parameter α adjusts the combination of methods. If α=0% only our method is applicable, whereas the value α=100% indicates that only Naive Bayes' method can be used.

4.5 Results

The ten fold cross-validation was used to test classifiers. Test examples represented 5% out of 15,125 in this case 756. These results show the average value of all tests. From that a conclusion can be drawn that accuracy for the single best class was 63.53%. This is obviously not accurate enough to be used for wider practice. However, when adding more possibilities one obtains significantly better results: 87.54% for 5, 92.54% for 10 and 95.05% for 20 best classes. Therefore, with a reasonable amount of suggested choices, the desired job definition was practically always included in the proposed list. This is presented in Figure 2.

Tests of the linear combination method are presented in Figure 3. In general, better accuracy is obtained in all cases. Although the accuracy gain might sometimes be small it cannot be regarded as harmful. To achieve significant improvements the tuning of the parameter α is necessary.

The results in Figures 2 and 3 are quite encouraging. One must note that study examples contain many errors such as noise, syntax errors, different abbreviations and inconsistent classifications. From a single text description even the best human experts shall give you several job definitions. All these properties of learning dataset influence the accuracy of classification. On this basis we can conclude that Profession Description Assistant has achieved the sufficient quality to be used in practice.



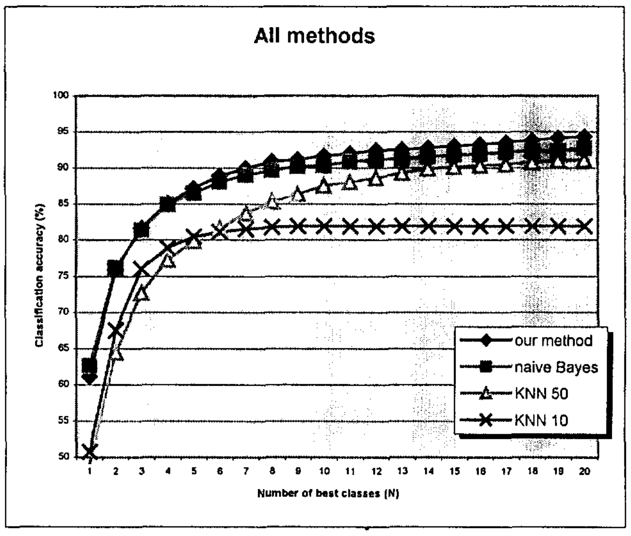


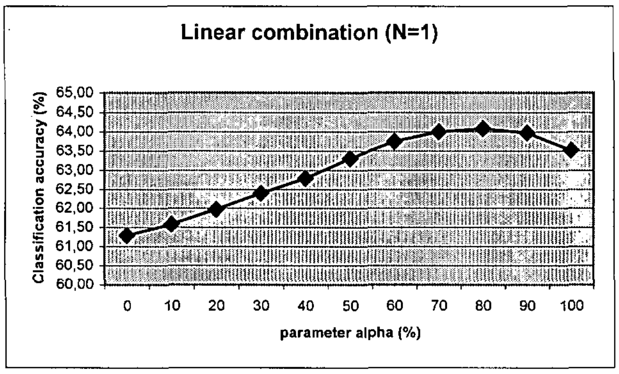
Figure 2: Testing methods on 15,125 records show that useful results are obtained with already 5-10 suggestions for the best job description.

5 Discussion

Profession Description Agent is a new agent module in the EMA employment agent. It shows that agents can help users with advanced functions based on domain-dependant knowledge. Introduction of new concepts enhances agent's functionality and usability. In this way the human-computer interaction is improved.

While modern systems are becoming more and more complex, thus overloading users with enormous amount of information, software designers must find ways to simplify interaction. Our implementation shows a possible solution to this problem.

Although intelligent agents are far from perfect they give additional functionality to existent systems. We think that future intelligent agents will incorporate the domain dependant knowledge. They will be able to execute advanced tasks and therefore be of greater importance to users.



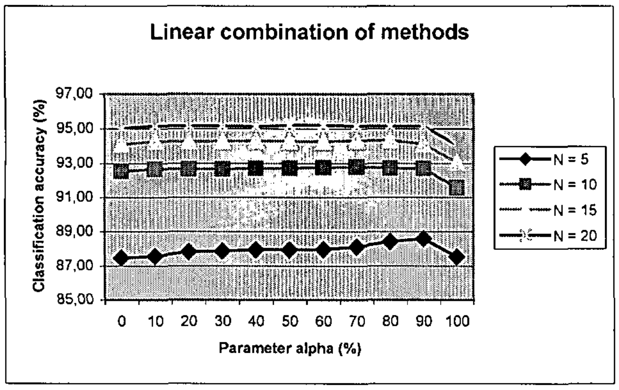


Figure 3: Linear combination practically always achieves better classification accuracy than any single method.

6 References

[1] James Hendler, Making Sense out of Agents, IEEE Intelligent Systems, pp. 32-37, March/April 1999.

[2] Matjaž Gams, Information Society and the Intelligent Systems Generation, Informatica 23, 4, pp. 449-454, 1999.

[3] Robert H. Gutmann, Alexandros G. Moukas, Pattie Maes, Agents as Mediators in Electronic Commerce, Electronic Markets, VOl. 8, No. 1, pp. 22-27, January 1998.

[4] Andraž Bežek, Automatic classification of job descriptions, Graduation thesis, University of Ljubljana, Slovenia, September 1999.

[5] List of job huntings sites, http://www-ai.ijs.si/~ema/EMA_Links-e.html

[6] Tomaž Šef, Aleš Dobnikar, Matjaž. Gams, Improvements in Slovene Text-to-Speech Synthesis, Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP '98), pp. 2027-2030, 1998, Sydney

[7] Chidanad Apté, Fred Damerau and Sholom M. Weiss, Towards language independent automated learning of text categorization models. Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval, pp. 23 – 30, 1994, Dublin, Ireland,

[8] Tom Mitchell, Machine Learning, McGraw Hill, 1997

[9] Matjaž Gams, Pavle Golob, Aram Karalič, Matija Drobnič, Marko Grobelnik, Jože Glazer, J., Pirher, Tone Furlan, Erik Vrenko, Radovan Križman, EMA - zaposlovalni agent, 1998, http://www-ai.ijs.si/~ema/EMA_Info-e.html

[10] Dana Moore, Ed Greengrass, Design Considerations for Agent Systems That Glean the Internet, IEEE Intelligent Systems, pp. 76-81, March/April 1999.

[11] Yiming Yang, Xin Liu, A re-examination of text categorization methods, Proceedings on the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 42 – 49, 1999

[12] Masand, B., Linoff, G. in Waltz, D. (1992). Classifying news stories using memory based reasoning. Proceedings of the Fifteenth Annual International ACM SIGIR conference on Research and development in information retrieval, pp. 59 – 65

[13] Apté, C., Damerau, F. Weiss in Sholom M. (1994) Towards language independent automated learning of text categorization models. Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval, pp. 23 – 30

[14] Cohen, William W. in Singer, Y. (1999) Context-sensitive learning methods for text categorization. ACM Trans. Inf. Syst. 17, 2, pp. 141 – 173

[15] Tokunaga, T. in Iwayama, M. (1995) Cluster-based text categorization a comparison of category search strategies. Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, strani 273 – 278

[16] J. van Rijsbergen, C. (1979) Information retrieval. Butterworths, http://sherlock.berkeley.edu/IS205/IR_CJVR/Preface.html

[17] Yang, Y. (1994) Expert network: effective and efficient learning from human decisions in text categorization and retrieval. Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval, pp. 13-22

A nested combinatorial-states model of sentence processing

Arthur C. Brett, Tadao Miyamoto and Joseph F. Kess
Department of Linguistics
University of Victoria
Victoria, British Columbia
Canada V8W 3P4

Keywords: natural language processing, psycholinguistics, finite automata

Received: March 23, 1999

Natural languages differ in the directionality of their syntactic build-up, developing from the phrasal head or matrix clause in either a left-branching direction or a right-branching direction. Japanese, for example, is quite consistently left-branching, while English is generally right-branching. We argue that the purported greater complexity of left-branching sentences arises from a confusion of analytical syntactic structure with sentence processing mechanisms, and we propose a model based upon the Chalmers' Combinatorial-State Automaton (CSA) which we suggest more nearly approximates actual neural processes. Our model, a Nested Combinatorial-States Automaton (NCA), can be constrained to require only finite memory resources, and assumes that the greater part of sentence processing occurs at the lexical level.

1 Introduction

It is well-known that natural languages differ in the directionality of their syntactic build-up, developing from the phrasal head or matrix clause in either a left-branching direction or a right-branching direction. For example, Japanese is quite consistently a left-branching language, while English is generally a right-branching language. >From a processing point of view, both languages face the psycholinguistic task of extracting information from the syntactic aspects of the input, whether it be the speech signal or the written word. But from a language-specific point of view, the two languages pose very different parsing settings for this psycholinguistic task of comprehension. For example, specific features of Japanese grammar, such as the relatively free word order and noun phrase deletion, interact with its left-branching syntactic build-up to pose the Japanese parser with many syntactic choice points at which there is potential ambiguity in respect to how what has occurred previously in the sentence relates to the next word or phrase. This is not as common in right-branching languages like English where one can often predict syntactic closure by constructing a likely parsing interpretation once a verb is encountered. Given this typological state of affairs, a realistic processing model must account for the psycholinguistic ease with which both types of syntactic structures are processed by the users of left- versus right-branching languages.

The focus of the research reported here is on the issue of space complexity, that is, on the extent and organisation of the transient memory and storage resources required to recognise sentences of different structures. Arguments for the greater complexity of left-branching languages, that is, that languages such as Japanese are more demanding to *recognise*, frequently cite the work of Yngve (1960) who demonstrated that the *production* of English-language sentences with predominantly left-branching constituent structures requires greater temporary memory than do sentences consisting entirely of right-branching constituents. We employ a method based on that of Yngve to establish that the *recognition* of left-branching structures actually entails less transient memory resource than do right-branching structures. This result invalidates claims based on Yngve's (1960) argument for the greater complexity of left-branching languages over right-branching languages. We attribute the invalid inferences drawn from Yngve's (1960) argument, and the discrepancy between our results and his, to a confusion of analytical syntactic structure with sentence processing mechanisms. Rather than abandoning premises regarding the constituent structure of sentences, however, we adapt these syntactic principles in devising the operating principles of our computational model of the sentence recognition process.

We base our model on a proposal introduced by Chalmers (1996a, 1996b) in his programme to develop the computational foundations of consciousness. Chief premise of the computational theory of mind is that the brain implements an automaton. To resolve issues related to the inadequacy of the *Finite-State Automaton* (FSA) as a model for the machine implemented by the brain, Chalmers introduced the concept of a *Combinatorial-State Automaton* (CSA), a machine the states of which have internal structure, represented as vectors of substates, and for which there are complex constraints on its transitions.

In developing our model, we adopt the thesis that the language processing components of the brain implement an automaton, a CSA, and that the states of this machine can be associated with the neural configurations which are realised as lexical items are processed. We posit further that the processing of lexical items is undertaken by a *Nested Finite Automaton* (NFA), and only on acceptance of an item, and values get instantiated on the substates of the CSA, is the CSA enabled to effect a transition. We identify the machine we propose as a *Nested Combinatorial-States Automaton* (NCA).

An NCA undertakes the greater part of its processing at the lexical level, and since it does not employ phrase-structure rules in the conventional fashion, the model realised in this machine can be described as "lexicalist" in the sense of Karttunen (1990) and Tugwell (1995). We demonstrate the equivalence of an NCA to a context-free phrase-structure grammar; however, in order that the capabilities of the automaton be consonant with those of the language processing apparatus of the brain, we introduce constraints on the extent of its temporary memory and storage resources. The resulting machine is a variety of finite device such as those proposed by Pereira & Wright (1991), Pulman (1986), and Tugwell (1995). The processing operation of this device is essentially "flat," consistent with our contention that the brain does not construct tree-like phrase markers in the course of its recognising sentences. Consequently, the acceptance of left- and right-branching structures does not entail disparate demands on the resources of either the automaton or the neural system.

In Section 2 we review the psycholinguistic issues involved in the processing of left-branching languages versus right-branching languages, and in Section 3 we outline the argument whereby we identify a contradiction inherent in Yngve's (1960) demonstration of the greater space complexity of left-branching sentences. We briefly describe Chalmers' (1996a, 1996b) CSA and present the details of our model in Section 4. In Section 5 we discuss the application of the model to the problem of the relative space and temporary memory demands of left- and right-branching syntactic structures. We also demonstrate the operation of an NCA as it accepts sentences with embedded relative clauses, and we discuss the constraints on its resources such that they be consistent with the capabilities of the language processing systems of the brain.

2 Psycholinguistic issues

Many of the world's 6000 or so languages are left-branching like Japanese and Korean, while somewhat fewer are right-branching like English, German, and Slovene. Speakers of both left- and right-branching language types experience little difficulty in processing the input string for information sufficient to build a mental model of what was said. Thus, one of the main problems faced by a universal processing model which claims to handle parsing preferences is the fundamental difference in ambiguity points in left- versus right-branching languages.

In terms of sentential expansion, indeterminacy is the rule rather than the exception in Japanese, given its salient

```
              S
             / \
           NP   VP
          /  \    \
         S    NP  kusatte-ita
        / \     \       'rotten was'
      NP   VP   tiizu wa
     /  \    \        'cheese'
    S    NP  tabeta
   /       \       'ate'
neko ga   nezumi ga
korosita     'rat'
 'cat
 killed'
```

Figure 1: Left-branching Japanese sentence.

```
         S
        / \
      NP   VP
     /    / \
  a cat  V    NP
        /    / \
  killed  NP     S
         /      / \
     a rat  Relp   VP
            /     / \
         that    V    NP
                /    / \
             ate  NP     S
                 /        \
             cheese    that was
                         rotten
```

Figure 2: Right-branching English sentence.

structural feature of left-branching directionality and the way this interacts with other important features of the language. First, let us consider the following Japanese sentences in respect to the increase in leftward directionality with the addition of relative clauses (see Kuno, 1973, p. 6–10). If we start with a basic SOV sentence such as *Neko ga nezumi o korosita* 'The cat killed a rat', each relative clause thereafter sends sentence development out in a leftward expansion.

1. *Neko ga nezumi o korosita* 'The cat killed a rat'
2. *Neko ga korosita nezumi ga chiizu o tabeta* 'The cat killed a rat that ate the cheese'
3. *Neko ga korosita nezumi ga tabeta chiizu wa kusatteita* 'The cat killed a rat that ate cheese that was rotten'

The left-branching structure of the final sentence in this sequence is show in Figure 1 (adapted from Kuno, 1973, p. 7). In contrast, its English translation yields the right-branching structure depicted in Figure 2 (also adapted from Kuno, 1973, p. 8).

The English sentences illustrate another source of indeterminacy in Japanese syntax, namely, the lack of overt markers to indicate the beginning of new clauses (see Kess and Nishimitsu, 1990). In Japanese, there is no commonly occurring equivalent of the English relativiser *THAT* which flags the subordinate clauses in the translations (2) and (3) above.

In addition to the lack of relativisers, the head-final syntax of Japanese is a further source of parsing ambiguities. The fact that the head always comes at the end of a sentence complicates the search for clause beginnings, and there is simply no grammatical element that overtly marks the beginning of an embedded clause. Though the case-marking particles present some subcategorisation information, this information is often insufficient to overcome the potential ambiguity. In addition, a relatively free word order in Japanese interacts with the frequent occurrence of empty Noun Phrases to multiply this indeterminacy. This essentially makes it impossible to know whether a single Noun Phrase marks the beginning of a clause or not. If there were no free word order, then it might be possible to regard any NP with *ga* as signalling a new clause; however, the situation is otherwise. Consider, for example, the following sentences:

Free word order:

John ga tegami o yonda 'John read the letter'

Tegami o John ga yonda 'John read the letter'

Empty NP:

[x] tegami o yonda '[x] read the letter'

John ga [x] yonda 'John read [x]')

Embedded clause with *ga*:

John ga [Mary ga syoonen ni kaita] tegami o yonda 'John read the letter that Mary wrote to the boy'

Free word order, embedded clause with *ga*:

John ga [syoonen ni Mary ga kaita] tegami o yonda 'John read the letter that Mary wrote to the boy'

Thus, in Japanese the parser, human or mechanical, is never really sure whether what follows is an embedded clause or not. If it is embedded, how deeply is it embedded? And it is never certain what kind of clause it is. What a processing model must contend with in explaining Japanese parsing is that Japanese allows a potentially large range of leftward embedding in its syntactic buildup. The processing path is unpredictable because the parser does not know where to place the beginning of a clause; and if the depth of recursion is potentially indefinite, the effectiveness of a look-ahead mechanism is theoretically strained in its efficacy.

But this is a problem only for a parser which is committed to a head-initial search for information. Head-final presentation of arguments is, as we have seen, just as common as head-initial presentation of arguments in the syntax of the world's language types. The thematic principle which seems to govern much discussion about processing models is that human language behavior can be ultimately explained not only by principles of a Universal Grammar of our underlying theoretical knowledge, but also by the principles of some as yet unspecified Universal Processing Grammar of our performance behavior. An implicit principle is that the abstract knowledge principles of the first will somehow be directly tied to, or at the very least reflected in, the performance behavior covered by the second. A third principle is that there is a logical priority to the Competence Grammar. That is, the expectation is that facts of performance will likely be anticipated, if not ultimately explained, by the dimensions of the competence grammar. This last assumption, however, is far from proven, and in fact, there is a growing interest in the opposite possibility that aspects of the competence grammar reflect features of the performance phenomena. For example, Hawkins (1990) convincingly argues that many cross-linguistic regularities follow from simple considerations of processing ease, and that performance is after all the driving force behind principles of linearisation in human grammars.

Most discussions of processing also typically assume that the human parser is both quick and efficient in what it does, further implying that reanalysis is costly (see Mazuka and Lust, 1988). The theoretical expectation is that the processor will use any information as soon as it becomes available, implying that keeping such information unused in on-line processing is inefficient. The conclusion is that parsing decisions must be made as quickly as possible. The concept of overlapping levels of redundancy and the fact of back-up systems is temporarily set aside in this essentially modular approach to understanding language processing. We may be over-emphasising the view that verbatim aspects of the short-term memory capacity are fundamental to the processing question. Ultimately it is the level of semantic representation that speaker and listener aspire to, the mutual creation of a mental model which is satisfactory to both for the conveyance of the appropriate information. And there is now increasing evidence from the psycholinguistic literature that meanings are continually negotiated, and that individual sentences are in reality embedded in the ongoing negotiation (see Clark and Wilkes-Gibbs, 1986; Schober and Clark, 1989; Wilkes-Gibbs and Clark, 1992).

Lastly, there has been a bias toward English-based theory-building, wherein branching tree diagrams are built top-downwards, based on words which are usually sequenced left-to-right in respect to syntactic relationships. This is a strategy which is relatively effective in a head-initial language like English where the head which determines the rest of the phrasal structure does appear first in that phrase. Material which comes later does not usually affect prior parsing decisions.

If all of the world's languages were right-branching, there would be nothing wrong with making this the fundamental assumption in a universal processing model. But such assumptions cannot be universally valid if they force wrong predictions on Japanese and other languages of the left-branching type (Mazuka and Lust, 1988). As we have demonstrated, if top-down, left-to-right parsing procedures are the norm, parsing for Japanese would require constant re-analysis. Very simply, the procedures which maximise efficiency in English, or are said to do so, have the opposite effect in Japanese. We have too often echoed claims attributed to Yngve's (1960) that left-branching gives rise to processing complexity, the argument being that left-branching structures are harder to process than right-branching structures because they require more nodes to be held in memory.

3 Yngve's model

Yngve (1960) limited his investigations to an analysis of the temporary memory required to produce English sentences with a variety of different syntactic structures. Although he did not deal with the issues of parsing, it is commonly assumed that Yngve's conclusions apply to the parsing process, and furthermore, that his results support claims for the greater processing complexity of left-branching languages, such as Japanese, relative to right-branching languages, such as English.

Because of its pervasive influence, and the misconceptions which have grown surrounding it, we briefly recapitulate Yngve's (1960) argument here. We then employ a variant of Yngve's argument to demonstrate that his results for sentence *production* cannot be used to support claims that the parsing or *recognition* of left-branching sentences is more demanding than recognition of right-branching sentences. This demonstration serves to motivate our introduction of the sentence processing model we present in the following section.

Register	Memory
S	·
NP	VP ·
S	NP VP ·
NP	VP NP VP ·
S	NP VP NP VP ·
NP	VP NP VP NP VP ·
neko ga	VP NP VP NP VP ·
VP	NP VP NP VP ·
korosita	NP VP NP VP ·
NP	VP NP VP ·
nezumi ga	VP NP VP ·
VP	NP VP ·
tabeta	NP VP ·
NP	VP ·
tiizu wa	VP ·
VP	·
kusatte-ita	·

Figure 3: Register and memory contents of an Yngve machine producing a "regressive" Japanese sentence.

Yngve assumed the sentences of a natural language to be *licensed* by a phrase-structure grammar, and he devised a "mechanism," an abstract computing machine, whereby the rules of the grammar might be applied to *produce* sentences of the language. The mechanism[1] consisted of input and output devices, a "register" to hold the one symbol upon which the machine operates, and two types of memory: permanent memory to contain the grammar rules, and a temporary memory, organised as a "last in – first out" store with finite capacity, for retention of intermediate results.

Figure 3. demonstrates the processing of an Yngve machine by showing the contents of the register and of the temporary memory as it produces the Japanese sentence with constituent structure depicted in Figure 1. The machine works *top-down*, traversing each branch of the structure as it performs *left-most expansions* of the grammar rules. Consequently, as it applies each rule while working down the tree, the machine must store the right-most right-hand side symbols of the rule, and then retrieve the stored symbols as it moves back up the tree. The memory contents reveal the evidence upon which Yngve based one of his principal conclusions, namely, that the *production* of sentences with such left-branching structures, which he described as "regressive," requires substantial temporary

[1]The Yngve (1960) machine can be identified as a variety of *Pushdown Automaton* (PDA), but with a finite store. Chomsky (1962) subsequently demonstrated that PDAs are acceptors of just the sentences generated by context-free phrase-structure grammars.

Register	Memory	Register	Memory
S	·	VP	·
NP	VP ·	V	NP ·
a cat	VP ·	*ate*	NP ·
VP	·	NP	·
V	NP ·	NP	S ·
killed	NP ·	*cheese*	S ·
NP	·	S	·
NP	S ·	Relp	VP ·
a rat	S ·	*that*	VP ·
S	·	VP	·
Relp	VP ·	*was rotten*	·
that	VP ·		

Figure 4: Register and memory contents of an Yngve machine producing a "progressive" English sentence.

Input	State	Store
neko ga	NP	·
korosita	VP	NP ·
·	S	·
nezumi ga	NP	S ·
·	NP	·
tabeta	VP	NP ·
·	S	·
tiizu wa	NP	S ·
·	NP	·
kusatte-ita	VP	NP ·
·	S	·

Figure 5: States and store contents of a PDA accepting a left-branching Japanese sentence.

memory.

Yngve observed that comparable temporary memory demands are inherent in the production of sentences with embedded constituents, which he also described as having regressive structures. Thus, in the case of both embedded and left-branching structures, the need to store intermediate results ultimately exhausts the machine's finite memory resources during protracted sequences of rule expansions.

Minimal temporary memory is required, however, during production of sentences with "progressive," right-branching structures, because each branch of the tree can be completed with few symbols being stored. Figure 4. demonstrates this effect, showing the register and temporary memory contents of an Yngve machine producing the English sentence with the structure illustrated in Figure 2. On the basis of results such as these, Yngve concluded that, although the grammar stored in permanent memory can in principle license infinitely many sentences, the finite temporary memory of his machine permits it to achieve equivalent productive capacity only in the case of right-branching, progressive structures.

While the effect of sentence structure on temporary memory is certainly an issue in language production, it is also a concern during recognition, the process of interest here. Consequently, we investigated the temporary memory requirements of a *Pushdown Automaton* (PDA)[2], a machine comparable to Yngve's mechanism, as it recognised left- and right-branching sentences. Figures 5. and 6. show the states and contents of the store of a PDA accepting the left-branching Japanese and right-branching English sentences with constituent structures illustrated in Figures 1. and 2., respectively. Comparison of the relative amounts of temporary memory required indicates that *recognition* of left-branching structures is actually less demanding of tem-

[2]Rather than running an Yngve machine "backward" to function as an acceptor, we choose to use the equivalent, but now more familiar PDA in this demonstration. Extent of the pushdown store can be limited to obtain a machine with finite memory such as that of the Yngve device.

Input	State	Store
a cat	NP	·
killed	V	NP ·
a rat	NP	V NP ·
that	Relp	NP V NP ·
ate	V	Relp NP V NP ·
cheese	NP	V Relp NP V NP ·
that	Relp	NP V Relp NP V NP ·
was rotten	VP	Relp NP V Relp NP V NP ·
·	S	NP V Relp NP V NP ·
·	NP	V Relp NP V NP ·
·	VP	Relp NP V NP ·
·	S	NP V NP ·
·	NP	V NP ·
·	VP	NP ·
·	S	·

Figure 6: States and store contents of a PDA accepting a right-branching English sentence.

porary memory than recognition of right-branching structures.

These results, when combined with those of Yngve, suggest that, while left-branching sentences (and languages) might be more demanding of temporary memory to produce, they are less demanding to recognise, and conversely, while right-branching sentences (and languages) might be less demanding to produce, they are more demanding of temporary memory to recognise. These conclusions conflict with the evidence that normal speakers of left- and right-branching languages produce and recognise the sentences of their languages with equal facility. Hence, we must doubt the validity of the processing model and question claims based upon it regarding the relative complexity of sentences (and languages) with different structures.

Yngve did acknowledge that the nature of the temporary memory of his device may have no valid basis in the structure of the human brain; however, in support of the last in – first out functioning of the machine's memory, he claimed that this organisation was determined by the form of the constituent-structure rules according to which the grammar was represented. Since the constituent-structure of natural language is supported by empirical evidence (see Garrett, Bever & Fodor 1966; Fodor, Bever & Garrett 1974; Johnson 1965, 1966), the source of the conflicting results presented above would seem to reside with the rule-based representation of the grammar and the implementations this representation determines.

While rule-based constituent-structure grammars may well serve the purposes of syntactic analysis, the foregoing demonstration suggests that they likely do not provide an accurate reflection of the sentence processing mechanisms employed by the brain. Hence, to obtain a realistic assessment of the relative processing demands of sentences (and languages) with different structures, a representation of constituent-structure grammar must be devised which is amenable to implementation by a plausible neural mechanism. This representation and its implementation should then account for evidence that left- and right-branching sentences can be produced and recognised with essentially equal facility by the speakers of predominantly left- and right-branching languages, respectively, but that speakers of both classes of languages experience comparable difficulty in processing sentences with excessive constituent embedding.

4 The nested combinatorial-states automaton

A widely held belief in cognitive science is that the brain implements an abstract automaton and that identification of a suitable class of automata is sufficient to at least *describe*, and perhaps *explain* mental properties and processes. Chalmers (1996a, 1996b) contends that the conventional *Finite-State Automata* (FSAs), with their monadic, structureless states, are inadequate to capture the complex structure and interactions inherent in physical systems such as the brain. Furthermore, FSAs lack the constraints on their transitions necessary to restrict computations to the processes characteristic of mental activity. Consequently, Chalmers introduced the *Combinatorial-State Automaton* (CSA)[3], each state of which possesses a complex internal structure that can be represented as a vector, $< s_1, s_2, \ldots >$, of substates, s_j, each of which may assume finitely many different values. The inputs and outputs of the CSA also have a vector structure.

Chalmers posits that a physical system such as the brain implements a CSA if the states of the physical system can be specified as vectors of substates and there exists a mapping of these substates onto the substates of the CSA such that, for every transition of the CSA, there is a transition of the physical system the initial and final states of which, together with the input and output, map to the corresponding states, and to the input and output, of the CSA. Chalmers suggests further that implementation conditions can be derived for other computational systems, such as Turing machines for example, by translating them into the CSA formalism, and thereby imposing suitable constraints on their computations. He also argues that the implementation conditions entailed by the fine-grained internal structure and the complex constraints on the transitions of CSAs preclude the trivial computations to which FSAs are susceptible while enabling a CSA to provide a suitable model of cognitive processes.

We adopt the thesis that the language processing neural apparatus implements an automaton, and we propose that this machine can be regarded as consisting of two automata, one nested within the other. The "outer" machine is a Chalmers CSA, the states of which are composed of paths in the transition network of the "inner," nested machine which we treat as a finite-state device. The two machines together constitute a *Nested Combinatorial-States Automaton* (NCA).

The nested machine, the *Nested Finite Automaton* (NFA), reads the individual tokens of a putative sentence and performs lexical-level processing which results in either acceptance, or rejection, of each separate token. If a token be rejected, the sentence is not accepted; however, should the token be accepted, the recognition path in the transition network of the NFA becomes the state of the CSA to which that machine then moves.

The CSA itself undertakes no input, although acceptance of a token read by the NFA is required in order that the CSA execute a transition. In addition to reading, the NFA yields output, and it is this output, together with that produced during acceptance of other tokens in the putative sentence, which determines whether or not the CSA will undergo a

[3]Chalmers proposed the CSA in response to Putnam (1988, p. 120–125) who argued against the computational foundations of mind by showing that every open physical system implements every finite automaton. Chalmers contends that the complex structure and fine-graining of the CSA enable it to be "tied" sufficiently closely (constrained) to the states, inputs, outputs, and transitions of a physical system that the "trivial computations" demonstrated by Putnam are prohibited.

transition.

The states of the NFA correspond to the syntactic categories of a phrase-structure grammar[4], and we propose that these categories can be associated with configurations realised in the neural circuitry of the language processing regions as the brain identifies lexical items according to their syntactic properties. Depending upon the particular formalism of the grammar, individual syntactic categories can be complex structures, or labels connoting complex structures, or simply denotational symbols (see Gazdar, *et al.* 1985, p. 20); they nonetheless can be associated unambiguously with particular neural configurations, where the more complex structures potentially admit a more finely-grained specification of the correspondence between neural configurations and machine states.

We propose further that the transitions of the machine can be associated with neural pathways of the brain in such fashion that transitions executed by the NFA correspond to changes in the neural configurations which occur in the course of recognising a token. Hence, following Chalmers (1996a, 1996b), we contend that, in principle, an implementation relation can be devised which maps neural configurations into syntactic categories, and that this mapping associates changes in neural configuration with transitions of the automaton. The transitions of the NFA are governed by two principles which we conceive to be grounded in processes undertaken by the brain as it recognises the individual tokens of a sentence, these principles being *category affiliation* and *category inclusion.*

Category inclusion reflects processes undertaken by the neural system as it establishes the syntactic role a token might play within the sentence. Application of the category inclusion principle by the NFA is initiated as it reads the token while moving from its initial state to a state corresponding to the lexical category of the token. Then, in subsequent transitions, the machine visits a sequence of one or more states corresponding to categories each of which includes the category of the preceding state as a constituent in the sense of a constituent-structure grammar. For example, in reading the token *'ate,'* the machine undergoes a transition from its initial state to a state which may be labelled 'V' to identify a lexical verbal category. In the next transition, the machine moves to a state labelled 'VP,' a phrasal category containing the lexical category 'V' as a constituent. The machine subsequently moves to a state labelled 'S,' a category of which 'VP' is a constituent. Hence, following recognition of the category of a lexical item, realisation of the category inclusion principle consists of identifying the hierarchy of constituents containing the item.

Application of the category affiliation principle is effected concurrently with the inclusion principle and reflects neural processing associated with licensing the syntactic environment of a token. Neural configurations realised in processing the token include a valence for those preceding and succeeding configurations which are syntactically legitimate. The implementation relation maps this valence into outputs produced by the automaton. Therefore, as it executes transitions, the NFA yields output specifying those categories that may precede and follow a token in the sentence. For example, in accepting the token *'ate'*, as the machine moves between the states labelled 'V' and 'VP,' it yields output expressing affiliation for a following 'NP' category, an oblique complement of the verbal category. Then, in moving between states 'VP' and 'S,' the machine expresses affiliation for a preceding 'NP,' a nominative complement. Consequences of the affiliation principle in this case are that the verbal category token *'ate'* will be accepted only if it follows an 'NP'-labelled category, and subsequent tokens will be accepted only if they are included in an 'NP'-labelled category.

The category inclusion and affiliation processing principles have analogues in some contemporary natural language grammar formalisms. Affiliation can be compared with *subcategorisation*, a formal device whereby such information as the complements of head categories is specified (see Gazdar, *et al.* 1985, pp. 31–35; Pollard & Sag 1994, pp. 23–24, 33–34). In our model affiliation is also defined for non-head categories. Consequently, since it entails an order on categories, affiliation can be identified with a *linear precedence* relation, and the category inclusion principle can then be identified with the *dominance* relation of generalised context-free phrase-structure rewriting rules wherein the left-hand side category is said to dominate right-hand side categories (see Gazdar, *et al.* 1985, pp. 44–50).

Although a correspondence of machine transitions to context-free rewriting rules can be established (Section 4.2), our model does not include explicit application of rewriting rules; it is our contention that the neural system does not employ such rules in the conventional sense. We assume instead that the grammar is encoded in neural pathways the architecture of which is such that they function in the manner of the transitions of a finite automaton.

Furthermore, we conceive that, in the course of its processing sentences, the brain does not construct tree-like phrase markers such as those which are generally employed in constituent-structure analyses. Rather, we posit that, as it receives tokens, the brain simply "categorises" them according to a category inclusion principle and either accepts or rejects them on the basis of affiliation co-occurrence constraints. Thus, the sentence recognition process may be viewed as essentially "flat" or "linear:" all processing occurs at what can been described as the "lexical level," with the analogue of structural analysis, consisting of the application of category inclusion and affiliation principles, being performed immediately as each token is received.

Although the category affiliation principle applies during recognition of individual tokens by the NFA, its effect is to impose constraints on the transitions permissible to

[4]The model is not contingent upon any particular set of categories. For this presentation, we employ a small collection of generally recognisable, albeit not necessarily universally recognised categories; but, we are not wedded to a particular repertoire of constituents.

the CSA in which the NFA is nested. Hence, the consequent category co-occurrence constraints apply also to the combined machine, the NCA, and determine the transitions admissible to it during acceptance, or rejection, of the putative sentence which is made up of the individual tokens.

4.1 The nested finite automaton

We begin our formal specification of the NCA with a description of the NFA, which we define in terms of a conventional FSA, an ordered quintuple

$$A = < Q, T, M, q_0, H >$$

where Q is a finite nonempty set of *states*; T is a finite vocabulary of *input symbols*; M is a mapping, called the *transition function*, from $Q \times T$ to Q; $q_0 \in Q$ is the *initial state*; and, $H \subseteq Q$ is a set of *accepting states*. (See Révész 1983, p. 60.)

To define an NFA, we introduce the following extensions to the FSA:

- The elements of Q, with the exception of the initial state, q_0, are identified with the *syntactic category labels* of a natural language phrase-structure grammar[5], and two nonempty sets $Q_T, Q_C \subset Q$ are distinguished where the elements of Q_T correspond to the *lexical categories* of the tokens in T such that, for each $t \in T$, there exists a state $q_t \in Q_T$, and where elements of Q_C correspond to the *constituent* or *phrasal categories* of the grammar. The set $Q' = Q_T \cup Q_C$ is introduced[6] and $Q = Q' \cup \{q_0\}$.
- A further nonempty set $P' \subseteq Q'$ of *output symbols* is distinguished, and a special, *null output* symbol, denoted "$\circ$", is added to obtain the set $P = P' \cup \{\circ\}$ of output symbols.
- The transition mapping M is defined to consist of the following relations:

$$M_T : \{q_0\} \times T \rightarrow Q_T$$
$$M_{P_L} : Q' \rightarrow P' \times \{\circ\} \times Q_C$$
$$M_{P_R} : Q' \rightarrow \{\circ\} \times P' \times Q_C$$
$$M_{P_O} : Q' \rightarrow \{\circ\} \times \{\circ\} \times Q_C$$

such that $M = M_T \cup M_{P_L} \cup M_{P_R} \cup M_{P_O}$. Thus, the machine undergoes input transitions, represented by M_T, and output transitions, of which there are three varieties, represented by M_{P_L}, M_{P_R}, and M_{P_O}, and identified as *left output*, *right output*, and *null output* moves, respectively. During an input transition, the machine reads the i^{th} token $t_i \in T$ of a putative sentence while moving from the initial state, q_0, to a state $q_t^i \in Q_T$. Such moves correspond to arcs $< q_0, t_i, q_t^i >$ in paths in the transition network of the machine. Each subsequent transition is one of the three varieties of output move, M_{P_L}, M_{P_R}, and M_{P_O}, corresponding respectively to the transition network arcs $< q_j^i, p_u^{i-1}, \circ, q_k^i >$, $< q_j^i, \circ, p_u^{i+1}, q_k^i >$, and $< q_j^i, \circ, \circ, q_k^i >$, where $q_k^i \in Q_C, p_u^{i-1}, p_u^{i+1} \in P$, and either $q_j^i \in Q_T$, or $q_j^i \in Q_C$, depending upon whether or not the move immediately follows the initial input transition. The category inclusion principle also applies so that for any transition from q_j^i to q_k^i, the category labelled q_k^i includes the category labelled q_j^i as a constituent.
- The set of accepting states, $H \subseteq Q_C$.

With the foregoing interpretations ascribed to Q, P, M, and H, an NFA is defined as an ordered sextuple

$$\mathrm{A} = < Q, T, P, M, q_0, H >$$

with T being a finite lexicon, and q_0 the initial state of the machine.

Paths in the transition network of the NFA are constrained by conditions imposed on transitions by the category affiliation principle. The constraining conditions are contingent upon the position, i, of a token and the length, n, of the putative sentence as follows:

1. For $i = 1$, there are two cases as follows:
   (a) If $n > 1$, there exists a right output arc, and any arcs in the path between it and the input arc are null output arcs.
   (b) If $n = 1$, the input arc is followed by null output arcs, and there are no subsequent left or right output arcs.
2. For $1 < i < n$, there are two conditions:
   (a) There exists an arc $< q_j^i, p_u^{i-1}, \circ, q_k^i >$, which is the only left output arc in the path, and there exists an arc $< q_u^{i-1}, \circ, p_j^i, q_v^{i-1} >$, which is the first right output arc in the path for the token t_{i-1}, such that $p_u^{i-1} = q_u^{i-1}$, $p_j^i = q_j^i$, and $q_k^i = q_v^{i-1}$;[7] and,
   (b) There exists a right output arc in the path for token t_i.
3. For $i = n > 1$,
   (a) Condition 2(a) must be satisfied; and,
   (b) There are no right output arcs in the path.

[5]The syntactic categories can constitute complex structures and the states of the machine may reflect this structure, although, in this presentation, we treat the states as corresponding to simple labels or atomic symbols.

[6]It can be the case that $Q_T \cap Q_C = \emptyset$, with the elements of Q_T treated as qualitatively distinct from the members of Q_C. For present purposes we regard Q_C, Q_T as not necessarily disjoint.

[7]Where states possess complex structure, equality is replaced by unification to yield the following conditions: $p_u^{i-1} \sqcup q_u^{i-1}$, $p_j^i \sqcup q_j^i$, and $q_k^i \sqcup q_v^{i-1}$.

A token is accepted, and the transition network path traversed by the NFA is identified as an *accepting path*, just in case the appropriate conditions above are satisfied, and the final state of the path is in H.

4.2 The combinatorial-state automaton

We adapt Chalmers' (1996a, 1996b) proposal to define a CSA with neither input nor output as an ordered quadruple

$$\mathcal{A}_c =< \mathcal{Q}, \mathcal{M}, \mathrm{q}_0, \mathcal{H} >$$

where $\mathcal{Q}$ is a finite nonempty set of states each of which consists of a vector, $< a_1, a_2, \ldots >$, of *substates*, a_j. $\mathcal{M}$, the *transition function*, is a mapping from $\mathcal{Q}$ to $\mathcal{Q}$; $\mathrm{q}_0 \in \mathcal{Q}$ is the *initial state*; and, $\mathcal{H} \subseteq \mathcal{Q}$ is a set of *accepting states.*

We then define an NCA as an ordered pair

$$\mathcal{A} =< \mathcal{A}_c, \mathsf{A} >$$

where A is an NFA and $\mathcal{A}_c$ is CSA such that

- The elements of $\mathcal{Q}$ are just the accepting paths of A, and q_0.
- For $\mathrm{q}_i \in \mathcal{Q}$, with the machine processing a putative sentence of n tokens, and $0 \leq i < n$, $\mathrm{q}_{i+1} \in \mathcal{Q}$ and $< \mathrm{q}_i, \mathrm{q}_{i+1} > \in \mathcal{M}$ just in case A accepts token $t_{i+1} \in T$.
- For $\mathrm{q}_{n-1} \in \mathcal{Q}$, with the machine processing a putative sentence of n tokens, $< \mathrm{q}_{n-1}, \mathrm{q}_n > \in \mathcal{M}$, $\mathrm{q}_n \in \mathcal{H}$ just in case A accepts token $t_n \in T$.

4.3 The equivalent grammar

An NCA is equivalent to a context-free phrase-structure grammar in the sense that, given a context-free grammar G, it is possible to construct an NCA $\mathcal{A}$ such that $L(\mathcal{A}) = L(G)$, where $L(\mathcal{A})$ denotes the language accepted by $\mathcal{A}$ and $L(G)$ denotes the language generated by G.

A context-free grammar is an ordered quadruple

$$G =< V_T, V_N, S, R >$$

where V_T and V_N are vocabularies of *terminal* and *nonterminal* symbols, respectively, with $V_T \cap V_N = \emptyset$; $S \in V_N$ is the *root* symbol of G; and, R is a set of rewriting rules each element of which has the form $A \to \alpha$, where $A \in V_N$ and $\alpha \in (V_T \cup V_N)^*$, the closure of the union of the two vocabularies. (See Révész 1983, pp. 2–3.) We restrict members of R to the following forms:

1. $W \to w$, where $w \in V_T$ and $W \in V_L \subseteq V_N$, with V_L being a set of *preterminal* or *lexical category* symbols;
2. $X \to Y$, where $X, Y \in V_N$; and,
3. $X \to YZ$, where $X, Y, Z \in V_N$.

These restrictions yield a Chomsky normal form grammar (Chomsky 1959), but with the revision that we retain *chain rules*, $X \to Y$, and we distinguish the subset V_L of preterminals. We identify such a grammar as a *Revised Chomsky Normal Form* grammar, G_R. Given a context-free grammar G, we can devise a G_R such that $L(G) = L(G_R)$; and hence, the foregoing constraints on the rewriting rules of G_R do not limit generality of a demonstration[8] of the equivalence of context-free grammars and NCAs.

Since configurations of an NCA are determined by the nested finite automaton, we construct an NFA such that $T = V_T$, $Q_T = \{q_W | W \in V_L\}$, $Q_C = \{q_X | X \in V_N\}$, $Q' = \{q_X | X \in V_N\}$, and $Q = Q' \cup \{q_0\}$ with $H = \{q_X | X \in (V_N - V_L)\}$, and $S \in H$ identified with the final state of the last arc in the accepting path for the last token in sentences $\sigma \in L(G_R)$, while $P' = \{q_X | X \in V_N\}$ and $P = P' \cup \{\circ\}$. Elements of the transition relation of the NFA are specified as follows:

1. $< q_0, w, q_W > \in M_T \iff W \to w \in R$;
2. $< q_Y, \circ, \circ, q_X > \in M_{P_O} \iff X \to Y \in R$; and,
3. Both $< q_Y, \circ, q_Z, q_X > \in M_{P_R}$ and $< q_Z, q_Y, \circ, q_X > \in M_{P_L} \iff X \to YZ \in R$.

In the procedure to establish $L(G) = L(G_R)$, we employ a *bottom-up construction*; that is, starting with $\sigma \in V_T^+$, we reduce it to S, provided $\sigma \in L(G_R)$, by a derivation method which consists of applying the right-hand sides of the rewriting rules to the sentential forms and replacing the matched symbols with the left-hand sides of the rules. Further, we employ a "dotted rules" strategy following the practice implemented in algorithms based upon *well-formed substring tables* (Earley 1970, Kay 1980). Thus, during a derivation, a rule $X \to YZ$ is applied if Y matches a symbol in the sentential form, but Z does not, and the result is recorded as $X \to Y \cdot Z$. When a match is subsequently obtained for Z, the result $X \to YZ\cdot$ is recorded. This device enables us to accommodate the correspondence of grammar rules of the form $X \to YZ$ with the pair of machine transitions specified in 3., above. The correspondences in 1. and 2. are straightforward.

To show that $L(\mathcal{A}) \subseteq L(G_R)$, we assume that $\sigma \in L(\mathcal{A})$ so that there exists a sequence of transitions of $\mathcal{A}$ which leaves the machine in an accepting state, that is, a path of A in $\mathcal{H}$, the final state of which is q_S. That $\sigma \in L(G_R)$ can be demonstrated by starting with σ and performing a bottom-up construction which consists of matching the application of an appropriate rule from R with each of the transitions executed by $\mathcal{A}$. Matching of rules of the form $W \to w$ and $X \to Y$ with the corresponding transitions $< q_0, w, q_W >$ and $< q_Y, \circ, \circ, q_X >$, respectively, is direct. When the machine undergoes a transition of the form $< q_Y, \circ, q_Z, q_X >$, however, partial application of the corresponding rule $X \to YZ$ must be undertaken, yielding

[8]To simplify the demonstration, we assume that G is λ-free; that is, there are no rules of the form $X \to \lambda$ in R so that $L(G)$ does not include the null word, λ.

Input	State
neko ga	NP (o,VP) S
korosita	VP (NP,o) S (o,NP) NP
nezumi ga	NP (S,o) NP (o,VP) S
tabeta	VP (NP,o) S (o,NP) NP
tiizu wa	NP (S,o) NP (o,VP) S
kusatte-ita	VP (NP,o) S

Figure 7: States of an NCA accepting a left-branching Japanese sentence.

Input	State
a cat	NP (o,VP) S
killed	V (o,NP) VP (NP,o) S
a rat	NP (o,S) NP (V,o) VP
that	Relp (o,VP) S (NP,o) NP
ate	V (o,NP) VP (Relp,o) S
cheese	NP (o,S) NP (V,o) VP
that	Relp (o,VP) S (NP,o) NP
was rotten	VP (Relp,o) S

Figure 8: States of an NCA accepting a right-branching English sentence.

$X \rightarrow Y \cdot Z$. Since the definition of A requires that there be a complementary transition $< q_Z, q_Y, \circ, q_X >$, completion of the application of $X \rightarrow YZ$ is guaranteed, yielding $X \rightarrow YZ\cdot$. The foregoing procedure reduces σ to S, establishing that $\sigma \in L(G_R)$, and hence, $L(\mathcal{A}) \subseteq L(G_R)$.

To show that $L(G_R) \subseteq L(\mathcal{A})$, we follow essentially the same procedure, save that we assume $\sigma \in L(G_R)$, so that there there exists a bottom-up construction which reduces σ to S, and during the process of this derivation, we match transitions of $\mathcal{A}$ with rewriting rules of G_R as these are applied. The result of this procedure is the demonstration that $\sigma \in L(\mathcal{A})$, and hence, $L(G_R) \subseteq L(\mathcal{A})$. Thus, we have $L(G_R) = L(\mathcal{A})$.

5 Results and discussion

Figure 7. records the states visited by an NCA (Nested Combinatorial-States Automaton) in the course of its accepting a left-branching Japanese sentence, the structure of which is illustrated in Figure 1. Each state is the path[9] traversed by the NFA (Nested Finite Automaton) as it recognises a lexical item. In reading the item *'neko ga'*, the NFA moves from its initial state to a state labelled 'NP,' a transition corresponding to the arc $< q_0$, *neko ga*, NP>. During its next transition, the machine moves from the state 'NP' to one labelled 'S' corresponding to the arc <NP, o, VP, S>. In the figure, the output is shown enclosed in parentheses, as '(o, VP),' between the state labels. Since no affiliation is expressed for a category preceding *'neko ga'*, Condition 1(a) in Section 4.1 is met, and with S $\in H$, *'neko ga'* is accepted. The NCA is consequently enabled to undergo a transition to the resulting state from its initial state.

Output produced in accepting *'neko ga'* licenses a 'VP'-labelled category included in an 'S'-labelled category. The next token in the sentence, *'korosita'*, is identified as a 'VP' category item, and in the transition to the state labelled 'S,' corresponding to the arc <VP, NP, o, S>, affiliation is expressed for a preceding 'NP' category included in an 'S' category. Hence, the licensing constraint and the category inclusion condition expressed during recognition of *'neko ga'* are met, and Condition 2(a) in Section 4.1 is satisfied. In the final transition, to the state labelled 'NP' and corresponding to the arc <S, o, NP, NP>, the machine yields output licensing a subsequent 'NP' category included in an 'NP' category, and consequently, Condition 2(b) in Section 4.1 is satisfied. With NP $\in H$, *'korosita'* is accepted, and the NCA moves to the resulting state.

[9]Only the output arcs are recorded in the figure; however, the starting state of the first such arc is labelled with the lexical category of the token read by the NFA during its initial transition.

States and transitions of the NCA corresponding to the remaining tokens of the sentence can be analysed in a fashion similar to that demonstrated for the initial two, but with the additional observation that the path for the final item, *'kusatte-ita,'* consisting of the one arc <VP, NP, o, S>, includes no expression of affiliation for a subsequent category. Hence, Condition 3(b) is satisfied, with the consequence that the NCA arrives in an accepting state, and the sentence is accepted.

An analysis analogous to that undertaken with Figure 7. can be presented for Figure 8. which shows the states of an NCA as it accepts a right-branching English sentence, the structure of which is depicted in Figure 2. Comparison of the two figures reveals them to be alike in all substantive respects, with at most marginal differences emerging upon closer examination. There is one arc in those states associated with recognition of the initial and final tokens, and two arcs in states associated with recognising the intervening tokens, with the number of arcs being determined by the number of other lexical items which are immediately adjacent to each token in the sentence. In the processing of an individual token, the state of the NCA includes an output arc expressing affiliation for the preceding (or the following) category, thereby "linking" the token to the previous (or the next) one in the sentence. Further examination reveals that the right- and left-branching sentences differ only in that a right output arc precedes a left output arc as the machine recognises a right-branching structure, while the reverse is true as the machine accepts a left-branching structure, with a left output arc preceding a right output arc.

It is evident that the states of the NCA consist of the same number of arcs at comparable stages of the process of accepting both the left- and the right-branching sentences. Hence, if the space resources consumed by the machine in a recognition were measured in terms of the numbers of categories or arcs in the paths comprising its states, then it would appear on the basis of Figures 7. and 8. that the acceptance of left- and right-branching sentences requires equal amounts of storage.

The NCA requires storage wherein the nested machine, the NFA, records the arcs comprising its states. The sequence of categories in a state of the NCA is a subset in the range of the implementation mapping which associates the categories with a collection of neural configurations. We consider that these configurations are likely spacially coincident. Thus, while the automaton must consume storage space during a recognition, the language processing system experiences a collection of concurrent neural configurations all of which occupy essentially the same physical regions of the brain. To facilitate our presentation, we identify the need to maintain several contemporaneous configurations during the recognition of a lexical item as a space requirement of the neural system, and we measure its extent in terms of the number of arcs in the corresponding state of the automaton.

We distinguish between the space used by an NCA during a recognition, and its temporary memory capabilities. The NCA does not have an auxiliary memory device such as a push-down store, or other comparable structure such as that with which an Yngve mechanism is equipped. Whatever temporary memory the machine does possess must reside within the arcs comprising its states. These are instantiated as the NFA recognises the individual tokens of a sentence, and reflect ensembles of contemporaneous neural configurations. Since the temporary memory aspect of the neural system might be viewed as arising from the persistence of these configurations during the processing of several tokens, the memory of the NCA must be manifested as a persistence or retention of categories in the arcs of successive states which are instantiated during recognition of successive tokens in a sentence. The retained arcs include expressions of affiliation for constituents which are not adjacent in the sentence and thereby link constituents separated by intervening categories.

Examination of Figures 7. and 8. reveals that no arcs expressing affiliation for non-adjacent constituents are instantiated in the states of the NCA during its acceptance of either the left-branching Japanese sentence of the right-branching English sentence. Hence, we may infer that the machine requires no temporary memory in either case. It is clear, therefore, that the substantially greater demand on temporary memory, which many have inferred from Yngve's (1960) results should be required to process left-branching sentences, is not realised in the case of the NCA. Nor is significant memory entailed in the recognition right-branching sentences, as one might conclude from the results obtained with a push-down automaton which we presented in our analysis of Yngve's (1960) model.

The functioning of the temporary storage mechanisms of the NCA becomes evident upon examination of the behaviour of the machine during the process of its accept-

```
(a)       S
         / \
        NP   VP
      /  \     \
     NP    S
   /     / \
        NP   VP
      /        \
```

```
(b)       S
         / \
        NP   VP
      /  \     \
     NP    S
   /     / \
        NP   VP
      /  \     \
     NP    S
   /     / \
        NP   VP
      /        \
```

```
(c)         S
           / \
          NP   VP
        /  /     \
            NP   V
          / \     \
         S    NP
        / \    \
      NP    VP
     /    /  \
         PP   V
       /       \
```

Figure 9: Backbone structures of sentences with embedded relative clauses: (a) single clause in a subject NP; (b) two such clauses; and (c) single clause in an object NP.

ing sentences with embedded constituents. Comparisons involving different levels of embedding also furnish an instrument with which to measure relative extents of the temporary memory and storage entailed in processing such structures. This information can then be employed to identify and estimate the limitations which must be imposed upon the resources of the automaton in order that they be consonant with the capabilities of the language processing components of the brain. More importantly, information of this nature can be employed to illuminate the mechanisms and functioning of the pertinent neural systems.

In Figure 9. we illustrate the backbone structures of typical sentences with embedded relative clauses. Figure 9(a) shows the skeleton of a sentence with a single relative clause which is attached as a constituent of a subject NP, while (b) depicts a second level of embedding of this nature. Such constructs are commonly encountered in English. Figure 9(c) shows a relative clause modifying an object NP, a structure characteristic of SOV languages such as Japanese.

The sentence *'a cat the dog chased ran'* is one of the simplest with a structure such as that depicted in Figure 9(a). In this instance, the embedded relative clause, *'the dog chased,'* is included in the subject NP and modifies the subject, *'a cat,'* of the matrix clause *'a cat ran.'* The states of an NCA accepting this sentence are shown in Figure 10.

In recognising the initial item, *'a cat,'* the nested component of the NCA executes transitions corresponding to the two arcs, <NP, ∘, S, NP> and <NP, ∘, VP, S>, whereby

Input	State
a cat	*NP* (∘,S) NP (∘, **VP**) S
the dog	NP (∘,VP) S (*NP*,∘) NP (∘, **VP**) S
chased	VP (NP,∘) S (*NP*,∘) NP (∘, **VP**) S
ran	**VP** (NP,∘) S

Figure 10: States of an NCA accepting an English sentence with an embedded relative clause.

two subsequent categories, labelled 'S' and 'VP,' respectively, are licensed, with the VP (shown in sans-serif bold in the figure) being the predicate phrase, *'ran,'* of the matrix clause, and S being the embedded relative clause, *'the dog chased.'* During its recognition of the second item, *'the dog,'* the machine undergoes transitions corresponding to three arcs. The first of these, <NP, ∘, VP, S>, licenses a following 'VP'-labelled category, the predicate phrase, *'chased,'* of the embedded clause. The second arc, <S, NP, ∘, NP>, includes an affiliation for a preceding 'NP'-labelled category (identified with italics in the figure), the subject phrase, *'a cat,'* of the matrix clause, and consequently, satisfies the licensing constraint expressed when that item was recognised. The third and last arc in the path, <NP, ∘, VP, S>, expresses affiliation for a subsequent 'VP'-labelled category (sans-serif bold in the figure), the predicate phrase of the matrix clause, consisting of the item *'ran.'* It is in this arc, which carries the licensing of the matrix predicate phrase over the embedded relative clause, that one aspect of the temporary memory resource of the NCA is manifested.

During its recognition of the next token, *'chased,'* the machine undertakes moves represented by three arcs, the first of which, <VP, NP, ∘, S>, includes the expression of affiliation for a preceding 'NP'-labelled category, the subject phrase, *'the dog,'* of the relative clause. The second transition, represented in the arc <S, NP, ∘, NP>, expresses affiliation for another preceding 'NP'-labelled category (italics in the figure), the subject phrase of the matrix clause. The linking effected by this affiliation, however, has already been accomplished, and hence, the arc serves no purpose in the recognition beyond preserving the category inclusion hierarchy of the token. It is this function which nonetheless reveals a second element in the temporary memory capability of an NCA: maintenance of structural relationships such as, in this instance, the embedded nature of the relative clause in the constituent structure of the sentence.

The third and final transition in the recognition of the token *'chased'* corresponds to the arc <NP, ∘, VP, S>, which has been identified already as storing the licensing constraint which is necessary to link the subject and predicate phrases of the matrix clause during processing of the embedded clause. This link is completed when the final token of the sentence, *'ran,'* is recognised. The machine executes a transition corresponding to the arc <VP, NP, ∘, S>, expressing affiliation for a preceding 'NP'-labelled category, the subject phrase of the matrix clause. The licensing constraint expressed in the "stored" arc <NP, ∘, VP, S> is thereby satisfied.

In processing this sentence, the automaton initially recognises the subject of the matrix clause, *'a cat,'* without "foreknowledge" that there exits a modifying relative clause. Hence, it instantiates the one arc, <NP, ∘, VP, S>, licensing an immediately following verb phrase. On encountering another noun phrase, however, rather than the "anticipated" verb phrase, the machine backs up and reanalyses the initial noun phrase. It instantiates an arc, <NP, ∘, S, NP>, licensing the relative clause, and effectively inserts it before the arc <NP, ∘, VP, S>, which then must be retained because it licenses the verb phrase that must follow the intervening embedded clause. When the machine returns to process the second noun phrase, *'the dog,'* subject of the relative clause, it instantiates the three arcs shown in Figure 10. The second of these, <S, NP, ∘, NP>, completes the link of the relative clause to the subject noun phrase of the matrix clause. This arc is then retained during processing of the subsequent verb phrase, *'chased,'* to preserve position of the embedded clause relative to the matrix in the constituent hierarchy. Processing from this point on is conducted in a straightforward, deterministic fashion, yielding acceptance of the sentence as the verb phrase licensed by the arc <NP, ∘, VP, S> is recognised in the final lexical item, *'ran.'*

The foregoing analysis suggests that the NCA possesses temporary memory resources which are manifested in two forms, one associated with each of the two processing principles identified in the definition of the machine, namely, the category affiliation and category inclusion principles. The latter yields an aspect of temporary memory devoted to preserving the category inclusion hierarchy which is necessary to maintain structural relationships within a sentence. The former is associated with that element of the memory required to connect fragments of constituents which are interrupted by other, intervening constituents such as embedded clauses.

Thus, an NCA does not possess a peripheral storage device, such as a push-down store, or other memory facility of this nature. Rather, temporary memory capacity is accommodated as an aspect of the architecture it employs to recognise individual lexical items and license their syntactic environment. Consequently, the memory function is realised in the adaptation of this architecture to the licensing of non-adjacent constituents.

Few listeners experience significant difficulty accepting such constructs as are realised in sentences with a single embedded relative clause. Hence, we may confidently conclude that neither the space nor the memory resources consumed by the NCA exceed those available to the language processing regions of the brain. In this example, the absence of relativisers, such as *'that'* in *'a cat that the dog*

Input	State
a cat	*NP* (∘,S) NP (∘, VP) S
the dog	*NP* (∘,S) NP (∘, VP) S (*NP*,∘) NP (∘, VP) S
Mary	NP (∘,VP) S (*NP*,∘) NP (∘, VP) S (*NP*,∘) NP (∘, VP) S
owned	VP (NP,∘) S (*NP*,∘) NP (∘, VP) S (*NP*,∘) NP (∘, VP) S
chased	VP (NP,∘) S (*NP*,∘) NP (∘, VP) S
ran	VP (NP,∘) S

Figure 11: States of an NCA accepting an English sentence with two levels of clause embedding.

chased ran,' would appear to exert minimal influence on processing facility. Regardless whether or not relativisers are present, listeners do exhibit substantial difficulty in processing a sentence such as *'a cat the dog Mary owned chased ran,'* which includes two embedded relative clauses, and hence, possesses the backbone structure depicted in Figure 9(b). The states experienced by an NCA in the course of its accepting this sentence are recorded in Figure 11.

During recognition of the initial item, *'a cat,'* the machine identifies the arc <NP, ∘, VP, S>, licensing the predicate verb phrase (sans-serif bold in the figure) corresponding to the last token, *'ran,'* of the sentence. This arc must now be stored and "carried forward" over both of the two relative clauses in order that the sentence be accepted. With recognition of the second item, *'the dog,'* which is the first in the first embedded clause, another such arc is identified licensing the verb phrase (sans-serif bold in the figure) *'chased,'* and this one must be carried forward over the second relative clause. Thus, while it is processing the second embedded clause, the machine is effectively storing two elements in that aspect of its memory operation which is associated with the category affiliation processing principle. Also, with the recognition of the second token, *'owned,'* of the second relative clause, the machine is employing the memory function associated with the category inclusion principle to store two arcs of the form <S, NP, ∘, NP>, whose purpose has "expired" following completion of their function to link each relative clause to the NP category it modifies (identified with italics in the figure). Finally, during its processing of the second embedded clause, *'Mary owned,'* the two states of the NCA each consist of five output arcs.

The difficulty listeners experience recognising this sentence likely can be attributed to its exceeding the temporary memory capacity of the language processing system; however, before examining the memory demands of such sentences, we first assess their processing space requirements. It seems probable that there are limits on the analogue of space in the language processing mechanism of the brain and that these limits will affect its ability to recognise sentences. Hence, we might speculate that the difficulty of sentences such as the example in Figure 11. stems from limitations on the number of contemporaneous configurations that the neural system can maintain during the processing of individual lexical items. The implementation mapping associates these neural configurations with the categories comprising the arcs instantiated in the states of an NCA. Thus, it would appear that the resources of the automaton, as evidenced in its employing up to five arcs while recognising lexical items in the example, exceed an upper bound on neural capabilities. This bound must correspond to at least three arcs, however, because that is the maximum number in the states of an NCA recognising sentences with a single embedded relative clause, which listeners normally exhibit minimal problems accepting.

Imposition of a bound on the space resources of an NCA such that its states may consist of at most four output arcs causes the machine to fail to accept the English sentence *'a cat the dog Mary owned chased ran.'* The fifth arc, which would be instantiated in the state of the machine as it attempted to recognise *'Mary,'* subject of the second embedded clause, could not be accommodated. Consequently, the affiliation this arc expresses, which licenses the verb phrase *'ran'* of the matrix clause, would be lost. In effect, the subject noun phrase cannot be linked with the verb phrase, and hence, the sentence cannot be recognised. This processing fault would then be manifested as a violation of the category affiliation principle, and likely be experienced as such by the listener, notwithstanding the failure is precipitated by a limit on the number of configurations which the neural system can maintain concurrently.

That a bound corresponding to four arcs might be reasonable is supported by the success Japanese listeners evidence in accepting sentences such as *'John ga Mary ga syoonen ni kaita tegami o yonda'* ('John read the letter Mary wrote to the boy') with the backbone structure depicted in Figure 9(c). The single embedded relative clause, *'Mary ga syoonen ni kaita'* ('Mary boy to wrote'), is attached to the predicate noun phrase, *'tegami o'* ('letter'), of the matrix clause *'John ga tegami o yonda'* ('John letter read'). States of an NCA accepting this sentence are shown in Figure 12.

In its processing of the relative clause, the machine requires states consisting of four output arcs. In the recognition path for the subject NP of the relative clause, *'Mary ga,'* the fourth and last arc, <VP, NP, ∘, S>, links the subject NP, *'John ga,'* of the matrix clause to its predicate. This link is not required during the processing of subsequent tokens; however, the second and third arcs, <S, ∘, NP, NP> and <NP, ∘, V, VP>, respectively, must be carried forward during processing of the predicate phrase, *'syoonen ni kaita,'* of the relative clause. (Labels of the licensed categories, **NP** and **V**, are shown in sans-serif bold in the figure.) They become the third and fourth arcs during the recognition of these items. The arc <S, ∘, NP, NP> is

Input	State
John ga	NP (o,VP) S
Mary ga	*NP* (o,VP) S (o, NP) NP (o, V) VP (NP,o) S
syoonen ni	PP (o,V) VP (*NP*,o) S (o, NP) NP (o, V) VP
kaita	V (PP,o) VP (*NP*,o) S (o, NP) NP (o, V) VP
tegami o	NP (S,o) NP (o, V) VP
yonda	V (NP,o) VP

Figure 12: States of an NCA accepting a Japanese sentence with an embedded relative clause.

necessary to link the relative clause to the predicate phrase NP, *'tegami o,'* it modifies in the matrix clause, and the arc <NP, o, V, VP> links the predicate phrase NP of the matrix clause to its verb, *'yonda.'* Furthermore, the second arc, <VP, NP, o, S>, instantiated during the recognition of *'syoonen ni'* to link the subject NP (italics in the figure) to the predicate VP of the relative clause, must be carried into the state realised during the recognition of the verbal category item *'kaita.'*

Processing of the verbal item requires four arcs, with one of these devoted to preserving a category inclusion relation, and two of them employed to store category affiliation for non-adjacent categories. The remaining arc of the four is needed to connect adjacent categories. Since listeners experience minimal difficulty accepting this sentence, it is evident that the language processing system must be capable of sustaining simultaneously the configurations which map into the categories represented in up to four output arcs in the states of the automaton. While not establishing an upper bound, this demonstration does confirm that the language processing system requires the analogue of space during a recognition, and provides some indication of its extent.

We now return to the issue of temporary memory requirements. The temporary memory of the automaton is realised as the retention of arcs in its states which preserve category inclusion hierarchies and maintain affiliations for non-adjacent constituents during the processing of intervening categories. We assume that the temporary memory of the language processing system takes the form of neural configurations which persist during the recognition of several lexical items. The implementation mapping associates these configurations with categories comprising the arcs retained by the automaton. We conjecture that recognition difficulties can be attributed to the inability of the language processing system to sustain those neural configurations. They decay, the category affiliation and inclusion relations they express are lost, and consequently, the sentence cannot be accepted.

In accepting the Japanese sentence with one embedded relative clause, the machine retains up to three output arcs. One of these preserves a category inclusion relation which persists during the recognition of just one other item. The other two arcs express affiliations for subsequent constituents. The first-instantiated of these must be retained during the processing of three intervening items before the category it licenses is identified, and the second is retained as two subsequent items are recognised before its affiliation is satisfied. While processing the English sentence with two embedded clauses, the machine must retain up to four arcs. Two of these maintain category affiliations, the first-instantiated of which must be retained as four intervening items are recognised before the category it licenses is encountered. The second of these arcs licenses a category which is separated by three items. Of the two arcs which preserve inclusion relations, one is sustained during the processing of three subsequent items, and the other is retained as one further item is recognised.

In order that listeners accept these sentences, neural configurations corresponding to the arcs which have been identified must be sustained throughout the course of processing the utterance. The greater difficulty of the English sentence might therefore be attributed to the relatively longer durations over which neural configurations must persist to achieve recognition. One might infer that these configurations possess a limited lifetime. Hence, attenuation of the configurations mapping into the arc instantiated in recognising the subject of the matrix clause, *'a cat,'* which must be retained during the processing of four intervening items, would jeopardise licensing of the matrix clause verb phrase, *'ran.'* Listeners would experience this situation as a confusion regarding where the final verb might be attached in the structure already processed. Extinction of the configuration would precipitate their failure to recognise the sentence.

These circumstances could be modelled in the automaton by incorporating the condition that an arc be removed if the category it licenses is not encountered after a specified number of items have been processed. Evidence provided by the Japanese sentence with one embedded clause suggests a lifetime corresponding to three intervening items.

Other explanations for the difficulty of sentences with two embedded clauses are possible. For example, neural configurations licensing categories not yet encountered, such as the verb phrase of the matrix clause, might persist, albeit with much attenuated strength, while configurations expressing affiliations which have already been satisfied might decay more rapidly. These configurations correspond to arcs which are instantiated to attach relative clauses to the noun phrases they modify. Once this linking has been effected, the configurations might evaporate during the processing of later lexical items, entailing loss of the structural information relating an embedded clause to its matrix. Hence, although configurations licensing the matrix verb phrase might persist, information regarding where they must be applied is lost. Listeners would experience a "which goes with what" quandary as they are unable to associate the subject noun phrases with the appropriate verbs.

We speculate that the attenuation of configurations may in fact be hastened by memory management strategies which could have evolved to enable the language processing system to optimise its resources. Such strategies could well include the re-use of those resources employed in the recognition of a lexical item and which experience has taught would likely not be required in processing subsequent items. In other words, some configurations might not decay; they could actually be extinguished, likely with a lag corresponding to the processing of one or two subsequent items to ensure that the information the configurations contain is indeed not required. While normally successful, application of this heuristic during the processing of sentences with two embedded clauses could precipitate problems. This explanation is supported by the evidence that listeners can learn to process successfully particular instances of sentences with two embedded clauses, suggesting that the automatic application of the heuristic can be suspended and the necessary configurations retained.

Implementation of this heuristic into the operations of the automaton could entail elimination of any arc expressing an affiliation which has already been satisfied following the recognition of two subsequent lexical items. Acceptance of the English and Japanese sentences with one relative clause would not be endangered because category inclusion arcs need not be retained beyond the processing of one additional item. In the case of the English sentence with two embedded clauses, however, application of this strategy would see elimination of the arc <S, NP, o, NP>, instantiated during recognition of the item *'the dog,'* but which must be propagated forward over three subsequent items into the state of the automaton realised in its recognition of *'chased.'* Since this arc preserves inclusion of the subordinate clause in the matrix clause, its loss removes information required to process the sentence successfully.

In order to reflect capabilities of the neural system, and reproduce the difficulties listeners exhibit in their processing of embedded structures, constraints must by applied to this memory function. Imposition of any limit on its resources, however, yields a machine which is no longer equivalent to a context-free grammar. The device becomes a variety of finite machine such as those introduced by Pereira & Wright (1991), Pulman (1986), and Tugwell (1995). Like these devices, a constrained NCA accepts only those sentences with space complexity which does not exceed the capacity of the human language processing mechanism.

The foregoing observations do not enable us to establish incontrovertible bounds on the temporary memory and space resources of an NCA. Nor is it likely, given human variability, that universally applicable limitations can be determined, although one would hope that further investigation will permit estimation of ranges for such bounds. Preliminary investigation does suggest, however, that the operations of an NCA constitute a credible model for the mechanisms of the temporary memory and storage functions of the language processing systems of the brain; and hence, we conjecture that estimated ranges for the resources of the automaton will reflect actual bounds on the capabilities of the neural system.

6 Conclusions

Examination of the operations undertaken by the automaton we have introduced reveals that the recognition of left-branching structures entails no greater temporary memory and space capacity than the acceptance of right-branching structures: they in fact require entirely comparable processing resources, consistent with evidence provided by the speakers of left- and right-branching languages. Speakers of both classes of language, however, experience comparable degrees of difficulty in recognising sentences with embedded relative clauses. Analysis of the operation of an NCA accepting such sentences illuminates the source of the problem as being attributable to their demands on temporary memory, and perhaps on the analogue of processing space in the neural system.

The automaton we have devised, and the analyses we have undertaken employing it, are based upon the constituent-structure model of natural language. While the details and appropriate implementation of this model are the subject of active enquiry and debate, the psychological reality of structural constituents is supported by empirical evidence. Our implementation of the constituent-structure model of natural language, however, is adapted to processing rather than analysis, and consequently is somewhat at variance with more conventional applications directed principally toward grammatical description. This approach is concordant with the philosophy espoused by Shieber (1988) that the exigencies of processing must supervene the notations and "notions" of analytical formalisms[10].

We contend that the organisation and functioning of an NCA are more nearly consonant with the architecture of the brain and the mechanisms it exploits to process language. Our adaptation of the CSA might not be in complete accord with Chalmers' (1996a, 1996b) purposes; however, to the extent that the automaton might afford a faithful reflection of neural mechanisms, we maintain that a variant of the CSA such as we have devised comprises a potent conceptual instrument for the investigation of human language processing systems.

References

[1] Chalmers D. J. (1996a) Does a Rock Implement Every Finite-State Automaton? *Synthese*, 108, 3, p. 309–333.

[10]Notwithstanding the admonition of Reyle & Rohrer (1988, p. 18) that Shieber's arguments be taken *cum grano salis*.

[2] Chalmers D. J. (1996b) *The Conscious Mind: In Search of a Fundamental Theory*. Oxford: Oxford Univ. Press.

[3] Chomsky N. (1959) On Certain Formal Properties of Grammars. *Information and Control*, 2, p. 137–167.

[4] Chomsky N. (1962) Context-Free Grammars and Pushdown Storage. *MIT Research Laboratory of Electronics, Quarterly Research Report*, No. 65, p. 187–194.

[5] Clark H. H. & Wilkes-Gibbs D. (1986) Referring as a collaborative process. *Cognition*, 22, p. 1–39.

[6] Earley J. (1970) An Efficient Context-free Parsing Algorithm. *Communications of the ACM*, 14, p. 453–460. Reprinted in B. J. Grosz, K. Sparck Jones & B. L. Webber (eds), *Readings in Natural Language Processing*, 1986. Los Altos, U.S.A.: Morgan Kaufmann, p. 25–33.

[7] Fodor J. A., Bever T. & Garrett M. F. (1974) *The Psychology of Language: An introduction to psycholinguistics and generative grammar*. New York: McGraw-Hill.

[8] Garrett M. F., Bever T. & Fodor J. A. (1966) The Active Use of Grammar in Speech Perception. *Perception and Psychophysics*, 1, p. 30–32.

[9] Gazdar G., Klein E., Pullum G. & Sag I. (1985) *Generalized Phrase Structure Grammar*. Cambridge, U.S.A.: Harvard Univ. Press.

[10] Hawkins J. A. (1990) A Parsing Theory of Word Order Universals. *Linguistic Inquiry* 21, p. 223–261.

[11] Johnson N. (1965) The Psychological Reality of Phrase Structure Rules. *Journal of Verbal Learning and Verbal Behavior*, 4, p. 469–475.

[12] Johnson N. (1966) On the Relationship between Sentence Structure and the Latency in Generating the Sentence. *Journal of Verbal Learning and Verbal Behavior*, 5, p. 375–380.

[13] Karttunen L. (1990) Radical Lexicalism. In M. R. Baltin & A. S. Kroch (eds), *Alternative Conceptions of Phrase Structure*. Chicago: Univ. of Chicago Press, p. 43–65.

[14] Kay M. (1980) Algorithmic Schemata and Data Structures in Syntactic Processing. Reprinted in B. J. Grosz, K. Sparck Jones & B. L. Webber (eds), *Readings in Natural Language Processing*, 1986. Los Altos, U.S.A.: Morgan Kaufmann, p. 35–70.

[15] Kess J. F. & Nishimitsu Y. (1990) *Linguistic Ambiguity in Natural Language: Japanese and English*. Tokyo: Kuroshio Shuppan Publishing.

[16] Kuno S. (1973) *The Structure of the Japanese Language*. Cambridge, U.S.A.: MIT Press.

[17] Mazuka R. & Lust B. (1988) Why is Japanese Not Difficult to Process?: A Proposal to Integrate Parameter in Universal Grammar and Parsing. *The Proceedings of NELS*, 18, p. 333–356.

[18] Pereira F. C. N. & Wright R. N. (1991) Finite-State Approximation of Phrase Structure Grammars. *Proc. 29th Meeting of the Assoc. Comput. Ling.*, Berkely, California, p. 246–255.

[19] Pollard C. & Sag I. A. (1994) *Head-Driven Phrase Structure Grammar*. Chicago: Univ. Chicago Press.

[20] Pulman S. G. (1986) Grammars, Parsers, and Memory Limitations. *Language and Cognitive Processes*, 1, 3, p. 197–225.

[21] Putnam H. (1988) *Representation and Reality*. Cambridge, U.S.A.: MIT Press.

[22] Révész G. E. (1983) *Introduction to Formal Languages*. New York: McGraw-Hill.

[23] Reyle U. & Rohrer C. (1988) Introduction. *Natural Language Parsing and Linguistic Theories*. Dordrecht, Holland: D. Reidel, p. 1–32.

[24] Shieber S. M. (1988) Separating Linguistic Analyses from Linguistic Theories. In U. Reyle & C. Rohrer (eds), *Natural Language Parsing and Linguistic Theories*. Dordrecht, Holland: D. Reidel, p. 33–68.

[25] Schober M. F. & Clark H. H. (1989) Understanding by addressees and overhearers. *Cognitive Psychology*, 21, p. 211–232.

[26] Tugwell D. (1995) A State-Transition Grammar for Data-Oriented Parsing. *Proc. 7th Conf. European Chapter of the Assoc. Comput. Ling.*, Dublin, Ireland, p. 272–277.

[27] Wilkes-Gibbs D. & Clark H. H. (1992) Coordinating beliefs in conversation. *Journal of Memory and Cognition*, 31, p. 183–194.

[28] Yngve V. H. (1960) A Model and an Hypothesis for Language Structure. *Proc. American Philosophical Society*, 104, 5, p. 444–466.

Petri nets and IDEF diagrams: Applicability and efficacy for business process modelling

Vesna Bosilj-Vuksic
University of Zagreb, Faculty of Economics, Department of Business Computing
Trg J.F.Kennedya 6, 10000 Zagreb, Croatia
vbosilj@efzg.hr

Vlatka Hlupic
Brunel University, Department of Information Systems and Computing
Uxbridge, Middlesex UB8 3PH, United Kingdom
Vlatka.Hlupic@brunel.ac.uk

Keywords: Business Process Reengineering (BPR), business process modelling, simulation modelling, IDEF0 diagrams, IDEF3 diagrams, Petri nets, DES-nets

Received: October 4, 2000

It is apparent that developing dynamic models of business processes prior to their radical change could increase the success of BPR projects. This paper investigates a suitability of IDEF diagrams and Petri Nets for modelling business processes. Information modelling and simulation modelling are discussed from the business process re-engineering perspective. Examples of business process modelling using IDEF diagrams and Petri nets are presented. The suitability of these two graphical methods for business process modelling is discussed, and a comparison of usage of these two methods for BPR is provided.

1 Introduction

In order to survive in a competitive economic environment, many organisations need to continuously improve their business processes. This can be facilitated by using various methods and tools for business process modelling, so that any changes to business processes can be tested on models. Business Process Re-engineering (BPR) has become one of the most popular topics in organisational management creating new ways of doing business (Tumay, 1995). This management concept relates to the radical redesign of business processes in order to achieve more efficient, higher quality and more competitive production (Hammer and Champy, 1993). It is also a method of improving the operation and therefore the performance of organisations, enabling analysing and altering the business processes of the organisation as a whole (Kovacic and Vintar, 1998).
An important initial activity for BPR projects is to acquire descriptions of the relevant business systems and to develop "AS-IS" model of the company's processes. "AS-IS" model (model of current business processes) provides BPR participants with the information needed to decide what to change, how to change and what will be the result of the change. The next phase is the development of "TO-BE" models that represent both existing and alternative processes. These models must be validated and tested before the implementation. They can be used to predict characteristics that cannot be directly measured and to predict economic and performance date that otherwise would be expensive or impossible to acquire.

Growing interest amongst academic and industrial communities in organizational change and business process re-engineering has resulted in a multitude of approaches, methodologies, and techniques to support these design efforts (Wastell *et al.*, 1994), (Harrison and Pratt, 1993). Kettinger *et al.* (1997) conducted an empirical review of existing methodologies, tools, and techniques for business process change and developed a reference framework to assist positioning of tools and techniques that help in re-engineering strategy, people, management, structure, and technology dimensions of business processes.

Different methods are used for analysis and/or modelling of business processes such as: IDEF diagrams, Activity Based Costing Method (ABC), Total Quality Management (TQM), benchmarking, simulation and Workflow analysis. Some of the frequently mentioned problems related to BPR include the inability to accurately predict the outcome of a radical change, difficulty in capturing existing processes in a structured way, shortage of creativity in process redesign, the level of costs incurred by implementing the new process, or inability to recognise the dynamic nature of the processes.

This paper investigates the suitability of IDEF diagrams and Petri nets for business process modelling. A discussion on business processes related issues and an overview of business process modelling methods are presented. IDEF diagrams and Petri nets are described and compared according to their usage criteria and their basic elements. An example of business process modelling using both methods is provided, and the suitability of this method for modelling business processes is investigated. It is shown that these two methods are complementary and can be combined and jointly used as a powerful tools to support the BPR project.

The paper is structured as follows. Following a discussion on information system and simulation modelling, the basic principles of IDEF diagrams and Petri nets are presented. Examples of business process modelling using IDEF diagrams and Petri nets are further provided. These methods were compared and their suitability for business process modelling is discussed. Conclusions outline the main findings of this research.

2 Information system modelling and simulation modelling as a support for BPR

Business processes can be defined as a series of logically connected activities that use the company's resources. Davenport and Short (1990) defined a process as "a structured, measured set of activities designed to produce a specified output for a particular customer or market. It implies a strong emphasis on how work is done within an organization. There are numerous other definitions of business processes available in the literature (Hlupic and Robinson, 1998). Some common elements can be identified in a majority of definitions. These elements relate to the process itself (usually described as transformation of input, work flow, or a set of activities), process input, and process output, usually related to creating value for a customer, or achieving a specific goal (Paul *et al.*, 1998).

One of the important methods in the implementation of BPR is information system modelling. The other is simulation modelling that is process-oriented, and thus fits naturally with the BPR concept. These two methods are discussed in the subsequent sections.

2.1 Information system modelling and BPR

Awareness of IT capabilities should influence the design of business processes. In addition to investing in information technology, a new type of information systems models should be designed. The structure of information system model could be divided into the static and the dynamic part. The static structure of the model consists of functions, human and other resources, while the dynamic part consists of data, processes and events. The dynamic structure of information systems demands the implementation of process-oriented methods and tools.

Process models are often developed by graphical software tools that show business processes, activities and participants with flow diagrams and process charts. A disadvantage of these tools is that they are unable to perform process analysis. Process modelling tools must be able to show interconnections between the processes and to conduct a decomposition of the processes. These tools must help users to conduct "what-if" analysis and to identify and map nonvalue steps, costs, and process performance (bottleneck analysis). They should be able to develop "AS-IS" and "TO-BE" models of business processes.

The ability to support these requirements makes IDEF methods and tools valuable in BPR. IDEF methods and tools use visual models that facilitate the quantitative analysis of proposed changes to processes to achieve the highest performance at the lowest costs (deWitee and Pourteau, 1997).

2.2 Simulation modelling and BPR

Simulation has an important role in modelling and analyzing the activities in introducing BPR since it enables quantitative estimations on influence of the redesigned process on system performances (Bhaskar *et al.*, 1994). Recent development in simulation software made simulation particularly suitable to use in BPR (Van Ackere *et al.*, 1993). A re-engineering business process involves changes in people, processes and technology. As these changes happen over time, simulation appears to be a suitable process modelling method. Simulation is often called a technique of last resort because it is used when the system to be modelled is too complex for analytical models (Oakshot, 1997). The interaction of people with processes and technology results in an infinite number of possible scenarios and outcomes that are not possible to predict and evaluate using widely popular static process modelling methods. Kettinger *et al.* (1997) mention simulation as one of the modelling methods in their survey on business process modelling methods.

Reasons for introducing simulation modelling into process modelling can be summarized as follows:

- simulation enables modelling of process dynamics,

- influence of random variables on process development can be investigated,
- anticipation of reengineering effects can be specified in a quantitative way,
- process visualization and animation are provided,
- communication between clients and analyst is facilitated by simulation models.

Modern simulation software tools are able to model dynamics of the processes and show it visually, which then can enhance generating the creative ideas on how to redesign the existing business processes. Modern simulation software includes graphic user interface (GUI) that enables process animation and graphical display of simulation results.

One of the methods that could be used for modelling business processes is Petri nets. This method, described in more detail in Section 4, can allow a graphical representation of the dynamics and structure of business processes. The following section describes business process modeling method related to IDEF diagrams.

3 Business process modelling using IDEF diagrams

IDEF (Integrated Definition) diagrams, based on SADT (Structured Analysis and Design) diagrams, were introduced in 1981 as an integrated part of Integrated Computer-Aided Manufacturing (ICAM) project (Marca and Gowan, 1988). There are numerous IDEF methods, but two of them serve as the basis for simulation models: IDEF0 method that focuses on activities modelling and IDEF3 method that accomplishes process description and can be used to rapidly generate discrete-event simulation model specifications (Mayer *et al.*, 1998).

3.1 IDEF0 diagrams

IDEF0 is a modelling technique used frequently in Computer Integrated Manufacturing systems analysis to get a good understanding of the system before BPR begins (Withers *et al.*, 1993). IDEF0 model in the rectangle represents the activity (Figure 1). The left arrows in the rectangle display inputs, whilst the right arrows display outputs. The top arrows show constrains (controls) that start or stop all activities, and the bottom arrows display the resources (mechanisms) that participate in converting inputs into outputs.

3.2 IDEF3 diagrams

IDEF3 diagram is a process-flow modelling method describing how activities work together to form a process (deWitte and Pourteau, 1997; Mayer *et al.*, 1998). IDEF3 diagram identifies the behavior of the system. It builds structured descriptions about "what" a system actually does and "how" activities work together to form a process. There are two description modes: *Process Flow Diagram* and *Object State Transition Network.*

An IDEF3 process flow describes a process and the relations that exist between processes. The activities of the process appear as labeled boxes. The term for elements represented by boxes (activities, processes, events, operations, procedure) is a Unit Of Behavior (UOB). The boxes are connected by arrows that define the logical flows. The arrows are the same as in IDEF0 diagrams, but there are no bottom arrows (mechanisms). There are also the smaller boxes that define logic junctions: AND (&), OR (O), and exclusive OR (X). Logic junctions could present asynchronous or synchronous behavior among UOBs (they present inputs that proceed and outputs that follow the UOB). Each UOB can be decomposed and can be associated with a description of the objects and their relations, called elaboration.

The Object State Transition Network summarizes all the transitions an object may undergo throughout a particular process that is very useful for simulation modelling.

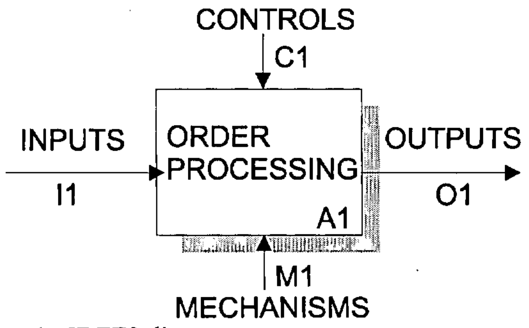


Figure 1: IDEF0 diagram

4 Simulation modelling and Petri Nets

Petri nets are a method which enables graphical modelling of system behavior simultaneously enabling introduction of mathematical formal rules for system behavior definition (Törn, 1985; Yao, 1994; Oberweis and Sänger, 1992). This method can be used for modelling of any system where the regulation of object and data flow is important. Petri nets are one of the most used methods in modelling of parallel dynamic systems because of their characteristics: simplicity, representation power comprising concurrency, synchronization and resource sharing, strong ability of their mathematical analysis and application of software tools (Kamper, 1989; Thomas, 1993; David and Alla, 1994; Ceric, 1995; Dietz and Barjis, 2000).

The example of standardized Petri nets are ***Discrete Event Simulation nets*** or ***DES-nets*** (Ceric, 1995). DES-nets incorporate some simulation graphs extensions already mentioned such as arcs with weight, timed transitions and inhibitor arcs. DES-nets incorporate also additional extensions, such as three types of *decision rules*. Priority rules assign priorities to all output transitions (lower priority value means higher priority). Probability rules assign probabilities to all output transitions (the sum of probabilities is equal to 1). Conditional rules provide a condition to be evaluated at the moment of decision making. DES-nets use the following elements of coloured Petri nets: token colours, token colours sets, place with inscription (coloured place), arc with inscription (coloured arc) and transition with a guard.

5 An example of business process modeling using IDEF diagrams and Petri nets

In order to demonstrate the suitability of IDEF diagrams (IDEF0 and IDEF3) and Petri nets (DES-nets) for business process modelling, an example of modelling selling processes using those methods is provided.
IDEF0 diagram in Figure 2 shows the highest hierarchical level of a simple selling process model. This example was chosen because of its simplicity enables an easier comparison of the two methods, where the emphasis is on demonstration the use of Petri nets and IDEF diagrams for business process modelling rather than to model examples of complex real-life processes. Selling process is divided into three elementary basic activities: processing the order, dispatch of goods and invoicing.

Order is the input data for the "processing order" activity. The sales department is involved in this activity and the control mechanism is used to compare the quantity of ordered goods with the quantity on stock. Output data is the order for dispatching goods. The order is at the same time input data for the "dispatch of goods" activity. In the dispatch activity warehouse staff takes part in and the control of the dispatched goods takes place (comparison of ordered and actually dispatched goods). Output data are: the goods delivered to the customer, a copy of the accounts dispatch list which initiates the invoicing and a copy of dispatch list to the customer. In the "invoicing" activity participates the account clerk, who at the same time settles the dispatch list and makes the price calculation. Output data is the invoice to the customer.

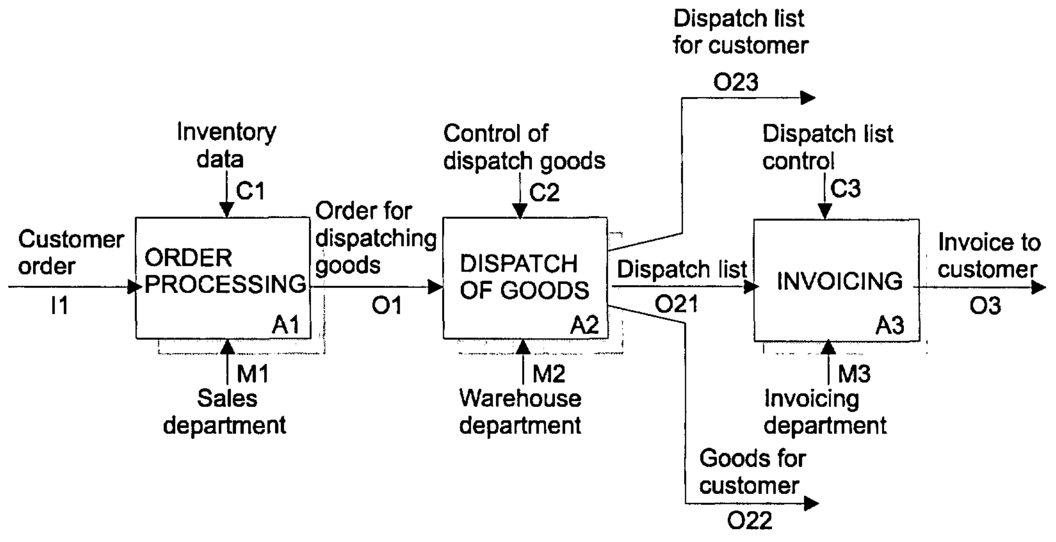
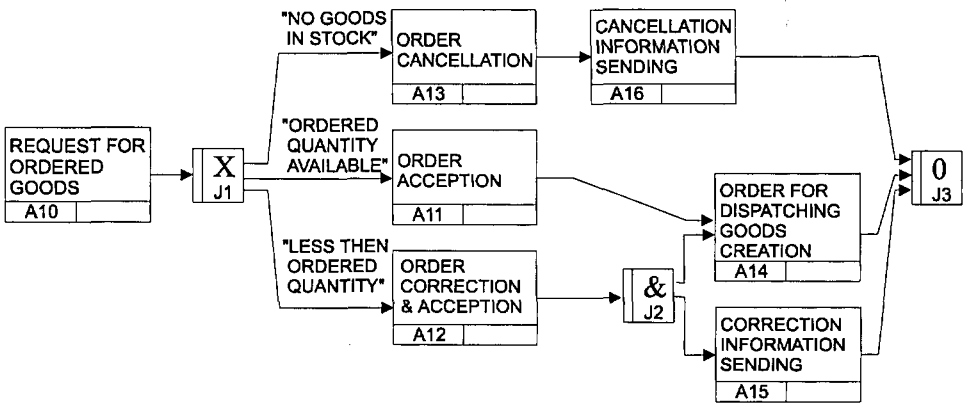


Figure 2: IDEF0 diagram of the simple selling process

IDEF3 diagram shown in Figure 3 is used for detailed representation of the "order processing" activity. It represents the decomposition of A1 UOB. The "processing order" activity consists of several activities. The first one is the "request for ordered articles". It represents a comparison of the ordered quantities with the level of inventory that can result in three exclusive activities. These activities are connected with the preceding activity A1 by logic junction J1 (asynchronous, exclusive OR). This junction means that the preceding activity A1 must complete first, before exactly one of the following activities will start. If the ordered quantity is available, the order is confirmed (A11) and the order for dispatching goods is created (A14). If there is less than ordered quantity, the order will be corrected and accepted (A12) and then two activities will start simultaneously:

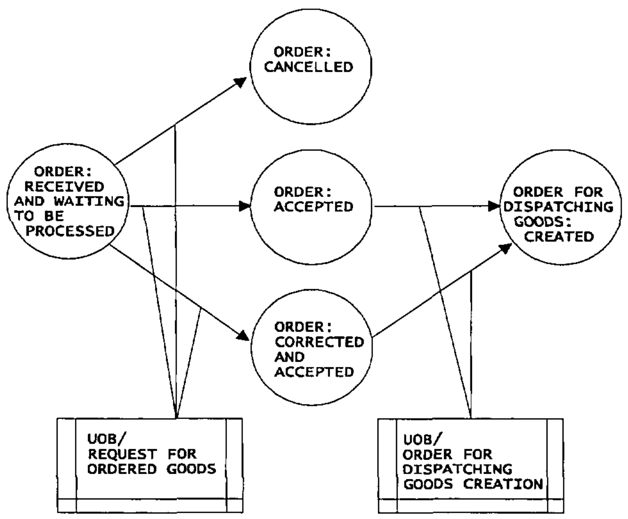
A14 and A15 (the information about correction is sent to the customer). It is shown by the asynchronous AND junction (all preceding processes must complete and all following processes will start). If there are no goods in stock, the order is cancelled (A13) and the information about cancellation is send to the customer (A16).



REQUEST FOR ORDERED GOODS
A10
X
J1
"NO GOODS IN STOCK"
ORDER CANCELLATION
A13
"ORDERED QUANTITY AVAILABLE"
ORDER ACCEPTION
A11
"LESS THEN ORDERED QUANTITY"
ORDER CORRECTION & ACCEPTION
A12
&
J2
CANCELLATION INFORMATION SENDING
A16
ORDER FOR DISPATCHING GOODS CREATION
A14
CORRECTION INFORMATION SENDING
A15
0
J3

Figure 3: IDEF3 process flow diagram of the "order processing" activity

Entities (objects) and their states are explicitly shown by Object State Transition Network. Object states are represented by circles and object state transition arcs are represented by the lines connecting the objects. Figure 4 presents all the states of the "order" entity in the "order processing" activity. The order can be received, cancelled, accepted or accepted with the correction. Certain conditions must be fulfilled, or certain events must happen before the object state is changed. Between the "received order" state and the "accepted order" state the "request for ordered goods" must be finished. It is the entry condition that needs to be met before an object can transition from one to the another state. The exit condition characterizes the event under which an object transitions out of a state. The "accepted" order can become "the order for dispatching goods" if the event of its creation is finished.



ORDER: CANCELLED
ORDER: RECEIVED AND WAITING TO BE PROCESSED
ORDER: ACCEPTED
ORDER FOR DISPATCHING GOODS: CREATED
ORDER: CORRECTED AND ACCEPTED
UOB/ REQUEST FOR ORDERED GOODS
UOB/ ORDER FOR DISPATCHING GOODS CREATION

Figure 4: The example of "order" Object State Transition Network diagram

Application of simulation modelling requires defining the following elements: resource capacities, the time duration of activities, rules and probability of activities occurring, and the dynamic of entities coming into the system. Figure 5 shows DES-nets developed for the previously defined scenario of the simple selling process.

The arrival of orders is generated. Orders arrive every t_t minutes (the value of the number t_t is generated by a random number generator). After the arrival occurs, one token is again deposited back to the "outside" place and another one to the "orders waiting" place. There are two conditions which must be fulfilled to initiate the "order processing" transition: (a) at least one token in the "orders waiting" place and (b) at least one token in the "salesman ready" place (initially there are 5 salesmen). The "order processing" transition fires and after t_o

minutes one token of type (N,K) is deposited in the "ready for control" place.

Depending on the probability rule it fires one of the transitions: "order acceptation", "order correction and acceptation" or "order cancellation". If the "order cancellation" transition fires, the salesman becomes free. In the other two cases one token of type K is deposited in the "salesman ready" place, while one token of type (N) participates in the "dispatch order creation" transition. After the firing of this transition one token is deposited in the "waiting for dispatch" place. The "goods dispatch" transition fires if there is: (a) at least one token of type (N) in the "waiting for dispatch" place, (b) two tokens of type S in the "warehouseman ready" place and (c) at least one token of type V in the "forklift truck ready" place.

This transition is finished after t_f minutes and then two tokens of type S are deposited to the "warehouseman ready" place, one token of type V moves to the "forklift truck free" place, one token of type N (representing the order, the copy of the dispatch list and the goods) leaves the system, while one token of type D (representing the dispatch list) is deposited to the "waiting for invoicing" place. If there is a free accounts clerk (at least one token of type A in the "accounts clerk ready" place) the "invoicing" transition fires that completes the process.

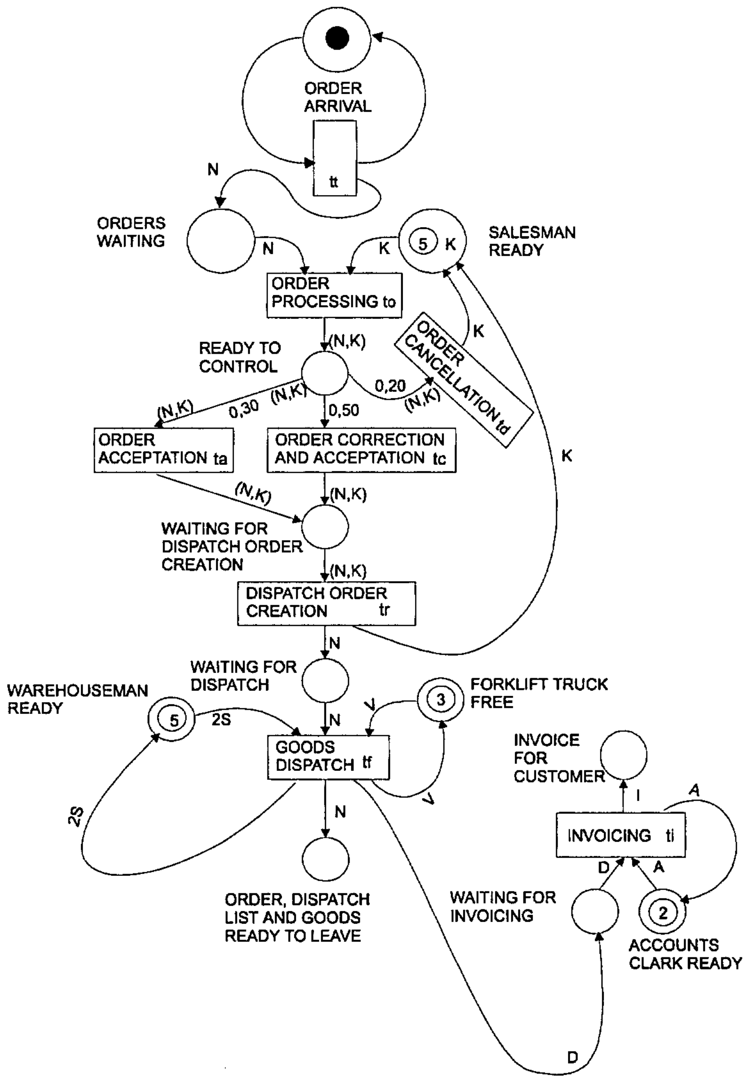


Figure 5: DES-net of the simple selling process

6 A comparison of IDEF diagrams and Petri nets

In this research IDEF diagrams were selected and proposed because:

- they are used widely, especially for business process analysis and modelling (Pinci and Shapiro, 1993),
- they represent the only standard modelling and analysis methods for enterprise engineering (deWitte and Pourteau, 1997, Tatsiopoulos *et al.*, 1999),

IDEF diagrams support particular reengineering activities such as simulation modelling and information system modelling (Gladwin and Tumay, 1994). IDEF diagrams provide a mechanism for analyzing and documenting processes. They are designed to model decisions, actions and activities of an organization or a system. IDEF modelling is a very effective tool for communication between the analyst and the participants of the processes. IDEF diagrams explicitly show activities. Entities are shown with the data flow, whilst resources are presented implicitly, throughout the mechanisms. They can not represent all the elements important for simulation modelling, such as queues, random behavior and process dynamics, but could provide the basic elements for simulation model development. Only two IDEF diagrams are presented in the paper: IDEF0 and IDEF3. Due to their similarities, but also the differences, they could be conveniently used together.

IDEF0 diagrams support the following functions:

- identifying basic elements of the process,
- identifying core processes,
- enabling hierarchical representation of system structure,
- helping focus attention on *what* happens in an organization.

IDEF3 diagrams accomplish the following:

- describing *how* process work,
- providing hierarchical representation and decomposition of the model,
- facilitating Top-down and Bottom-up modelling,
- providing both: "process-centered" and "object-centered" perspective,
- managing timing and decision logic of the process that is important for simulation modelling.

The example presented in Figure 5 shows that Petri nets are fairly simple since they use a limited number of symbols, but have large power of representation of system complexities. They can use hierarchical structures (each of the activities can be represented as a detailed simulation graph). Petri nets present activities and events by transitions while entities and resources are shown by tokens. There is a special symbol representing the queue, whilst control mechanisms are included by using conditions for transitions firing.

Petri nets are in particular well-suited for systems in which communication, synchronization and resource sharing are important since they have powerful abilities for representation of system dynamics: entity arrivals dynamics, availability of resources, interdependencies of resources, start and termination of activities, queuing time, number of entities in queue, conditions for events firing and other control mechanisms. These characteristics of Petri nets accomplish the typical goals of BPR to increase service level, reduce total process cycle time and waiting time, reduce activity, resources and inventory costs, increase throughput.

Petri nets are "cost-effective" methods of exploring "what-if" scenarios quickly and finding an optimum solution to a problem because they are supported by a number of software tools that enable graphical representation of the systems by the executable models. The presented modelling methods are compared in order to show their similarities and differences. Table 1 shows the comparison of IDEF diagrams and Petri nets.

Usage criteria	IDEF diagrams	Petri nets
Simplicity	Very simple, but not available for very complex models	Fairly simple, even for complex models
Power of representation of system complexities	Not very large	Very large
Hierarchical structure	Possible	Possible
Formalism	Not existing, or very small (elaboration)	Existing strong formalism
Standardization	Existing, very strong	Lot of versions, lack of standardization
Software	Numerous	Numerous

Table 1: A comparison of IDEF diagrams and Petri nets according to their usage criteria

The most important advantages of IDEF diagrams are the simplicity and the standardization that is very important for the communication between the analysts and the users. Petri nets have following advantages: very large power of representation of system complexities and strong formalism.

Petri nets are supported by a number of software tools that enable graphical representation of the system by the executable models, such as: Alpha/Sim (ALPHATECH, Inc.), Design/CPN (MetaSoftware Corp.), MOBY (Department of Computer Science, University of Oldenburg), XsimNet (Department of Computer Science, Abo Akademi) and many others (DAIMI, 2000). There are numerous software tools for both methods, but there is also a possibility of automatic translation of Petri nets into IDEF diagrams. This possibility is widely used in business process modelling, especially in information system modelling (Pinci and Shapiro, 1991; Pinci and Shapiro, 1993). Some of the software tools for translation of Petri nets into IDEF diagrams are: Design/IDEF, Design/CPN, WorkFlow Analyser and WITNESS model (Pinci and Shapiro, 1991; Shapiro, 1994; Christensen *et al.*, 1997). IDEF3 based descriptions are used to automatically generate WITNESS simulation code in the target language using ProSim (Painter *et al.*, 1996).

IDEF diagrams and Petri nets can also be compared according to their basic elements, as shown in Table 2.

Elements	IDEF0 diagrams	IDEF3 diagrams	Petri nets
Process	Yes (connections of activity models in a network)	Yes (connections of activity models in a network)	Yes
Activity	Yes (box)	Yes (box)	Yes (transition)
Entity	Yes, implicitly (data flow through activity network)	Yes, implicitly (data flow through activity network and elaboration description)	Yes (tokens)
Resource	Yes (bottom arrow)	Yes (bottom arrow and elaboration)	Yes (tokens)
Queue	No	No	Yes (places)
Start and termination of processes	No	No	Special symbols are not used, process starts when the conditions are fulfilled
Event	No	Implicate (state transition arcs in OST network)	Yes (firing of transitions and tokens in places)
Control mechanisms	Yes (top arrow)	Yes (top arrow and logic junction)	Yes (conditions, rules, arc guards and inscriptions)
Process dynamics and behavior	No	Yes but not completely (temporal relation links and junctions)	Yes, completely

Table 2: Comparison of IDEF0, IDEF3 and Petri nets elements

According to the comparison of IDEF diagrams we can conclude that IDEF3 diagrams are more powerful method with the basic elements for simulation modelling. IDEF0 diagrams show what happens in the model (activities, entities, resources and controls), but IDEF3 diagrams show how it happens (by junctions, precedence or temporal relation links, object states and state transition arcs in Object State Transition network).

IDEF3 diagrams also capture detailed description and some elements of formalization in elaboration.

Petri nets are more powerful methods for simulation modelling because they capture all the elements important for process dynamics and system behavior presentation, like: firing conditions, entities arrival conditions, probability rules, random variables and queues.

Graphical symbols of IDEF diagrams can be translated into appropriate symbols for Petri nets (Table 3).

IDEF0 diagrams	IDEF3 diagrams	Petri nets
Action	Action	Transition
Link or arc	Link or arc	Arc
Text description of entities	Text description of entities, description in elaboration	Tokens
Resources	Resources	Tokens
Control mechanism	Control mechanism, logic junction, precedence links	Firing conditions, rules, arc inscriptions
-	Object states	Tokens in places
	State transitions arcs in OSTN diagram	Firing transitions rules
-	-	Queues

Table 3: Translation of IDEF diagrams symbols in Petri nets symbols

Activities represented by rectangles (boxes) in IDEF diagrams can be represented as transitions in Petri nets. Links in IDEF diagrams are transformed into a combination of places and arcs in Petri nets. Text description of entities participating in the process shown in IDEF diagrams, can be represented as tokens (token classes) in Petri nets.

A decomposition of an IDEF diagram can be represented as transition in Petri nets. Junction boxes in IDEF3 diagrams become transition rules in Petri nets, whilst entity states in Object State Transition Network become places in Petri nets. State transition arcs in IDEF3 diagrams become events in Petri nets. Dynamics, concurrent processes and the impact of random variables are not captured by IDEF diagrams. On the other hand, Petri nets are able to represent these concepts by arcs inscriptions and transition rules.

7 Conclusions

This paper has demonstrated the usability of IDEF diagrams and Petri Nets for modelling business processes. A comparison of these methods was also provided.

IDEF diagrams advance business process modelling by:

- enhancing the effectiveness of extracting the knowledge and information from the users,
- facilitating the presentation of business process model to the users in order to get their validation and evaluation.

Due to their simplicity and understandability, it seems appropriate to develop IDEF diagrams during preliminary phases of business process modelling project in order to develop "AS IS" models. In later phases, when "TO BE" models are developed, IDEF diagrams could be simply transformed into Petri nets which adds formal semantic to the models. Simulating the effects of redesigned processes before implementation improves the chances of getting the processes right at the first attempt. The advantages of simulation modelling were demonstrated on the example of a retail system process model using Petri nets (DES-nets):

- Petri nets are fairly simple and easy to learn and use since they use a limited number of symbols, but have large power of representation of system complexities
- as a graphical technique, they enable the visualisation of the system being modelled,
- as a mathematical tool Petri nets can be used as analytic technique and can be applied to small models or submodels,
- Petri nets can handle concurrent, parallel, or asynchronous activities (the inability to handle these system complexities is the main disadvantage of many other simulation methodologies).

Multiple methods exist for BPR, but usually several different methods are used to perform more useful and efficient process modelling. Such an approach can support BPR projects and increase the chance for their success. This research reveals that IDEF diagrams and Petri Nets complement each other and that they should be used simultaneously for business processes modelling. The more extensive use of business process modelling could reduce the risks of changes. This should result in increase in the rate of success of BPR and other change management related projects.

8 References

[1] Bhaskar, R., Lee, H.S., Levas, A., Petrakian, R., Tsai, F. and Tulskie, B., Analysing and Reengineering Business Processes Using Simulation, (J.D. Tew, S. Manivannan, D.A. Sadowski and A.F. Seila, eds.) *Proceedings of the 1994 Winter Simulation Conference*, Lake Buena Vista, Florida, USA, 1206-1213 (1994).

[2] Ceric, V, Petri Nets for Discrete Event Simulation, (F.Breitenecker and I. Husinsky, eds*), Eurosim 95, Simulation Congress Proceedings*, Vienna, Austria, 1169-1174 (1995).

[3] Ceric, V. and Paul, R., Diagrammatic Representations of the Conceptual Model for Discrete Event Systems, *Mathematic and Computers in Simulation* No. 34, 317-324 (1992).

[4] Ceric, V., Diagrammatic Modeling for Discret Event Simulation, *book in preparation* (2000).

[5] Christensen, S., Jřrgensen, J.B., Kristensen, L.M.: Design/CPN - A Computer Tool for Coloured Petri Nets, (E. Brinksma, ed.) *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'97, Lecture Notes in Computer Science*, vol. 1217, Springer-Verlag, 209-223 (1997).

[6] Davenport T.H. and Short J.E., The new industrial engineering: information technology and business process redesign, *Sloan Management Review* 11-27 (1990).

[7] David, R. and Alla, H., Petri Nets for Modeling of Dynamic Systems - A Survey, *Automatica* 30, 175-202 (1994).

[8] Department of Computer Science at the University of Denmark (DAIMI), Petri Net Tools, http://www.daimi.au.dk/petrinet/ (2000).

[9] deWitte, P. and Pourteau, C., IDEF enterprise engineering methodologies support simulation, *Information Technology for Manufacturing Managers*, March 1997, 70-75 (1997).

[10] Dietz, J.L. and Barjis, J., Petri Net Expressions of DEMO Process Models as a Rigid Foundation for Requirements Engineering, (B.Sharp, J.Cordeiro and J.Filpe, eds.) *Proceedings of 2nd Conference on Enterprise Information Systems*, Stafford, UK, 267-274 (2000).

[11] Gladwin, B. and Tumay, K., Modeling Business Processes with Simulation Tools, (J.D. Tew, S. Manivannan, D.A. Sadowski and A.F. Seila, eds.) *Proceedings of the 1994 Winter Simulation Conference*, Lake Buena Vista, Florida, USA, 1177-1184 (1994).

[12] Hammer, M. and Champy, J., *Reengineering the corporation*, NY: Harper Collins Books, New York (1993).

[13] Harrison B.D. and Pratt M.D., A methodology for reengineering businesses, *Planning Review* 21(2) 6-11 (1993).

[14] Vlatka Hlupic and Stewart Robinson, Business Process Modelling Using Discrete-event Simulation, Proceedings of the *Winter Simulation Conference WSC'98*, Washington DC, USA, December 1998, (Ed. by Madeiros D.J., Watson E.F., Karson J.S. and Manivannan M.S.), SCS, 1998, pp.1363-1369.

[15] Kamper, S., Object-Oriented Model Building with Petri Nets Using the Simulation Tool NET, *Proceedings of the 3rd European Simulation Congress*, Edinburgh, 142-147 (1989).

[16] Kettinger, W.J., Teng J.T.C. and Guha S. (1997), Business process change: a study of methodologies, techniques, and tools, *MISQ Quarterly* March 55-80 (1997).

[17] Kovacic, A., Vintar M., Reforming the Public Sector in Slovenia: Re-Engineering Administrative Units, Public and Private Sector Partnerships: Fostering Enterprise, *Proceedings of the Fourth International Conference*, Sheffield Hallam University Press (1998).

[18] Marca, D.A. and McGowan, C.L., *SADT*, McGraw-Hill, New York (1988).

[19] Mayer, R.J., Benjamin, P.C., Carawaz, B.E. and Painter, M.K., A Framework and a Suite of Methods for Business Process Reengineering, http://www.idef.com/articles/framework/

[20] (1998).Oakshot, L, *Business Modelling and Simulation*: Pitman Publishing, London, UK (1997).

[21] Oberweis, A. and Sänger, V., Evolutionary System Development: An Approach Based on Petri Net Simulation, (J. Stephenson, ed.) *Proceedings of the European Simulation Multiconference - ESM'92*, York, 1992., 172-176 (1992).

[22] Painter, M.K., Fernandes, R., Padmanaban, N. and Mayer, R.J., A Methodology for Integrating Business Process and Information Infrastructure Models, *Proceedings of the 1996 Winter Simulation Conference*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner and J.J. Swain, 1305-1312 (1996).

[23] Paul R.J., Hlupic V. and Giaglis G., Simulation modelling of business processes, (Avison D. and Edgar-Neville D., eds.), *Proceedings of the 3rd UK Academy of Information Systems Conference*, June 1998 (McGraw-Hill), Lincoln, UK, 311-320 (1998).

[24] Pinci, O. and Shapiro, R.M., An Integrated Software Development Methodology Based on Hierarchical Colored Petri Net, (G. Rozenberg, ed.) *Advances in Petri Nets, 1991, Lecture Notes in Computer Science*, Springer Verlag (1991).

[25] Pinci, O. and Shapiro, R.M., Work Flow Analysis, (G.W. Evans, M. Mollaghasemi, EC. Russell and W.E. Biles, eds.), *Proceedings of the 1993 Winter Simulation Conference*, Los Angeles, California, USA, 1122-1130 (1993).

[26] Pooley, R.J., Towards a Standard for Hierarchical Process Oriented Discrete Event Simulation Diagrams, Part 1: A Comparison of Existing Approaches, *Transactions of The Society for Computer Simulation* No. 8, 1-20 (1991).

[27] Shapiro, R.M., Integrating BPR with Image-Based Work-Flow, (J.D. Tew, S. Manivannan, D.A. Sadowski and A.F. Seila, eds.) *Proceedings of the 1994 Winter Simulation Conference*, Lake Buena Vista, Florida, USA, 1221-1227 (1994).

[28] Tatsiopoulos, I.P., Leopoulos, V.I.N., Panayiotou, N.A. and Tsikonis, L., A Comparative Simulation Analysis in the Customer Offer Process Using Petri Nest and Discrete Event Simulation, (G. Horton, D. Moller and U. Rude, eds.) *Proceedings of 11th European Simulation Symposium*, 61-65 (1999).

[29] Thomas, C., Hierarchical Object Nets - a Methodology for Graphical Modeling of Discrete Event Systems, (G.W. Evans, M. Mollaghasemi, EC. Russell and W.E. Biles, eds.), *Proceedings of the 1993 Winter Simulation Conference*, Los Angeles, California, USA, 650-656 (1993).

[30] Törn, A., Simulation nets, a simulation modeling and validation tool, *Simulation*, Vol. 45, No. 2, 71-75 (1985).

[31] Tumay, K., Business process simulation, in: Alexopoulos A., Kang K., Lilegdon W.R. and Goldsman D., eds., *Proceedings of the 1995 Winter Simulation Conference, Washington DC, USA* (1995) 55-60.

[32] Van Ackere, A., Larsen, E.R. and Morecroft, J.D.W., Systems Thinking and Business Process Redesign: an Application to the Bear Game, *European Management Journal* 11, 412-423 (1993).

[33] Wastell, G.W., White P. and Kawalek P., A methodology for business redesign: experience and issues, *Journal of Strategic Information Systems* 3(1) 5-22 (1994).

[34] Withers, B.D., Pritsker, A.A. and Withers, D.H., A Structured Definition of the Modeling Process, *Proceedings of the 1993 Winter Simulation Conference*, (Ed. by G.W. Evans, M. Mollaghasemi, E.C. Rusell and W.E. Biles), 1109-1117 (1993).

[35] Yao, Y., A Petri Net Model for Temporal Knowledge Representation and Reasoning, *IEEE Transactions of Systems, Man and Cybernetics*, Vol. 24, No. 9, 1374-1382 (1994).

The object model of splines

Mojca Indihar Štemberger and Janez Grad
University of Ljubljana,
Faculty of Economics,
Kardeljeva ploščad 17
SI-1000 Ljubljana,
Slovenia
Tel: +386 61 1892 400, Fax: +386 61 1892 698
E-mail: mojca.stemberger@uni-lj.si, janez.grad@uni-lj.si

Keywords: splines, object-oriented modeling, object-oriented database, UML, ODMG, Java

Received: June 8, 2000

Data analysis like Data Mining is getting more and more important every day because the organizations want to get their competitive advantage. Splines are piecewise polynomial functions that are used for the analysis by some Data Mining tools. Except that, they are widely used in CAGD applications for designing. The article presents the object model of splines that is independent of the concrete application but can serve as the basis for any application which uses splines and reduces the gap among such applications and their incompatibility. It was modeled by object-oriented modeling language UML. We also propose how to save splines in an ODMG compliant database. The prototype of a system was developed in Java and is presented in the article as well.

1 Introduction

Nowdays every organization has to consider data as an important resource if they want to be successful in every day more competitive world. This is the main reason for the growing use of data analysis tools such as OLAP (On-Line Analytical Processing) or Data Mining (Koutsoukis, Mitra, Lucas, 1999). One of the methods that Data Mining tools use for discovering connection among large amounts of data is the approximation or the regression. In this case the user gets the connection among the variables in the form of mathematical functions and the prediction of values of input variables in the points where they aren't known. The development in the field of mathematics and statistics proved that piecewise polynomial functions called splines served the best for the purpose of approximation or regression (De Boor, 1978), (Friedman, 1991), (Eubank, 1999), (Pagan, Ullah, 1999). The example of Data Mining tool that uses splines is called MARS (Salford Systems) – Multivariate Adaptive Regression Splines.

Another area where splines are widely used is Computer Aided Geometric Design (CAGD) where splines are used for designing. Designing with splines began in France and USA in 1960s because of the needs in car and aviation industry (Farin, 1997).

Splines are briefly introduced in section 2. The reader who is not familiar with the terms can read more about the splines in the extensive literature that is referred to at the References.

In the field of using splines, the problem lies in the gap among different applications that use splines. Since they do not use the common object model, their compatibility may be difficult. Our proposed solution is the use of the object model that is independent of the concrete application, DBMS and programming language, and can serve as the basis for each application that use splines. The object model of the splines was developed by the authors with the well-known Unified Modeling Language - UML (Booch, Rumbaugh, Jacobson, 1999) and is described in section 3.

The article also describes a solution for storing splines in a way that guarantees the maximum possible portability among different applications that use splines (section 4). The solution is based on the common object model. ODMG standard (Cattell et al., 2000) is used for the purpose of storing the produced objects. The prototype of such application was developed in the promising programming language Java as well and is described in section 5.

2 Splines

Polynomials have been widely used for the interpolation and approximation because it is rather simple to calculate their coefficients. But by increasing the number of points to be interpolated or approximated it is not always possible to find satisfying interpolation or approximation within the class of polynomials. Polynomials of a higher degree can have oscillations (Schumaker, 1993), which means that some other class of functions had to be introduced. Splines are such class of functions. They are piecewise polynomial functions that are smooth in the joint points. Spline can be expressed as sequence of polynomial function, one for each polynomial segment, or as a special form of spline like

B-spline or Bézier curve (de Boor, 1978), (Farin, 1997), (Schumaker, 1993).

The splines used by Data Mining tools usually belong to the first group, since in most cases they are expressed as a sequence of polynomials that are smooth in the joint points. Figure 1 shows an example of the cubic spline that consists of two polynomial segments and is smooth in the joint point $x = 1$:

$$y = \begin{cases} x^3; & x \le 1 \\ 2x^3 - 3x^2 + 3x - 1; & x > 1 \end{cases}$$

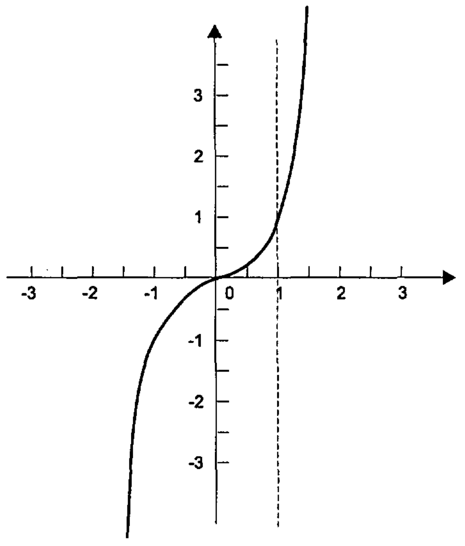


Figure 1: Cubic spline

B-splines are used for the special representation of polynomial splines that have additional advantages such as better numerical stability (Schumaker, 1993). They can be defined in several ways. Probably the easiest way to define them is the recursive formula

$$\begin{aligned} B_{i,1}(x) &= \begin{cases} 1; & t_i \le x < t_{i+1} \\ 0; & \text{otherwise} \end{cases} \\ B_{i,k}(x) &= \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \\ &+ \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x), \end{aligned}$$

where $\mathbf{t} = (t_i)_{i=0}^{m+k}$ is the no decreasing sequence of real numbers called the vertices and

$$\frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x)$$

and

$$\frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x)$$

is considered as 0 when $t_{i+k-1} - t_i = 0$ or $t_{i+k} - t_{i+1} = 0$.

B-spline of order k $B_{i,k}(x)$ is a polynomial of order k that has an important property:

$$\begin{aligned} B_{i,k}(x) &\ge 0, \text{ when } x \in [t_i, t_{i+k}], \\ B_{i,k}(x) &> 0, \text{ when } x \in (t_i, t_{i+k}), \\ B_{i,k}(x) &= 0, \text{ when } x \notin [t_i, t_{i+k}]. \end{aligned}$$

The reason for introducing B-splines is the definition of B-spline function and B-spline curve. The first one is mostly used for the approximation or interpolation and the second for designing.

B-spline function of order k with the sequence of vertices $\mathbf{t} = (t_i)_{i=0}^{m+k}$ is every linear combination of B-splines of order k, that is

$$f(x) = \sum_{i=0}^{m} \alpha_i B_{i,k}(x),$$

where α_i, $i = 0, 1, \ldots, m$ are real numbers. Several algorithms for interpolation or approximation with B-spline functions are known (De Boor, 1978), (Eubank, 1999).

B-spline curve is expressed parametrically with (Farin, 1997)

- B-splines of order k,
- the sequence of vertices $\mathbf{t} = (t_i)_{i=0}^{m+k}$ and
- the control points $V_0, V_1, \ldots, V_m$, where each point has x and y koordinate:

$$V_i = (x_i, y_i).$$

Figure 2 shows the cubic B-spline curve (order 4) with control points V_0, V_1, V_2, V_3, V_4, $V_5 = V_0$ and vertices $t_0 = -1$, $t_1 = 0$, $t_2 = 1$, $t_3 = 2$, $t_4 = 3$, $t_5 = 4$, $t_6 = 5$, $t_7 = 6$, $t_8 = 7$, $t_9 = 8$. The control polygon is the broken line that connects the control points. It is drawn in the figure too.

B-spline curve actually consists of two B-spline functions, one for each coordinate. By adding another coordinate we get B-spline surfaces that are also used for designing.

It is important to notice that B-splines, B-spline functions and B-spline curves are very convenient to be processed by computers since very little data have to be stored for their reproduction. One of their main advantages is that changing one segment only makes influence on the segments that are close to it.

Bézier curves are most widely applied in CAGD tools but they are actually the special form of B-spline curves. They were introduced in 1960s by Bézier and De Casteljau but they are named after Bézier while De Casteljau's work has not been published. They are defined as

$$\mathcal{B}_n[V_0, V_1, \ldots, V_n; a, b; t] = \sum_{i=0}^{n} V_i B_i^n(t), \ a \le t \le b,$$

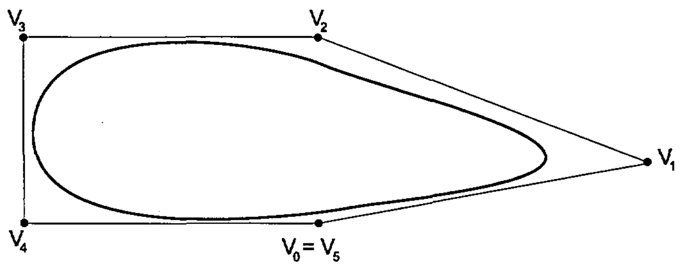


Figure 2: B-spline curve and its control polygon

where

$$B_i^n(t) = \binom{n}{i}\frac{(t-a)^i(b-t)^{n-i}}{(b-a)^n}, \qquad a \le t \le b, \quad i = 0, 1, \ldots, n$$

are Bernsteins polynomials of degree n and points $V_0, V_1, \ldots, V_n$ are control points.

Designing with B-spline curves or Bézier curves (Farin, 1997) is very easy. The control polygon mimics the curve since the curve is close to the polygon. If a control point is moved the curve segments close to it move too. Every operation such as rotating, moving or changing the size of the curve can be done by performing it on the control polygon and by redrawing the curve. In case a desired designing could not be done by moving existing control points, additional control points can be introduced. This process is called subdivision (Farin, 1997).

There are several other classes of spline functions that are described in (Farin, 1997) and (Schumaker, 1993). Other interesting properties of splines are described there as well.

3 The object model of the system

Standard language for object-oriented modeling UML has been used for the development of the model, since it is possible to map from a model in the UML to an object-oriented programming language such as Java or C++, or even to tables in a relational database or the persistent store of an object-oriented database (Booch, Rumbaugh, Jacobson, 1999). The developed model which is described bellow can serve as the basis for any application that uses splines because it is independent of the programming language and the application. Only the parts of the system that are independent of the concrete application have been modeled, because any object model can be easily extended with the parts that are specific for the application. The model is divided into four packages. If an application needs only a specific kind of splines then only one or more packages have to be employed. It is important to point out that in the object model not only data but also the operations on splines are modeled.

We used several diagrams for modeling the system. The behavior of the system was modeled with the use case diagram and some sequence diagrams. Figure 3 shows the use case diagram of the system. As it is evident from the figure we have two kinds of users (`Researcher` and `Designer`) that are presented as actors. The first one is in interaction with the use case `Approximation with splines` and `Interpolation with splines`, but the second one is in interaction with the use case `Designing with splines`. All the mentioned use cases include the use case `Saving spline` and the third one includes `Changing shape of spline` and `Create spline`.

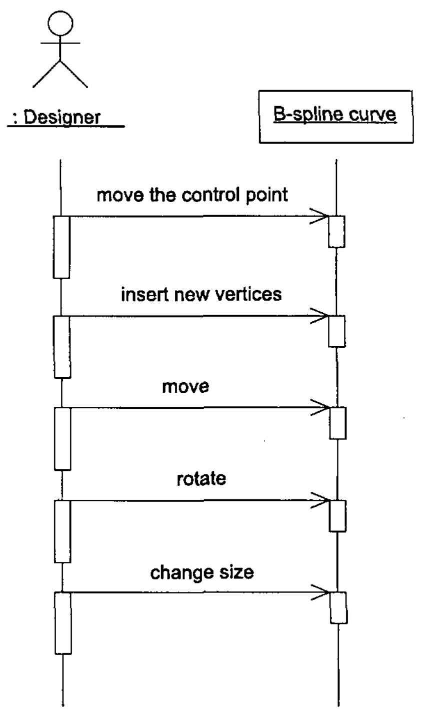


Figure 4: Sequence diagram for B-spline curve

We have modeled use case `Changing shape of spline` with sequence diagram. It is shown in Figure 4 for the B-spline curve. Similar diagram can be drawn for the Bézier curve.

The structure of the system was modeled with class diagrams. A type of certain parameters has been chosen but the rest are left to the phase of more detailed modeling or

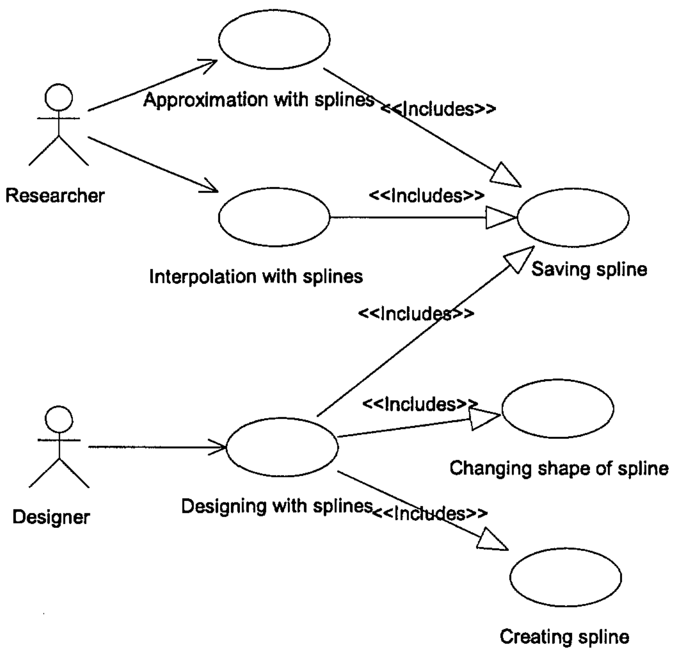


Figure 3: Use case diagram of the system

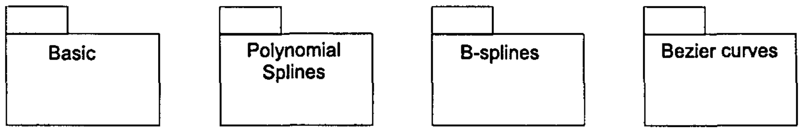


Figure 5: Main class diagram of the system

implementation. The messages captured in sequence diagrams are very important since they helped us to determine the operations of the classes. Figure 5 shows the main class diagram of the system that is divided into four packages.

Package `Basic` captures classes that form the basis for all other classes such as `Polynomial`, `Interval` and `Point`. As evident from Figure 6, which shows the main class diagram of the package, all the modeled classes have constructor `create()` and destructor `delete()`. Classes have also other attributes and operations that are self-describing. Classes `LinearFunction`, `QuadraticFunction` and `CubicFunction` are modeled specially because they represent the most widely used classes of functions for the approximation or interpolation. They inherit all attributes and operations from class `Polynomial` but some of them are overridden since they can be implemented differently.

The package called `Polynomial splines` consists of the classes needed to represent piecewise polynomial functions or polynomial splines. The central class of the package is called `PiecewisePolynomialFunction` but we have also modeled some other classes that all inherit attributes and operations from this class. The class diagram and details about attributes and operations are evident from Figure 7. All classes in the package are collections of polynomials where collection can be a list, an array or of some other type. The best way of implementing operations is to call the methods of the corresponding polynomial segments. Attribute `continuityAtJointPoints` represents a degree of continuity at every joint point (-1 stands for uncontinues function, 0 for continues, 1 for first derivative being continues, etc.). Other classes in the package inherit everything from the main class but some attributes are different. Similar as for polynomials some operations can be implemented differently. In case one's objective is to implement some special algorithms for approximation or interpolation with splines, one can simply introduce a new class to the system that inherits attributes and operations from modeled classes and have some additional attributes and operations.

Package `B-splines` consists of classes that model B-splines, B-spline functions and B-spline curves. The relationships of the classes are shown in Figure 8. The meaning of most attributes and operations is obvious but some of them need comments. Many operations of the class `B-Curve` are derived from translating the messages from the sequence diagram to corresponding operations. Operation `moveControlPoint()`, for example, serves to moving some control point of the curve to some other lo-

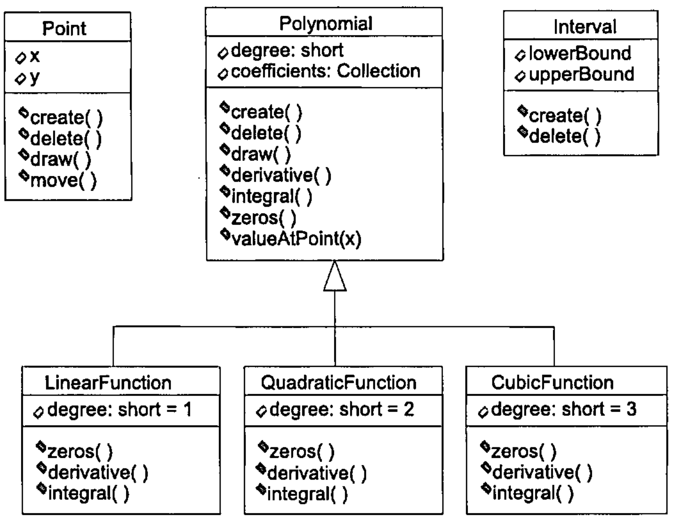


Figure 6: Class diagram of the package `Basic`

cation. In that case, a part of the curve has to be redrawn. The classes are connected with the relationship aggregation. New classes that inherit from the modeled classes can be simply added to the package.

Package `Bezier curves` is very similar to the package `B-splines` and is described in (Indihar Štemberger, 2000), where many other details about the model are also described.

It is very important to understand that the described model is independent of the purpose of using splines. It can serve as the basis of the system where splines are used for approximation or interpolation and of the system for designing with splines as well, because it is a common dominator of every such system.

4 Storing splines

In modern information systems we have to deal with complex objects. Classical relational DBMS are not the appropriate solution for storing such objects. The development in this field has gone further with the concepts of object or object-relational databases. The concept of the database has been extended to the point that it includes the execution of the processes as well (Domanjko, Heričko, Rozman, 1997). Therefore ono does not need to distinguish between applications and databases anymore.

The idea is to store splines in object-oriented database. Namely, they have already been used for storing splines produced by CAD applications (Cattell, 1994), but not in connection with Data Mining applications.

The fact is that there is no general agreement in the literature (Cattell, 1994), (Kim, 1995), (Khoshafian, 1995) about the definition of the object-oriented database. We understand the object-oriented database as the combination of object-orientation and database capabilities. It implies that the so-called "true" object-oriented databases and object-relational databases both belong to that category.

The ODMG standard (Cattell et al., 2000) was used to store splines. The standard is independent of the programming language and the DBMS. It can be used for storing objects in "true" object-oriented database and with special mappings, which map objects into relations, to the relational database as well.

By the ODMG standard the schema of object-oriented database can be written in one of the programming languages that have the ODMG mapping (C++, Smalltalk and Java) or in independent language ODL (Object Definition Language). Class diagrams from section 3 can be converted to ODL. Figure 9 shows the example of such schema for the classes from Figure 7.

The use of ODMG standard ensures the portability among different supported programming languages and also among different DBMS.

5 Prototype

The prototype of the system has been developed as well. We used Java programming language and the Poet ODBMS. The user interface of the prototype is presented in Figure 10. A user can enter the points he wants to approximate by a cubic spline simply by clicking the mouse. The approximation is obtained by algorithm described in (Eubank, 1999). The produced cubic spline corresponds to the class `PiecewiseCubicFunction` in the model. Actually it is the ancestor of that class since it has additional operation called `approximate()` which serves to obtaining the approximation to the given data. The spline can be stored in the database and the stored splines can be retrieved from the database.

One has to mention again that ODMG standard can be used for "true" object-oriented DBMS like the one we used

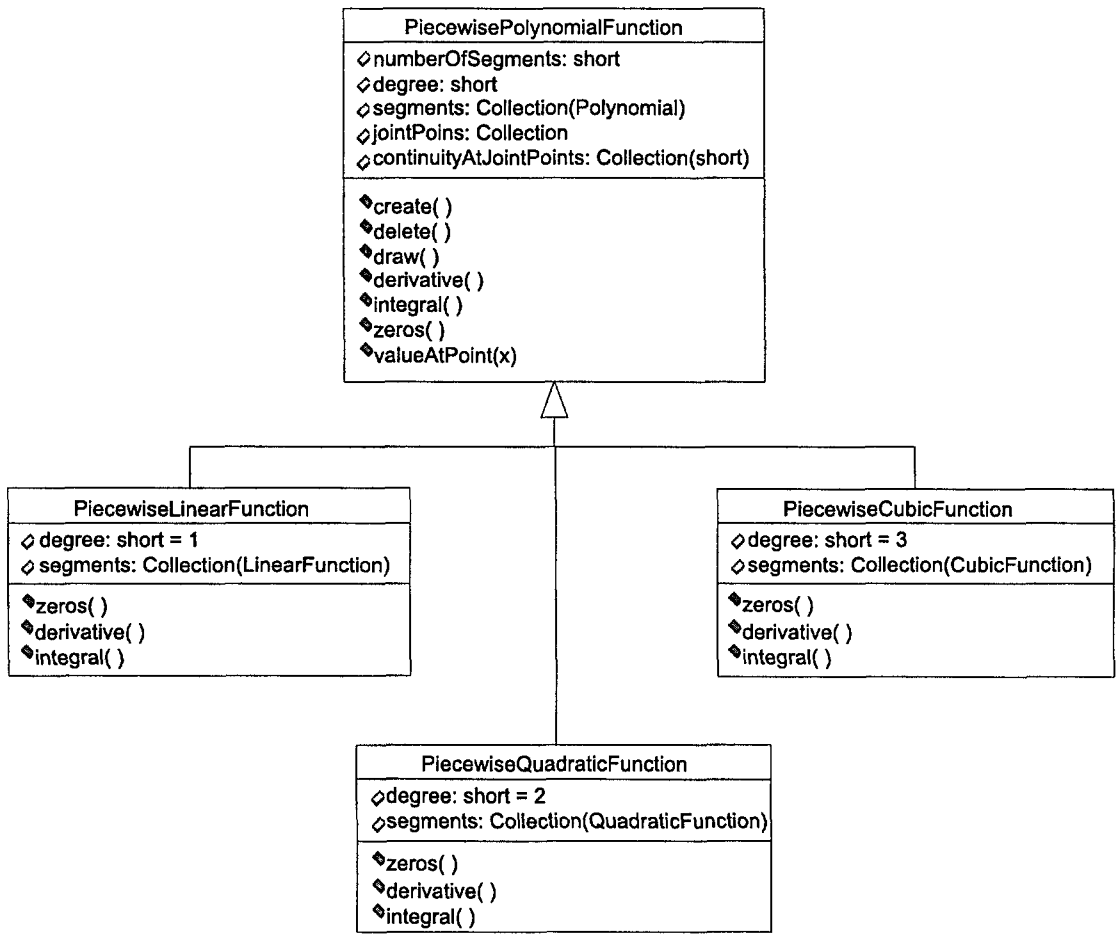


Figure 7: Class diagram of the package `Polynomial splines`

and also with the special mapping (Java Blend for example) for the relational DBMS.

6 Conclusion

We believe that the presented model can serve as a common dominator of a system that uses splines. Each application of that kind can be based on it and extended with additional classes. We are shure that the choice of UML modeling languege was correct since the variety of the environments which are suitable for the implementation of the UML models is very reach. The use of the standard ODMG, that has been developed by the major vendors of object-oriented database management systems and is the only existing standard in this field (Alagić, 1999), for describing the model of saving splines guarantees the maximum possible portability of the system. Produced objects can be stored in any DBMS that supports the standard. It is important to add that not only attributes but also the methods of the classes are persistent. The prototype has been developed with the purpose of expressing our ideas. Our intention was not developing Data Mining tool but in our opinion vendors of such tools should consider these ideas.

References

[1] Alagić Suad: A Family of the ODMG Object Models, Third East European Conference ADBIS'99 (Lecture Notes in Computer Science, Vol 1691), Berlin et al.: Springer, 1999, pp. 14-30.

[2] Booch Grady, Rumbaugh James, Jacobson Ivar: The Unified Modeling Language User Guide. Reading [etc.]: Addison Wesley Longman, 1999.

[3] Cattell Rick G. G.: Object Data Management. Reading: Addison-Wesley Publishing Company, 1994.

[4] Cattell Rick G. G., Barry Douglas, Bartels Dirk, Berler Mark, Eastman Jeff, Gamerman Sophie, Jordan David, Springer Adam, Strickland Henry, Wade Drew: The Object Database Standard ODMG 3.0. San Francisco: Morgan Kaufmann Publishers, 2000.

[5] De Boor Carl: A practical Guide to Splines. New York: Springer-Verlag, 1978.

[6] Data Mining and Knowledge Discovery in Databases, published by UNICOM Seminars, 1998.

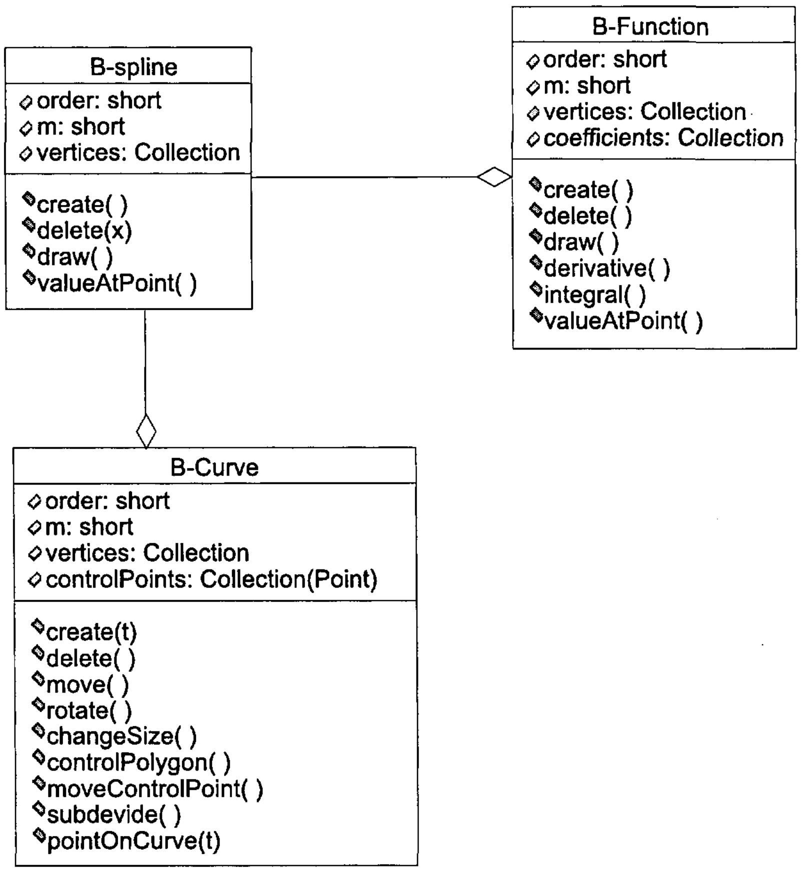


Figure 8: Class diagram of the package `B-splines`

[7] Domanjko Tomaž, Heričko Marjan, Rozman Ivan: The usage of object-oriented databases, COTL, 1 (1997), 2, in Slovene. [URL: http://lisa.uni-mb.si/cot/cotl/april97/index.shtml].

[8] Eubank Randall L.: Nonparametric Regression and Spline Smoothing (Statistics, Textbooks and Monographs, V. 157), Marcel Dekker, 1999.

[9] Farin Gerald: Curves and Surfaces for CAGD. Boston: Academic Press, 1997.

[10] Friedman J.H.: Multivariate Adaptive Regression Splines. Annals of Statistics, 19 (1991), pp. 1-141.

[11] Indihar Štemberger Mojca: Splines in object-oriented database environment. Ph.D. Thesis, Ljubljana: University of Ljubljana, Faculty of Economics, 2000.

[12] Kim Won, editor: Modern Database systems – The Object Model, Interoperability and Beyond. New York: Addison-Wesley Publishing Company, 1995.

[13] Khoshafian Setrag, Abnous Razmik: Object Orientation. New York: John Wiley & Sons, 1995.

[14] Koutsoukis N.S., Mitra G., Lucas C.: Adapting Online Analytical Processing for Decision Support: The interaction of information and decision technologies, Decision Support Systems, 26 (1999), pp. 1-30.

[15] Pagan Adrian, Ullah Aman: Nonparametric Econometrics, Cambridge: Cambridge University Press, 1999.

[16] Poet, [URL: http://www.poet.com/].

[17] Salford Systems, [URL: http://www.salford-systems.com/].

[18] Schumaker Larry L.: Spline Functions: Basic Theory. Malabar, Florida: Krieger Publishing Company, 1993.

```
class PiecewisePolynomialFunction
{
    attribute short degree;
    attribute short numberOfSegments;
    attribute Array<Polynomial> segments;
    attribute array<float> jointPoints;
    void create();
    void delete() raises (no_function);
    void draw();
    float valueAtPoint(in float x) raises(not_defined);
    bag<float> zeros();
    PiecewisePolynomialFunction derivative();
    PiecewisePolynomialFunction integral();
};

class PiecewiseLinearFunction extends
    PiecewisePolynomialFunction
{
    attribute Array<LinearFunkction> segments;
    bag<float> zeros();
    array<float> derivative();
    PiecewiseQuadraticFunction integral();
};

class PiecewiseQuadraticFunction extends
    PiecewisePolynomialFunction
{
    attribute Array<QuadraticFunction> segments;
    bag<float> zeros();
    PiecewiseLinearFunction derivative();
    PiecewiseCubicFunction integral();
};

class PiecewiseCubicFunction extends
    PiecewisePolynomialFunction
{
    attribute Array<CubicFunction> segments;
    bag<float> zeros();
    PiecewiseQuadraticFunction derivative();
    PiecewisePolynomialFunction integral();
};
```

Figure 9: ODL definitions for some classes

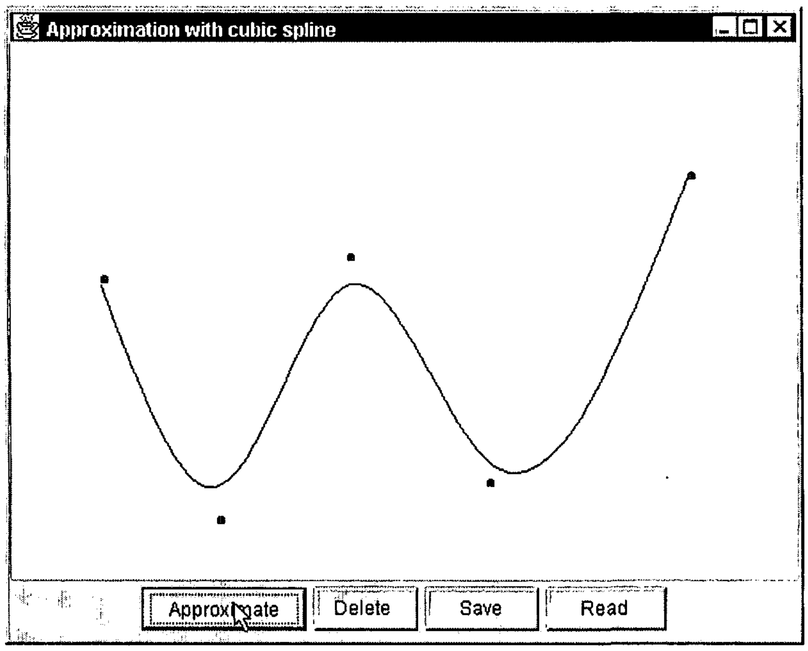


Figure 10: The user interface of the prototype

Call for a forum discussing informational consciousness on the Internet

Informational Consciousness (IC) is meant to be an artificial mind system embedded into the body of the World-Wide Web (WWW). The place of discussion is opened at

$\boxed{\text{http://groups.yahoo.com/group/artifico}}$

with a link to the study (book, written in an essay form) [1], entitled

Introduction to Artificial Consciousness

This study is a research in progress presented by a huge volume of more than 300 pages in LaTeX2ε format in Letter size.

The study [1] is a preliminary draft aimed for discussion by everyone interested in problems of consciousness identification, formalization and, finally, implementation by the so-called informational machine, discussed in the study. The reader and critical discusser should consider that the study is

- not completed yet,
- not standardized consequently by a unique symbolism through the entire text,
- not finally corrected in logical and conceptual errors, and
- not finally corrected in English.

The author works daily on improving the details and in elaborating the sections of the study not being written to the end or existing as section titles only. The initial half of the study is now under the procedure of the strict formal and verbal standardization. The author invites critical readers and discussers to join the Artifico Forum at the Yahoo! Site, to sign in by name and password, and to take the advantage of reading and interactive discussion. In this way, a consciousness researcher will profit by putting questions to the author and other participants via the Forum or directly, using the listed individual e-mail addresses.

What can be seen on the reader's screen viewing the file artifico.PS? At the beginning of the study, a short abstract with keywords is given. In the preface and acknowledgment, a history of the study emergence and author's critical life and intellectual influence of the environment is presented. Within the contents, set on 15 pages, 33 sections are listed, following by a list of figures (60) and tables (13).

The sections deal with informational constitution of consciousness (page #1), supervenience (3), operands (14), operators (20), formulas (23), formula systems (37), primitive formula systems (41), gestalts (44), schemes (65), measures (75), frames (85), formula graphs (89), graphs of formula systems (92), experiments (98), axioms (115), primitive transition (144), circularism and rotationalism (145), general decomposition (147), metaphysicalistic decomposition (182), multiple decomposition (210), decomposition by modi informationis (217), metadecomposition (219), informational shells (210), consciousness as informational phenomenalism (222), premetaphysicalistic concepts (227), shells of consciousness (228), informational communication (240), meaning and understanding (248), the nothing, the all, and informational emergence (249), informational machine (259), conclusion (269), list of 102 references (264), and index (267–280).

As mentioned before, some of sections and subsections are titled only and not written yet. The decision to put the study into the public domain grounds in the author's belief that an interactive discussion and critic of the written matter can bring the subject of informational consciousness to other philosophical and scientific disciplines, for instance, to show what a contemporary philosophy of the informational and its mathematical formalization in a new informationally emergent way could become in the future. The author believes that a number of original concepts of formalization, methodology, and informational organization could enrich a wide field of humanistic and engineering disciplines.

When joining the Artifico Group, by signing in, the participant will be advised to look at

$\boxed{\text{http://lea.hamradio.si/\~{}s51em/}}$

for the possibility of loading the file $\boxed{\text{artifico.PS}}$, and for the technical conditions of its decompression, reading on the screen and printing. The viewing and printing tool can be loaded from the advised Internet site. Everything in this loading operation is free of charge. If someone has already installed LaTeX system, the reading and printing by GsView is guaranteed immediately.

References

[1] ŽELEZNIKAR, A.P. Introduction to Artificial Consciousness. An Informational Approach, Formalization, and Implementation. 2001. Currently on 303 pages of Letter size. Available (can be loaded automatically) from

http://lea.hamradio.si/~s51em/ or
http://www.artifico.org

When loaded and decompressed, the file artifico.PS is readable and printable arbitrarily from page to page or entirely by the use of GsView, available free from the designer.

Anton P. Železnikar

INFORMATION SOCIETY 2001
INFOS, Cankarjev dom, Ljubljana, Slovenia
22.-26. October 2001

You are kindly invited to cooperate on multi-conference Information Society – IS 2001, which will be held under INFOS from 22nd to 26th of October 2001 in Cankarjev dom in Ljubljana. The multi-conference will include important achievements on the fields mentioned below. Emphasis will be given on the exchange of ideas and particular suggestions, which will be included in the final paper of individual conferences.

IS 2001 exists of nine carefully chosen conferences:

Collaboration and information society
Data mining and warehouses
Development and reingeeniring of information systems
Education in information society
Intelligent systems
Management and information society
Medical and cognitive science
Speech technologies
New information technologies in fine arts

Further information is available at http://is.ijs.si/ or http://ai.ijs.si/is/is2001/index01.html.

Institutions, enterprises and donators are invited to present interesting new developments on their fields of work as 'normal' contributions. They can make a review of new developments and existing situation in their institutions and talk about problems of development in Slovenia, attitude of governmental institutions, and about the way Slovenia should be developing in the direction of information society. They can grant certain interesting activities, related to their work (please turn to the organizator, for example matjaz.gams@ijs.si).
The emphasis is on development, new ideas and trends in information society. If you have something interesting to tell or show to Slovenia, Information Society is the right place to be.

Invited are primarily all those, who have some knowledge about information society. Presentations of enterprises are welcome, especially from the functional point of view. To summarize, we will meet to tell what can we do in Slovenia, to exchange our experiences and to help Slovenia make a step forward in the direction of information society.

You are kindly invited to make a presentation and actively take part in the open exchange of ideas with your knowledge and achievements. The submission deadline is fall 2001.

Pictures from the IS 2000 conference can be found at http://ai.ijs.si/IS/is2000/index00.html.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slove**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in .eps format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than five years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Informatica WWW:

http://ai.ijs.si/informatica/
http://orca.st.usm.edu/informatica/

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editórial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 25 Number 1 April 2001 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics