The 18th European Conference on Artificial Intelligence

Proceedings

1st International Workshop on Combinations of Intelligent Methods and Applications (CIMA 2008)

Tuesday July 22, 2008
Patras, Greece

Ioannis Hatzilygeroudis, Constantinos Koutsojannis and Vasile Palade

Table of Contents

Workshop Organization

Chairs-Organizers

Ioannis Hatzilygeroudis
University of Patras, Greece

Constantinos Koutsojannis
TEI of Patras, Greece

Vasile Palade
Oxford University, UK

Program Committee

Ajaith Abraham, IITA, South Korea

Ao Sio Iong, Oxford University, UK

Plamen Agelov, Lancaster University, UK

Emilio Corchado, University of Burgos, Spain

George Dounias, University of the Aegean, Greece

Artur S. d'Avila Garcez, City University, UK

Melanie Hilario, CUI - University of Geneva, Switzerland

Elpida Keravnou-Papailiou, University of Cyprus, Cyprus

Rudolf Kruse, University of Magdeburg, Germany

George Magoulas, Birkbeck College, Univ. of London, UK

Vasilis Megalooikonomou, University of Patras, Greece

Toni Moreno, University Rovira i Virgili, Spain

Amedeo Napoli, CNRS-INRIA-University of Nancy, France

Ciprian-Daniel Neagu, University of Bradford, UK

Jim Prentzas, TEI of Lamia, Greece

Han Reichgelt, Southern Polytechnic State Univ., GA, USA

David Sanchez, University Rovira i Virgili, Spain

Douglas Vieira, University of Minas Gerais, Brazil

Contact Chair

Ioannis Hatzilygeroudis
Dept. of Computer Engineering & Informatics
University of Patras, Greece
Email: ihatz@ceid.upatras.gr

Preface

The combination of different intelligent methods is a very active research area in Artificial Intelligence (AI). The aim is to create integrated or hybrid methods that benefit from each of their components. It is generally believed that complex problems can be easier solved with such integrated or hybrid methods.

Some of the existing efforts combine what are called soft computing methods (fuzzy logic, neural networks and genetic algorithms) either among themselves or with more traditional AI methods such as logic and rules. Another stream of efforts integrates case-based reasoning or machine learning with soft-computing or traditional AI methods. Some of the combinations have been quite important and more extensively used, like neuro-symbolic methods, neuro-fuzzy methods and methods combining rule-based and case-based reasoning. However, there are other combinations that are still under investigation. In some cases, combinations are based on first principles, whereas in other cases they are created in the context of specific applications.

The Workshop is intended to become a forum for exchanging experience and ideas among researchers and practitioners who are dealing with combining intelligent methods either based on first principles or in the context of specific applications.

There were totally 20 papers submitted to the Workshop. Each paper was reviewed by at least two members of the PC. We finally accepted 12 papers (10 full and 2 short). Revised versions of the accepted papers (based on the comments of the reviewers) are included in these proceedings in alphabetic order (based on first author).

Five of the accepted papers deal with combinations of Genetic Programming or Genetic Algorithms with either non-symbolic methods, like Neural Networks (NNs) and/or Kalman Filters (Georgopoulos etal, Spanoudakis etal), or symbolic ones, like Decision Trees (Kalles etal) and Temporal Logic (Bennett and Magee). Another four papers deal with combinations of Case-Based Reasoning (CBR). One of them presents a short survey of CBR combinations (Prentzas and Hatzilygeroudis) and another one a combination with Agents (Teodorescu and Petridis). The rest two of them present CBR combinations with a Neuro-Fuzzy (Cocea and Magoulas) and a Neuro-Symbolic (Prentzas etal) approach respectively, leading to multi-combinations. Also, another two papers concern combinations of Fuzzy Logic with either NNs (Anastassopoulos and Iliadis) or Bayesian

Nets (Fogelberg etal). Finally, one of the papers combines a NN-based approach with a Natural Language Processing one (Foster etal).

Four of the above papers present combinations developed in the context of an application. Applications involve Medicine (Anastassopoulos and Iliadis), Education (Cocea and Magoulas, Kalles etal) and Economy (Spanoudakis etal).

We hope that this collection of papers will be useful to both researchers and developers.

Given the success of this first Workshop on combinations of intelligent methods, we intend to continue our effort in the coming years.

Ioannis Hatzilygeroudis
Constantinos Koutsojannis
Vasile Palade

ANN for prognosis of abdominal pain in childhood: use of fuzzy modelling for convergence estimation

George C. Anastassopoulos, Lazaros S. Iliadis

Abstract. This paper focuses in two parallel objectives. First it aims in presenting a series of Artificial Neural Network models that are capable of performing prognosis of abdominal pain in childhood. Clinical medical data records have been gathered and used towards this direction. Its second target is the presentation and application of an innovative fuzzy algebraic model capable of evaluating Artificial Neural Networks' performance [1]. This model offers a flexible approach that uses fuzzy numbers, fuzzy sets and various fuzzy intensification and dilution techniques to perform assessment of neural models under different perspectives. It also produces partial and overall evaluation indices. The produced ANN models have proven to perform the classification with significant success in the testing phase with first time seen data.

1 INTRODUCTION

The wide range of problems in which Artificial Neural Networks can be used with promising results, is the reason of their growth [2, 3]. Some of the fields that ANNs are used are: medical systems [4-6], robotics [7], industry [8 – 11], image processing [12], applied mathematics [13], financial analysis [14], environmental risk modelling [15] and others.

Prognosis is a medical term denoting an attempt of physician to accurately estimate how a patient's disease will progress, and whether there is chance of recovery, based on an objective set of factors that represent that situation. The inference about prognosis of a patient when presented with complex clinical and prognostic information is a common problem, in clinical medicine. The diagnosis of a disease is the outcome of combination of clinical and laboratorial examinations through medical techniques.

In this paper various ANN architectures using different learning rules, transfer functions and optimization algorithms have been tried. This research effort was motivated form the fact that reliable and seasonable detection of abdomen pain constitute attainments in effective treatment of disease and avoidance of relapses. That is why the development of such an intelligent model that can collaborate with the doctors will be very useful towards successful treatment of potential patients.

2 DIAGNOSTIC FACTORS OF ABDOMINAL PAIN

Several reports have described clinical scoring systems incorporating specific elements of the history, physical examination, and laboratory studies designed to improve diagnostic accuracy of abdominal pain [16]. Nothing is guaranteed, but decision rules can predict which children are at risk for appendicitis (appendicitis is the most common surgical condition of the abdomen). One such numerically based system is based on a 6-part scoring system: nausea (6 point), history of local RLQ pain (2 point), migration of pain (1 point), difficulty walking (1 point), rebound tenderness / pain with percussion (2 point), and absolute neutrophil count of >6.75 x 10`3/μL (6 point). A score <5 had a sensitivity of 96.3% with a negative predictive value of 95.6% for AA.

To date, all efforts to find clinical features or laboratory tests, either alone or in combination, that are able to diagnose appendicitis with 100% sensitivity or specificity have proven futile. Also, there is only one research work [4] in bibliography based on ANN that deals with the abdominal pain prognosis in childhood.

The incidence of Acute Appendicitis (AA) is 4 cases per 1000 children. However appendicitis despite pediatric surgeons' best efforts remains the most commonly misdiagnosed surgical condition. Although diagnosis and treatment have improved, appendicitis continues to cause significant morbidity and still remains, although rarely, a cause of death. Appendicitis has a male-to-female ratio of 3:2 with a peak incidence between ages 12 and 18 years. The mean age in the pediatric population is 6-10 years. The lifetime risk is 8.6% for boys and 6.7% for girls.

The 15 factors that are used in the routine clinical practice for the assessment of AA in childhood are: Sex, Age, Religion, Demographic data, Duration of Pain, Vomitus, Diarrhea, Anorexia, Tenderness, Rebound, Leucocytosis, Neutrophilia, Urinalysis, Temperature, Constipation. The sex (males), the age (peak of appearance of A.A in children aged 9 to 13 years), and the religion (hygiene condition, feeding attitudes, genetic predisposition) were in relation with a higher frequency for AA. Anorexia, vomitus, diarrhea or constipation and a slight elevation of the temperature (37^0 C - 38^0 C) were common manifestation of AA. Additionally, abdominal tenderness principally in the RLQ of the abdomen and the existence of the rebound sign, are strongly related with AA. Leucocytosis (>10.800 K/μl) with neutrophilia (neutrophil count > 75%) is considered to be a significant clue for AA. Urinalysis is useful for detecting urinary tract disease, normal findings on urinalysis are of limited diagnostic value for appendicitis.

The role of race, ethnicity, health insurance, education, access to healthcare, and economic status on the development and treatment of appendicitis are widely debated. Cogent arguments have been made on both sides for and against the significance of each socioeconomic or racial condition. A genetic predisposition appears operative in some cases, particularly in children in whom appendicitis develops before age 6 years. Although the disorder is uncommon in infants and elderly, these groups have a disproportionate number of compilations because of delays in diagnosis and the presence of comorbid conditions.

Democritus University of Thrace, Hellenic Open University
anasta@med.duth.gr, liliadis@fmenr.duth.gr

As diagnosis, there are four stages of appendicitis, including acute focal appendicitis, acute supurative appendicitis, gangrenous appendicitis and perforated appendicitis. These distinctions are vague, and only the clinically relevant distinction of perforated (gangrenous appendicitis includes into this entity as dead intestine functionally acts as a perforation) versus non-perforated appendicitis (acute focal and supurative appendicitis) should be made.

The present study is based on data set that is obtained from the Pediatric Surgery Clinical Information System of the University Hospital of Alexandroupolis, Greece. It consisted of 516 children's medical records. Some of these children had different stages of appendicitis and, therefore, underwent operative treatment. This data set was divided into a set of 422 records and another set of 94 records. The former was used for training of the ANN, while the latter for testing. A small number of data records were used as a validation set during training to avoid overfitting. Table 1 represents the stages of appendicitis as well as the corresponding cases for each one. The 3rd column of Table 1 depicts the coding of possible diagnosis, as they used for ANN training and testing stages.

Table 1. Possible diagnosis and corresponding cases.

Diagnosis		Coding	Cases
Normal	Discharge	-2	236
	Observation	-1	186
Operative treatment	No findings	0	15
	Focal appendicitis	1	34
	Phlegmonous or Supurative appendicitis	2	29
	Gangrenous appendicitis	3	8
	Peritonitis	4	8

3 NEURAL NETWORK DESIGN

Data were divided into two groups, the training cases (TRAC) and the testing cases (TESC). The TRAC consisted of 417 concrete medical data records and the TESC consisted of 101. Each input record was organised in a format of fifteen fields, namely sex, age, religion, area of residence, pain time period, vomit symptoms, diarrhoea, anorexia, located sensitivity, rebound, wbc, poly, general analysis of urine, body temperature, constipation. The output record contained a single field which corresponded to the potential outcome of each case.

The determination if the TRAC and TESC data sets was performed in a rather random manner. The training and testing sample size which would be sufficient for a good generalization was determined by using the Widrow's rule of thumb for the LMS algorithm which is a distribution free, worst case formula [2] and it is shown in the following equation 1. W is the total number of free parameters in the network (synaptic weights and biases) and ε denotes the fraction of the classification errors permitted during testing. The O notation shows the order of quantity enclosed within [2].

$$N = O\left(\frac{W}{\varepsilon}\right) \quad (1)$$

In the case examined here with 417 training examples used, the classification error that could be tolerated would be about 4%.

3.1 Description of the experiments performed

During experimentations, numerous ANN architectures, learning algorithms and transfer functions were combined in an effort to obtain the optimal network. For the Tangent Hyperbolic (TanH) transfer function the input data were normalized (divided properly) in order to be included in the acceptable range of [-3, 3] to avoid problems such as saturation, where an element's summation value (the sum of the inputs times the weights) exceeds the acceptable network range [17]. Standard back-propagation optimization algorithms using TanH, or Sigmoid or Digital Neural Network Architecture (DNNA) transfer functions, combined with the Extended Delta Bar Delta (ExtDBD) or with the Quick Prop learning rules [18, 19] were employed. The ExtDBD is a heuristic technique reinforcing good general trends and damping oscillations [20].

Modular and radial basis function (RBF) ANN applying the ExtDBD learning rule and the TanH transfer function were also used in an effort to determine the optimal networks. RBFs have an internal representation of hidden neurons which are radially symmetric, and the hidden layer consists of pattern units fully connected to a linear output layer [21, 22].

3.2 ANN evaluation metrics applied

Traditional ANN evaluation measures like the Root Mean Square Error (RMS error), R^2 and the confusion matrix were used to validate the ensuing neural network models. It is well known that the RMS error adds up the squares of the errors for each neuron in the output layer, divides by the number of neurons in the output layer to obtain an average, and then takes the square root of that average. The confusion matrix is a graphical way of measuring the network's performance during the "training" and "testing" phases. It also facilitates the correlation of the network output to the actual observed values that belong to the testing set in a visual display [17], and therefore provides a visual indication of the network's performance. A network with the optimal configuration should have the "bins" (the cells in each matrix) on the diagonal from the lower left to the upper right of the output. An important aspect of the matrix is that the value of the vertical axis in the generated histogram is the Common Mean Correlation (CMC) coefficient of the desired (d), and the actual (predicted) output (y) across the Epoch.

Finally, the FUSETRESYS (Fuzzy Set Transformer Evaluation System) that constitutes an innovative ANN evaluation system has been applied offering a more flexible approach [1].

3.3 Technical description of the FUSETRESYS ANN evaluation model

Fuzzy logic enables the performance of calculations with mathematically defined words called "Linguistics" [1, 23-25]. FUSETRESYS faces each training/testing example as a Fuzzy Set. It applies triangular or trapezoidal membership functions in order to determine the partial degree of convergence (PADECOV) of the ANN for each training/testing example separately. The following equations 2 and 3 represent a triangular and a trapezoidal membership functions respectively [1].

$$\mu_s(x;a,b,c)=\max\{\min\{\frac{x-a}{b-a},\frac{c-x}{c-b}\},0\}\ a<b<c \quad (2)$$

$$\mu_s(x;a,b,c,d)=\max\{\min\{\frac{x-a}{b-a},1,\frac{d-x}{d-c}\},0\} a<b<c<d \quad (3)$$

The model can produce various overall degrees of convergence (OVDECOV) for all of the training examples by applying either fuzzy T-Norm or fuzzy S-Norm conjunction operations, depending on the optimistic or pessimistic point of view of the developer. T-

Norms tend to produce lower aggregation indices so in the case of ANN evaluation they can be considered as a pessimistic approach, whereas the opposite happens with S-Norms [26]. In fact, each distinct Norm evaluates the performance of an ANN under a different perspective. For example the drastic product assigns the ANN a high OVDECOV only if it does not have extreme deviations between the desired and the produced classifications during the training/testing process [1] whereas the Einstein T-Norm acts in a more average mode. The following equations 4 and 5 present the drastic product and the Einstein product T-Norms. More details on fuzzy conjunction operators can be found in [26-28].

$$\mu_{\left(\tilde{A}\cap\tilde{B}\right)} = \text{Min}\{\mu_{\tilde{A}}(X),\mu_{\tilde{B}}(X)\} \;\; if \;\; Max\{\mu_{\tilde{A}}(X),\mu_{\tilde{B}}(X)\}=1 \;\; \text{else}$$

$$\mu_{\left(\tilde{A}\cap\tilde{B}\right)} = 0 \quad (4) \qquad \mu\left(\tilde{A}\cap\tilde{B}\right) = \frac{\mu_{\tilde{A}}(X)\mu_{\tilde{B}}(X)}{2-[\mu_{\tilde{A}}(X)+\mu_{\tilde{B}}(X)-\mu_{\tilde{A}}(X)\mu_{\tilde{B}}(X)]} \quad (5)$$

The fact that the FUSETRESYS evaluates each training/testing example separately, offers a more clear view of the ANN's performance. In this way the developers know if the network operates extremely bad or well in specific cases.

Also when there are several neurons in the output layer, the traditional approaches produce separate evaluation results for each one whereas the FUSETRESYS can produce an additive performance index (ADPERI) of the ANN. This could be done under different perspectives and under different degrees of optimism [1].

Finally the application of *fuzzy set hedges* offers the "dilution" and the "intensification" options. In this way by using the dilution approach the developer softens the membership function over the fuzzy set and weakens the membership constraints so that a point of the Universe of discourse is "truer" than it would be before [1, 27]. On the contrary the intensification hardens the MF over the FS and strengthens the membership constraints so that a point on the domain is "less true" than it used to be [1, 27]. The following equations 6 and 7 correspond to the intensification and dilution functions respectively.

$$\mu_{\text{int ensify}(A)}(X_i) = \mu_A^n(X_i) \quad (6) \qquad \mu_{dilute\,(A)}(X_i) = \mu_A^{\frac{1}{n}}(X_i) \quad (7)$$

In this way the ANN can be evaluated strictly by using a "*very well fit*" evaluation option, or in a more relaxed way by using the "*somewhat fit*" option. Of course it is in the developer's hand to decide the potential type of the ANN's evaluation and the degree of dilution or intensification. For a more detailed description of FUSETRESYS please see [1].

4 RESULTS AND DISCUSSION

4.1 ANN analysis

Several experiments were performed. The following table 2 presents the structure of the four most effective Back Propagation (BP) multilayer (ML) neural networks. In all cases of ANN models, the classical approach for overcoming the overfitting problem has been followed. More specifically, a set of validation data have been provided to the algorithm in addition to the training data. The algorithm has monitored the error with respect to this validation set, while using the training set to drive the gradient descent search. The number of weight tuning iterations performed by the system, were determined in each case based on the criterion of lowest error over the validation set. Two copies of the best performing weights are kept: one copy for training and another one of the best performing weights thus far.

Table 2. Structure of the four most successful ML ANN

Optimization algorithm	Input Layer	Hidden sub-layer neurons	Second Hidden sub-layer	Learning Rule/Transfer Function
ANN ML#1. Reinforcement ANN using BackPropagation	15	7	7	Genetic Algorithm /TanH
ANN ML#2. Multilayer Backpropagation	15	9	0	Norm-Cum_Delta/ TanH
ANN ML#3. Multilayer Backpropagation	15	9	9	Norm-Cum_Delta/ TanH
ANN ML#4. Multilayer Backpropagation	15	7	0	ExtDBD/ TanH

Table 3 shows the architecture and structure of the four most successful *radial basis function* (RBF) ANN.

Table 3. Structure of the two most successful RBF ANN

Optimization algorithm	Input Layer	Proto	Number of neurons Hidden layer	Output Layer	Learning Rule/Transfer function
ANN 5. ANN6 *Radial Basis Function*	15	30	2	1	Norm Cum_Delta / **Sigmoid or TanH**

The following Table 4 presents the training and testing results for the most successful ML and RBF ANN using R^2. Also for the three most successful networks, namely 2,3,4 the FUSETRESYS model was applied to determine the average degree of convergence. According to the results, the most suitable network was ML#3. The structure and the architecture of this successful network have been described in the above Table 2.

Table 4. Evaluation of the most successful ML and RBF ANN

ANN code	R^2 in Training	R^2 in Testing	Average Degree of success in Testing using FUSETRESYS for the three best ANN
1	0.8258	0.8247	
2	***0.9615***	***0.9471***	**0.9699**
3	***0.9721***	***0.9489***	**0.9716**
4	***0.9352***	***0.9588***	**0.9799**
5	0.9114	0.9000	
6	0.9346	0.9400	

The following figure 1 is a graphical representation of the all the PADECOV of the ML#2, for each testing example. The absolute degree of convergence has the value of one. A serious effort was

made towards the development of modular ANN (MODANN) for the classification problem solution. The term MODANN refers to the "*adaptive*" mixtures of *local experts* (LOCEXP) as proposed by [29].

They consist of a group of BP ANN referred to as local experts competing to learn different aspects of a problem. A "*gating ANN*" controls the competition and learns to assign different parts of the data space to different networks.

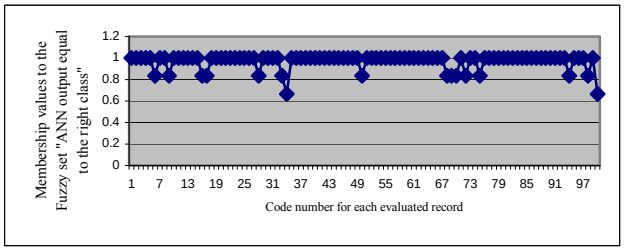


Figure 1. Representation of all the PADECOV of the network ML# 2

The LOCEXP have the same architecture but they can apply distinct learning rules or transfer functions. Also the number of the output processing elements of the gating network is equal to the number of LOCEXP used. The number of the neurons in the hidden layer of the gating network should be larger than the number of the output processing elements [17].

Table 5. Refereed Back Propagation using Gating Networks with two Competing local experts

Refereed #1 REF ANN using Gating ANN with 2 Local Experts				
Learning rule	**Transfer**	**Error**	**Output**	**Noise**
Local Expert's #1 functions				
ExtDBD	TanH	standard	Direct	Uniform
Local Expert's #2 functions				
Norm-Cum Delta	TanH	standard	Direct	Uniform
Local Expert's Architecture				
Input neurons 15		Hidden neurons 6		Output neurons 1
Gating ANN functions				
Learning rule	**Transfer**	**Error**	**Output**	**Noise**
ExtDBD	Linear	Standard	SoftMax	Uniform

The above table 5 presents the structure and the architecture of the optimal MODANN that was developed for the medical classification problem examined here. The performance of the developed modular network is very satisfying, having an R^2 value of 0.9434 and a FUSETRESYS produced average PADECOV equal to 0.9733 (using the Triangular membership function) in the testing process using the first time seen testing data set.

The following figure 2 depicts the *gating probabilities* for the optimal MODANN..

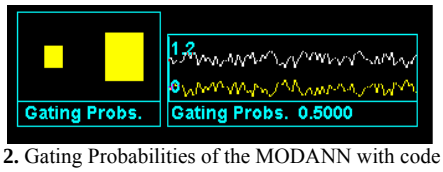


Figure 2. Gating Probabilities of the MODANN with code #1Ref.

Table 6. Small sample of the PADECOV indices

PADECOV by FUSETRESYS		
ML#2	ML#3	#1REF
0.83333	0.83333	1
0.83333	0.83333	0.833
1	1	1
0.83333	0.83333	1
1	1	1
0.83333	0.83333	1
0.833333	0.833333	1
OVDECOV Einstein		
0.98299	**0.97784**	**0.971**

The above Table 6 presents a small sample of the 101 distinct PADECOV values produced by the FUSTRESYS.

Also the Einstein T-Norm was applied for the determination of the overall degree of convergence of the ANN. The ML#2 ANN had a very high OVEDECOV index with a value of 0.98299 whereas the other ML#3 ANN and the MODANN #REF1 had OVEDECOV indices as high as 0.97. The Drastic Product T-Norm was not applied in this research effort because it was proven unnecessary from the data in table 5 where there were no serious indications of extreme bad ANN performance in any of the testing examples.

Table 7. OVDECOV values when intensification and dilution of first order was applied using Einstein product and Triangular membership function

	OVDECOV of ML **#2**	OVDECOV of **#1REF**	OVDECOV of ML **# 3**
Dilution "*Partly fit*"	0.99972	0.99934	0.99957
Intensification "*Very well fit*"	0.75932	0.64887	0.71033

5 CONCLUSIONS

The above research has obtained six ANNs with good level of convergence and it has proven that there exist at least four ANNs that have high performance indices, in the case of abdominal pain classification. Namely the best ANNs are two ML BP ANN, a RBF ANN and a MODANN using a referee gating network and two local experts. All of them have been described in the previous sections.

A very interesting part of the whole research effort is the application of an innovative ANN evaluation model called FUSETRESYS that uses fuzzy logic and fuzzy algebra proposed in [11].

The new evaluation scheme has performed individual convergence indices namely PADECOV, for the output of each single data record used in the testing phase. The worst PADECOV value equals to 0.6666 which actually is the degree of membership of each data record to the FS "*Actual output value equal to the desired value*". This worst case appears three times exactly in the

same cases of data records, for the ML#2, ML#3, #1REF ANN and it shows that the classification capacity of the developed networks is not bad even in the worst cases. This conclusion becomes stronger by considering the fact that the second worst PADECOV index has a value of 0.833.

If an overall ANN validation is performed the traditional evaluation instruments agree with the FUSETRESYS that the most suitable ANN is the ML BP with code# 4 whereas all of the other developed ANN have almost an equally good performance. The Einstein T-Norm produces a higher "*good performance index*" for the MODANN than the traditional methods.

As it can be seen in table 7, the OVDECOV indices have very high values for ML#2 and for REF#1 and ML#3 networks when a "Partly *fit*" validation is performed. There is significant differentiation when a very strict evaluation is done under the linguistic "*Very well fit*". The OVDECOV indices fall from 0.99 to 0.75 for ML#2, from 0.99 to 0.65 for #REF and from 0.99 to 0.71 for ML#3 respectively. This is a very useful approach and it shows the actual power of FUSETRESYS due to the fact that it shows the differentiation of the average convergence degree of the three ANN when more strict validation methods are applied. So ANN fed with the same data records in testing and appearing to have more or less the same performance, they are very seriously differentiated when more strict convergence validation methods are performed.

The proposed ANN architecture faces the appendicitis prediction quite satisfactory, based on both the above presented results, and the pediatric surgeon's opinion that used these ANNs in their everyday routine clinical practice.

The innovative ANN evaluation model that was applied successfully in this research effort will be used extensively in the future, in an integrated effort to check its validity under various perspectives.

ACKNOWLEDGEMENTS

We would like to thank the pediatric surgeons of the Pediatric Surgeon Department of Medical School of Democritus University of Thrace, for their contribution in the concession of the medical records.

REFERENCES

[1] L. Iliadis, 'An intelligent Artificial Neural Network evaluation system using Fuzzy Set Hedges: Application in wood industry', *Proceedings of the 19th IEEE ICTAI The Annual IEEE International Conference on Tools with Artificial Intelligence.* IEEE Volume II, 366-370, Los Alamitos California, 2007.

[2] S. Haykin, *Neural Networks: A comprehensive foundation,.* McMillan College Publishing Company, New York, 1999.

[3] P. Picton, *Neural Networks*, (2nd edition) Palgrave, New York, USA, 2000.

[4] D. Mantzaris, G. Anastassopoulos, A. Adamopoulos, I. Stephanakis, K. Kambouri and S. Gardikis, 'Abdominal Pain Estimation in Childhood based on Artificial Neural Network Classification', *Proc. of the 10th International Conference on Engineering Applications of Neural Networks (EANN 2007),* 129-134, August, 2007.

[5] G.P.K. Economou, D. Lymberopoulos, E. Karavatselou and C. Chassomeris, 'A new concept toward computer-aided medical diagnosis - A prototype implementation addressing pulmonary diseases', *IEEE Transactions on Information Technology in Biomedicine*, **5** (1): 55-66, (2001).

[6] J. Shieh, S. Fan and W. Shi, 'The intelligent model of a patient using artificial neural networks for inhalational anaesthesia', *J. Chin. Inst. Chem. Engrs.,* **33**, No. 6, 609-620, (2002).

[7] V. Rankovic and I. Nikolic, 'Control of industrial Robot using neural network compensator', *Theoretical Applications of Mech*., **32**, No. 2, 147-163, (2005).

[8] S. Avramidis and L. Iliadis, 'Wood-water sorption isotherm prediction with artificial neural networks: a preliminary study', *Holzforschung,* **59,** 336 – 341, (2005).

[9] S. Mansfield, L. Iliadis, S. Avramidis, 'Neural Network Prediction of Bending Strength and Stiffness in Western Hemlock', *HOLZFORSCHUNG*, **61**, Issue 6, 707-716 Walter De Gruyter & Co Berlin, New York, 2007.

[10] B. Cannas, A. Fanni, A. Montisci, G. Murgia and P. Sonato, 'Dynamic Neural Networks for Prediction of Disruptions in Tokamaks', *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks (EANN)*, Thessaloniki, Greece. 2007.

[11] L. Iliadis, S. Spartalis and S. Tachos, 'A Fuzzy intelligent Artificial Neural Network evaluation System: Application in Industry', *Proceedings of the 10th International Conference Engineering Applications of Neural Networks*, 320-326, Thessaloniki, Greece, 2007.

[12] R. Kuhn, R. Bordas, B. Wunderlich, B. Michaelis and D. Thevenin, 'Colour class identification of tracers using artificial neural networks', *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks (EANN),* Thessaloniki, Greece, 2007.

[13] L. Iliadis and S. Spartalis, 'Artificial Neural Networks equivalent to Fuzzy Algebra T_Norm conjunction operators', *Proceedings (Book of extended abstracts) of the ICCMSE 2007* Published by the AIP (American Institute of Physics), USA, 2007.

[14] P. Hajek and V. Olej, 'Municipal creditworthiness Modelling by clustering methods', *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks (EANN)*, Thessaloniki, Greece, 2007.

[15] L. Iliadis, 'A decision support system applying an integrated Fuzzy model for long - term forest fire risk estimation', *Environmental Modelling and Software*, **20**, Issue 5, 613-621, (2005).

[16] M. Blazadonakis, V. Moustakis, and G. Charissis, 'Deep Assessment of Machine Learning Techniques Using Patient Treatment in Acute Abdominal Pain in Children', *Artificial Intelligence in Medicine,* **8,** 527- 542, (1996).

[17] Neuralware, *Getting started. A tutorial for Neuralworks Professional II/PLUS,* Carnegie, PA,USA, 2001.

[18] R.A. Jacobs, 'Increased rates of convergence through learning rate adaption', *Neural Networks* **1**, 295-307, (1988).

[19] A.A. Minai, R.D. Wiliams, 'Acceleration of back-propagation through learning rate and momentum adaption', *International Joint Conference on Neural Networks*, **I**, 676-679, (1990).

[20] D.E. Rummelhart, G.E. Hinton and R.J. Wiliams, Learning internal representations by error propagation. Institute for Cognitive Science Report 8506. San Diego, University of California (1985).

[21] J. Platt, "A resource allocating network for function interpolation", *Neural Computation*, **3**, 213-225, (1991).

[22] J. Moody, C.J. Darken, 'Fast learning in networks of locally tuned processing units', *Neural Computation*, **1**, 281-294, (1989).

[23] R. Callan, *The Essence of Neural Networks.* Prentice Hall, UK, 1999.

[24] W. Pedrycz, 'Structural interpolation and approximation with fuzzy relations: A study in knowledge reuse', *Journal Studies in Fuzziness and Soft Computing,* **215**, 65-77, (2007).

[25] B. J. Park, W. Pedrycz, S.K. Oh, 'Fuzzy polynomial neural network: hybrid architectures of fuzzy modelling', *IEEE Trans. Fuzzy Systems*. **10** (5), 607-621, (2002).

[26] V. Kecman, *Learning and Soft Computing,* MIT Press. London England, 2001.

[27] E. Cox, *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Elsevier Science*, USA,* 2005.

[28] T. Calvo, G. Mayor and R. Mesira, *Aggregation Operators: New Trends and Applications,* (Studies in Fuzziness and Soft Computing). Physica-Verlag, Heildeberg, 2002.

[29] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton, 'Adaptive mixtures of local experts', *Neural computation*, **3**, 79-87, 1991.

Using Genetic Programming to Learn Models Containing Temporal Relations from Spatio-Temporal Data

Andrew Bennett and **Derek Magee** [1]

Abstract. In this paper we describe a novel technique for learning predictive models from non-deterministic spatio-temporal data. Our technique learns a set of sub-models that model different, typically independent, aspects of the data. By using temporal relations, and implicit feature selection, based on the use of 1st order logic expressions, we make the sub-models general, and robust to irrelevant variations in the data. We use Allen's intervals [1], plus a set of four novel temporal state relations, which relate temporal intervals to the current time. These are added to the system as background knowledge in the form of functions. To combine the sub-models into a single model a context chooser is used. This probabilistically picks the most appropriate set of sub-models to predict in a certain context, and allows the system to predict in non-deterministic situations. The models are learnt using an evolutionary technique called Genetic Programming. The method has been applied to learning the rules of snap, and uno by observation; and predicting a person's course through a network of CCTV cameras.

1 Introduction

Learning predictive models from spatial-temporal data is, in general, a hard problem. Events and activities can have variations in their spatial, and temporal scope; include multiple (variable numbers of) objects; can overlap temporally with other events, and activities; and happen in a non-deterministic manner. A model for predicting spatio-temporal events must support this complexity. Our novel technique learns a set of sub-models that model different, typically independent, aspects of data. The sub-models can, in addition to object properties, use temporal relations to describe the scene, and implicit feature selection, based on the use of 1st order logic expressions, to make them robust to irrelevant variations in the data. To combine the sub-models into a single model a context chooser is used. This picks the most appropriate set of sub-models to predict in a certain context, and allows the system to predict in non-deterministic situations. Using the combination of sub-models and the context chooser also reduces the complexity of the model search space, and allows the system to learn a global sub-model that matches most of the dataset, and then learn simple sub-models to cover the cases where the global sub-model does not work.

This approach extends our previous work [2], by allowing a qualitative, as well as a markovian representation of time. This is done by replacing the step-wise markovian view with temporal relations like Allen's intervals [1], and a set of four additional relations to relate the temporal state of objects to the current time. We use Genetic Programming to learn the models, and present an improved fitness function. The system has been successfully tested on handcrafted snap, and uno datasets, along with learning from video the structure of a set of mock CCTV cameras.

There has been much previous work on learning from spatio-temporal domains. Traditional methods usually require a fixed dimensionality vector, existing with canonical ordering / constant meaning, to represent the world. To construct this vector often requires knowledge of the domain, making these methods hard to use in a problem domain where the structure of the domain is variable, and not known a priori. One approach to modelling data of variable dimensionality is to take statistics of a variable size set [8]. This produces a fixed set description, however spatial relationship information is lost in this process. If this information is important within a domain this leads to a poor model. Feature selection can be used to find the most relevant subset of the data, which then allows for a more general model to be built. However, the relevant subset may change from one context to another.

Temporal modelling approaches such as Markov chains, Hidden Markov Models (HMMs) and Variable Length Markov Models (VLMMs) [7] use a description based on graphs to model state transitions. These methods also usually need a fixed dimensionality vector with canonical ordering for each observation. There does not have to be a fixed dimensionality for every observation vector, as theoretically each observation vector can have a different number of dimensions. It is possible to optimise their structure by using local optimisation approaches based on information theory [3]. In VLMMs this optimisation acts as kind of temporal feature selection, but as the input variables stay in the same fixed order spatial feature selection is not performed.

Bayesian networks are a generalisation of probabilistic graph based reasoning methods like HMMs and VLMMs. Again these networks require a fixed input vector, but again their relational structure can be optimised by local search [12], genetic algorithms [5], or MCMC [6] usually based on information theoretic criteria.

An alternative to using graph based methods is to use (1st order) logical expressions. Feature selection is implicit in the formalism of these expressions. Logical expressions also make no assumptions about the ordering of variables, so there is no need to have a have them in a fixed ordering. Progol [14] and HR [4] are Inductive Logic Programming (ILP) methods. In general ILP takes data and generates a set of logical expressions describing the structure of the data. Progol does this by iterative subsumption using a deterministic search with the goal of data compression. HR does this by using a stochastic search using a number of specialist operators. This is similar to Genetic Programming which is described below. These approaches suffer from a number of disadvantages. Firstly, logical expressions are deterministic, so it is hard for then to model non-deterministic situations. However, there has been much work on combining (1st order)

[1] University of Leeds, UK, email: {andrewb,drm}@comp.leeds.ac.uk

logic and probability to solve this problem [16] and [9]. Secondly Progol's search is depth bounded, which limits the size of problems it can work on, as explained in [15]. Thirdly Progol's fitness function is only based on how well the model compresses the data, and not how well the model predicts the data. This can cause incorrect, or invalid models to be produced.

Genetic Programming (GP) [10] is a evolutionary method, similar to genetic algorithms, for creating a program that model a dataset. In a similar way to HR, it takes a dataset data, a set of terminals, and a set of functions; and using a set of operators generates a binary tree that models the data.

Qualitative representations can be used to describe spatio-temporal data in an abstract manner. [1] describes a set of seven temporal relations to represent temporal interactions between objects.

There has been previous work in learning of spatio-temporal models from video by [15] who produced a system that could learn basic card games. It had three parts: an attention mechanism, unsupervised low-level learning, and high-level protocol learning. The attention mechanism uses a generic blob tracker, that locates the position of the moving objects. From this a set of features including: colour, position and texture are extracted. The data is clustered into groups. Using these clusters new input data is assigned its closest cluster prototype. A symbolic data stream is then created by combining together the clustered data, with time information. The symbolic stream is passed to Progol, which builds a model of the data. Once the model has been learnt it can be applied to new data. This allows the system to interact in the world.

[17] looked at learning event definitions from video. A raw video of a scene is converted into a polygon representation. This is then transformed into a force-dynamic model which shows how the objects in the scene are in contact with one another. Using this data and-meets-and (AMA) logic formulae describing the events are learnt using a specific-to-general ILP approach. Work in the area of learning from spatial-temporal data, such as the previous two approaches have inspired our work.

The reminder of this paper will take the following form. The second section looks at previous work about the architecture for the models. The subsequent section looks at an extension to this work to incorporate temporal relations into the sub-models. The subsequent section describes how these models are learnt by Genetic Programming. The subsequent section presents an evaluation of our system, and the final section shows the conclusions of the work and the further work.

2 Architecture for Models of Spatio-Temporal Data

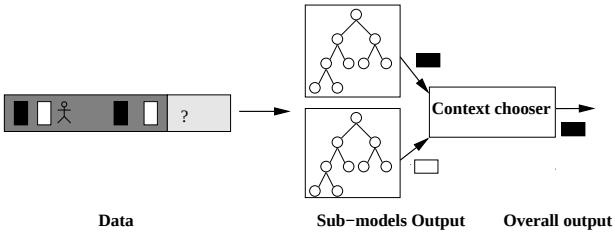


Figure 1. This figure shows the architecture of our model. It has two parts: a set of sub-models, and a context chooser to decide how to use the sub-models in different situations.

An architecture to represent a model of spatio-temporal data, along with associated learning methods is described in our previous work [2]. We use this architecture as shown in Figure 1. It is broken down into two parts: the sub-models, and the context chooser. The sub-models each model a separate part of the underlying process generating the data. Each sub-model contains two sections: a search section, and an output section. The search section looks for a particular pattern in the dataset. A query language, created by ourselves, having some similarity to SQL and Prolog, is used to describe the actual search, and a binary tree is used to represent it. The output section describes what is implied if the search returns true. This will be a set of entities and relations, and their properties the sub-model predicts. Figure 2 shows an example of a sub-model.

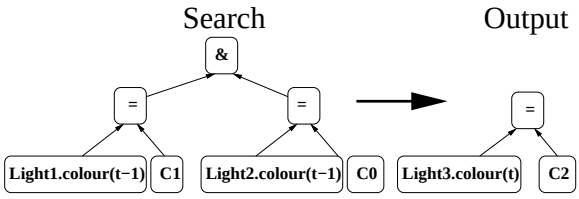


Figure 2. This shows a sub-model matching a traffic light with colour c1, and a different light having a colour c0 both at current time - 1. If the expression evaluates true it will output a new light which has a colour c2, at the current time.

The context chooser is used to decide how to combine the sub-models in different situations. It takes as its input a boolean vector describing which sub-models have evaluated true, and returned outputs, and using a probability distribution decides which ones will form the overall output. A context S_n is defined as a set of sub-models M producing an output in a given context, for example $S_n = M_1, M_2$ represents that M_1, and M_2 have search sections that have evaluated true at the same time. For each context a probability distribution over the possible combinations of model outputs for that context is defined, for example $P_n(M_1), P_n(M_2), P_n(M_1, M_2)$, where $\sum_j P_n(j) = 1$. This distribution is formed from the frequency of occurrence of each situation in the training data in the given context. This can be implemented as a sparse hash table.

3 Incorporating Temporal Relations into Sub-models

To evaluate the sub-models history data from the world is required. The search section of the sub-model uses data pointers to reference particular data items in the history. The search section of the sub-model is then evaluated with respect to this data. If the search section evaluates true, then the output section is implied. In our previous work [2] each data pointer could only reference fixed quantified time points in the history, as shown in Figure 2. The use of this qualitative markovian representation of time implies an exact ordering of the events. When multiple independent events are happening simultaneously this representation will fail, and an alternative method of representing temporal ordering is necessary. In order to quantify temporal ordering in the data we use a combination of Allen's intervals [1], and four novel temporal state relations. Allen's intervals describe temporal relations between objects. There are seven relations which are: meets, starts, finishes, during, before, overlaps, and equal to. Along with describing temporal relations between objects in the history, we need to describe how the objects relate to the current time. An object

goes through a series of temporal states, based on how its start and end time relates to current time, these are described Figure 3. Firstly the object is entering the world, its end time is unknown, but its start time is the same as the current time. Secondly the object exists in the world, again the end time is unknown, but its start time is less than the current time. Thirdly the object is leaving the world and its end time is equal to the current time. Finally the object has left the world, where both its start, and end times are less than the current time.

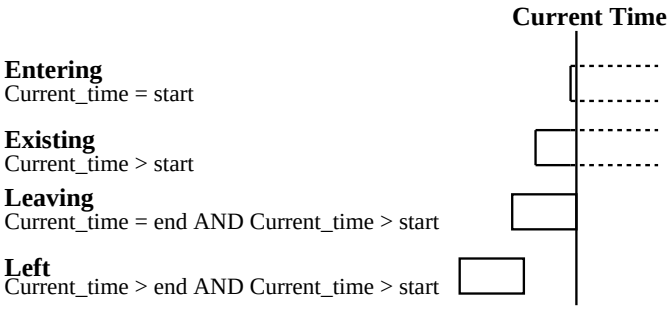


Figure 3. This shows the four temporal states, with respect to current time, an object can be in: entering, existing, leaving, and left. The dotted lines represent that we don't know when the object will leave the world.

Both the Allen's intervals, and our additional temporal state relations, are represented in the system as functions of the data, that appear in the search section of the sub-models. These relations do not appear in the data; only the temporal range of individual objects occurs in the data. As the data pointers can be used over the entire history, it is quite likely that a sub-model will evaluate on many different parts of the history. To resolve this issue we just use the result which includes the most recent data. The justification for this is the sub-model will have already output this information at a previous time in other situations.

4 Learning the Models from Data

Previously in our previous work [2] it has been shown that it was intractable to find the set of optimal sub-models by exhaustive search, for all but the simplest problems. The search space is complex, so a stochastic search method was chosen as an alternative. We use Genetic Programming [10], which has already been successfully used for pattern recognition tasks [11].

Genetic Programming (GP) [10] evolves a population of programs until a program with the desired behaviour is found. It is a type of genetic algorithm, but the programs are stored as binary trees, and not as fixed length strings. Functions are used for the nodes, and terminals (for example constants, and variables) are used for the leaf nodes. In order for the population to evolve a fitness function (in our case a predictive accuracy score) must be defined. This score will be used by the GP system to decide which programs in the current generation to use to produce the next generation, and which ones to throw away. To initialise the system, a set of randomly generated programs must be created. Each then receive a score using the fitness function. Algorithms including crossover, mutation and reproduction use the programs from the current generation to create a new generation. Crossover takes two programs and randomly picks a sub-tree on each program, these two trees are swapped over, creating two new programs. Mutation takes one program, randomly picks a sub-tree on it, and replaces it with a randomly generated sub-tree. Reproduction copies a program exactly as it is into the new generation. The programs in the new generation are then scored based on how well data is predicted, and the process is repeated. The GP system will stop when a certain fitness score is reached, or a certain number of generations has passed.

In our implementation of GP we assume that a program is a model containing a context chooser, and a set of sub-models. To initialise the population we generate a set of models just containing one randomly generated sub-model. The sub-model is produced using Koza's ramped half and half method [10]. We apply a hierarchical structure to our sub-models in a similar manner to [13], to try and cut down the search space, and to make finding a solution more efficient.

A set of operators is then used to evolve the population. There are two kinds of operators. Firstly there are operators that try to optimise sub-models which are used in the model, and secondly there are operators that optimise the sub-models themselves. A technique called tournament selection [10] is used to pick a model from the population. Tournament selection picks n models at random from the population, and returns the one with the lowest score, for our experiments we set n to be 5. The operators used to optimise sub-models which are used in the model are shown below:

Reproduction A set number of models are picked via tournament selection and copied directly into the new population.

Adding in a sub-model from another model Two models are picked by tournament selection. A sub-model from the first picked model is randomly selected, and added to the second chosen model.

Replacing a sub-model Again two models are picked by tournament selection, and a sub-model from the first chosen model is then replaced by a sub-model randomly selected from the second chosen model.

Removing a sub-model A sub-model is picked by tournament selection, and a randomly selected sub-model is removed.

The only operator used to optimise the sub-models themselves is crossover. In crossover two models are picked using tournament selection. A sub-model from each model is then randomly selected, and standard crossover [10] is performed on these sub-models.

To score the models a fixed length window is randomly moved over the dataset. At each generation two random locations are picked: one for training, and one for testing. In the training phase the probability distribution used in the context chooser is calculated. In the testing phase the fitness of a model (m) is evaluated over a windowed section of the dataset (w). For each position in the window the model is given a set of history data (h), calculated from the window, and is queried to produce a prediction. This produces a set of possible corresponding outputs (o), and a set of possible corresponding output likelihoods (ol). The similarity (C) of each output with the actual output (r), is computed using the $FindBestMatch$ function, as shown in Equation 1. This function takes the set of actual output, and the set of model output, and firstly pads out them out with blank data so that they are the same size. Then for each item in the actual output set, a unique match in the model output set is found. For each of the matches a comparison is done between the two objects. The comparison looks at how similar each of the properties in the two objects are. Each of the comparisons are summed together to produce a score that shows how good that set of matches is. An exhaustive search is then performed over all the possible combination of matches to find the best (maximal) matching score. The result is then multiplied by its output likelihood. From this the best (maximal) output is found. This

is then repeated over the rest of window, and the results summed and then normalised to produce (S), as shown in Equation 2. This fitness function is an improved version to the one described in our previous work [2], as it can be applied to non-deterministic datasets.

$$C(o, r) = FindBestMatch(o, r) \quad (1)$$

$$S(m, w) = \frac{1}{|w|} * \sum_{i} Max_n(ol_n * C(o_n, r_i)) \quad (2)$$

The system runs in two stages, and will stop running once it exceeds a maximum number of generations. Firstly the system is initialised in the manner described above, and then for five generations it works out the best set of sub-models to use in the models. To do this the system uses reproduction (10%), removing (10%), adding (40%), and replacement (40%). Next the system will optimise these models to find the best solution. It uses crossover (60%), reproduction (10%), removing (10%), adding (10%), and replacement (10%).

5 Evaluation

Our method was evaluated on three different datasets, which were: handcrafted uno data, handcrafted snap data, and data from people walking through a network of mock CCTV cameras. More detail about these datasets is presented in the following section.

5.1 CCTV Data of a Path

A 10 minute video of people walking along a path containing a junction was filmed. This was then used to mock up a network of CCTV cameras. Figure 4 shows a frame from the video. Virtual motion detectors, representing CCTV cameras, were hand placed over the video has shown in Figure 4. Using frame differencing, and morphological operations, the video was processed to determine the location of the motion. If the number of moved pixels in a region exceeded a fixed threshold then the virtual detector outputted that motion had occurred at that location. Hysteresis on the motion detection is implemented as a 2 state, state machine (where the states are motion/no motion). The state machine requires a numbers of frames (normally 10) of stability to change state. The data produced is then placed in a datafile with a motion event recorded per state change going from no motion to motion. This was used to create a training datafile containing 84 state changes and a test file containing 46 state changes.

5.2 Snap

The snap dataset was handcrafted, but the format of it was similar to the snap dataset used in the work of [15]. The snap sequence is the following: initially the computer will see a blank scene, then it will hear the word play, next two coloured cards will be seen. Either they will be both put down at the same time, or put down one by one. If they are the same then the word "equals" will be heard, otherwise "different" will be heard. Then the cards are removed, again either one by one, or at the same time. We ask the computer to only learn the sections where a human is speaking, as it would be impossible to accurately predict the next two cards because they are essentially random. Again three datasets were prepared: a non-noisy, and noisy training set, and a non-noisy test set. All the datasets contained around 50 rounds of snap. The noisy data was generated by adding 10% noise to the non-noisy training set. The noise took the form of removing cards, removing the play state, and changing the output state, for example making the output not equal when it should be equal.

5.3 Uno

The handcrafted uno dataset has a similar sequence to the snap dataset. Again the computer will initially see a blank scene. Then play will be heard. Next two cards, each one having one of three possible coloured shapes on them, will be placed down either at the same time, or one by one. If the two card have the same coloured shape on them the "same" is heard; or if they have shapes of the same colour then "colour" is heard; or if they have the same shapes on then "shape" is heard; or if the cards are different then "nothing" is heard. The cards are then removed either together, or one by one. Three datasets were created: a non-noisy training set, a noisy training set, and a non-noisy test set. Each one contained around 50 rounds of uno. Again noisy data was prepared by adding 10% of noisy data to the non-noisy training data. The noise took the same form as the noisy snap data.

Figure 4. This figure firstly shows a frame of the video with a person taking a decision at the junction point, and secondly it shows where the virtual detectors are on the video.

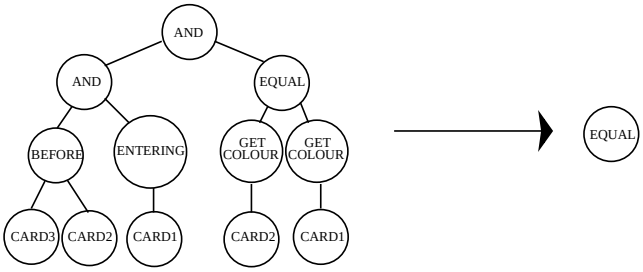


Figure 5. This shows one of the sub-model results for the snap dataset. It is predicting the equal state, by using the properties of three cards. If card3 occurs before card2; and card1 has just entered the world; and the colour of both card1, and card2 is the same then the sub-model evaluates true, and Equal will be returned.

6 Results

To test the system five runs were allocated to each possible combination of dataset. For each run a different random number seed was used to initialise the system. The tests were run on a 2GHz machine having 8GB memory.

To evaluate how well the models have been learnt they were tested on a separate test set. Two metrics were used to evaluate the results: coverage, and prediction accuracy. Coverage (C) scores if the system can correctly predict the dataset (ie. the probability of correct

prediction is greater than 0%) and is the number of correct predictions (pc) divided by the dataset size (d) as shown in Equation 3. Prediction accuracy (A) scores with what probability the correct prediction is made, and is the sum of the likelihoods of each correct prediction (pl) divided by the dataset size, as shown in Equation 4. In non-deterministic scenarios this will not be 100%.

$$C = \frac{pc}{d} \tag{3}$$

$$A = \frac{pl}{d} \tag{4}$$

Both the snap datasets were tested on a population size of 4000, and the system was run for 65 generations, taking around 5 hours to do each run. All the runs using the non-noisy datasets were successful. However the models did not get 100% coverage because they failed to produce any output at the start of the test dataset as there was insufficient items in the history. Figure 5 shows an example of this, as it will only evaluate once there are three cards in the history. Four of the results did not predict the first two items in the test dataset, and one of the results only failed to predict the first item. Two out of the five runs using the noisy snap dataset got an exact solution. The noise effected the models causing the sub-models to model incorrect parts of the dataset. This was because some of the noise added to the noisy training set changed the outcomes for some rounds of snap, this then causes the system to model this noise, and to incorrectly predict the outcomes in the test set. Again, like in the non-noisy snap models there was problems predicting the start of the test dataset. The models themselves made use of both the Allen's intervals, and the temporal state relations. Figure 5 shows one of the sub-models produced from the non-noisy snap training set. It shows the use of Allens intervals (the before relation), and the temporal state relations (the enter relation). Most of the models contained four sub-models in them.

The uno datasets were run on a population of size 6000, and for 65 generations, taking around 7 hours to do each run. One out of five runs on the non-noisy dataset managed to get the correct solution, but it did not get 100% coverage because it did not have enough history at the start of the test set to predict the initial items. The rest of the non-noisy results were very close to the solution, and probably needed more generations to find the exact solution. The models themselves were very similar to the models produced for the snap datasets. Both Allen's intervals, and the temporal state relations were used. None of the runs for the noisy dataset managed to produce an exact result, with the noise causing the sub-models to model incorrect parts of the dataset.

The runs using the path dataset used a population size of 2000, and the system was run for 65 generations, taking around 3 hours to do each run. All the runs using the non-noisy dataset predicted well in the main section of the test dataset, but failed to predict well at the start of the test dataset, due to lack of history. Some of the runs also failed to predict infrequently occurring actions in the test set. In the runs using the noisy training set all the models learnt the frequently occurring actions, but they all started to learn some of the noise in the dataset, and this effected their scores on the test dataset. Both the non-noisy and noisy models used Allen's intervals, and the temporal state relations.

Training Dataset	Number of runs	C(%)	A(%)
Snap No Noise	1	99.9	99.9
	4	99.8	99.8
Snap Noise	2	99.8	99.8
	1	99.8	96.6
	1	96.0	94.8
	1	96.0	93.3
Uno No Noise	1	99.7	99.7
	1	97.2	97.2
	1	94.0	92.0
	1	91.5	89.7
	1	88.8	88.8
Uno Noise	2	96.8	88.4
	1	95.1	95.1
	1	94.3	90.4
	1	88.5	87.1
Path No Noise	1	97.9	92.1
	1	97.9	89.3
	1	96.8	88.8
	1	95.8	90.0
	1	94.7	88.5
Path Noise	1	93.6	85.8
	1	92.6	85.2
	1	91.5	82.4
	1	90.8	83.3
	1	90.1	80.7

Figure 6. This figure shows the results for the snap, uno and path datasets. The number of runs column shows for each training set how many runs got the same coverage, and accuracy scores.

7 Conclusions

We have extended the previous work of [2] and shown that that it is possible, by the use of temporal relations, to use a qualitative, as well as a markovian representation of time. This technique is important for a number of reasons. Firstly it produces models that are robust to irrelevant variations in data. Secondly, it allows the system to learn from a dataset containing single actions, and then be able to predict from a dataset containing multiple overlapping actions.

In future work will be looking into using spatial, as well as temporal relations in the system. We are also looking into trying out quantitative relations, so that a relation will not work on objects that are either too close, or too far away. We will also be looking into changing the output from a sub-model based on what data the search section has evaluated on. Finally we will be looking at speed improvements to the system so that the run time can be reduced.

REFERENCES

[1] James Allen, 'Maintaining knowledge about temporal intervals', *Communications of the ACM*, **26**, 832–843, (1983).

[2] A. Bennett and D. Magee, 'Learning sets of sub-models for spatio-temporal prediction', in *Proceedings of AI-2007, the Twenty-seventh SGAI International Conference on Innovate Techniques and Applications of Artificial Intelligence*, pp. 123–136, (2007).

[3] Matthew Brand, 'Pattern discovery via entropy minimization', in *Artificial Intelligence and Statistics*, (1998).

[4] Simon Colton, Alan Bundy, and Toby Walsh, 'Automatic identification of mathematical concepts', in *International Conference on Machine Learning*, (2000).

[5] R. Etxeberria, P. Larranaga, and J.M. Picaza, 'Analysis of the behaviour of genetic algorithms when learning bayesian network structure from data', *Pattern Recognition Letters*, **13**, 1269–1273, (1997).

[6] Nir Friedman and Daphne Koller, 'Being bayesian about network structure', *Machine Learning*, **50**, 95–126, (2003).

[7] Aphrodite Galata, Neil Johnson, and David Hogg, 'Learning behaviour models of human activities', in *British Machine Vision Conference (BMVC)*, (1999).

[8] Kristen Grauman and Trevor Darrell, 'The pyramid match kernel:discriminative classication with sets of image features', in *International Conference on Computer Vision*, (2005).

[9] R Haenni, 'Towards a unifying theory of logical and probabilistic rea-

soning', in *International Symposium on Imprecise Probabilities and Their Applications*, pp. 193–202, (2005).
[10] John Koza, *Genetic Programming*, MIT Press, 1992.
[11] John Koza, *Genetic Programming II*, MIT Press, 1994.
[12] Philippe Leray and Olivier Francios, 'Bayesian network structural learning and incomplete data', in *Adaptive Knowledge Representation and Reasoning*, (2005).
[13] David Montana, 'Strongly typed genetic programming', in *Evolutionary Computation*, (1995).
[14] S.H. Muggleton and J. Firth, 'CProgol4.4: a tutorial introduction', in *Relational Data Mining*, 160–188, Springer-Verlag, (2001).
[15] Chris Needham, Paulo Santos, Derek Magee, Vincent Devin, David Hogg, and Anthony Cohn, 'Protocols from perceptual observations', *Artificial Intelligence*, **167**, 103–136, (2005).
[16] N. J. Nilsson, 'Probabilistic logic', *Artificial Intelligence*, **28**, 71–87, (1986).
[17] Jeffrey Mark Siskind, 'Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic', *Articial Intelligence Research*, **15**, 31–90, (2000).

Combining Intelligent Methods for Learner Modelling in Exploratory Learning Environments

Mihaela Cocea and **George D. Magoulas** [1]

Abstract. Most of the existing learning environments work in well-structured domains by making use of or combining AI techniques in order to create and update a learner model, provide individual and/or collaboration support and perform learner diagnosis. In this paper we present an approach that exploits the synergy of case-base reasoning and soft-computing for learner modelling in an ill-structured domain for exploratory learning. We present the architecture of the learner model, the knowledge formulation in terms of cases and illustrate its application in an exploratory learning environment for mathematical generalisation.

1 INTRODUCTION

Several AI techniques have been proposed in intelligent learning environments, such as case-based reasoning [27], [10], bayesian networks [4], [6], neural networks [2], genetic and evolutionary algorithms [24], neuro–fuzzy systems [26], as well as synergistic approaches, such as genetic algorithms and case-based reasoning [13], hybrid rules integrating symbolic rules with neurocomputing [11], and expert systems with genetic algorithms [18].

Exploratory Learning Environments (ELEs) belong to a particular class of learning environments built on the principles of constructivism paradigm for teaching and learning. ELEs place the emphasis on the opportunity to learn through free exploration and discovery rather than guided tutoring. This approach has proved to be beneficial for learners in terms of acquiring deep conceptual and structural knowledge. However, discovery learning without guidance and support appears to be less effective than step-by-step guiding learning environments [16]. To this end, an understanding of learner's behaviour and knowledge construction is needed [22].

Most existing ELEs use simulations as a way of actively involving learners in the learning process (e.g. [28], [14]) and exploit cognitive tools [29] to support their learning. Few such systems model learner's knowledge/skills; for example [4] and [6] use bayesian networks and [26] combines neural networks with fuzzy representation of knowledge. Another category of ELEs is closer to the constructivist approach by allowing the learner to construct their own models rather than explore a "predefined" one. Compared to conventional learning environments (even environments that use simulations), this type of ELE requires approaches to learner modelling that would be able to capture and model the useful interactions that take place as learners construct their models.

In this paper, we present an approach to learner modelling in ELEs (suitable for both exploring simulations and constructing models) that combines case-based reasoning with other AI techniques. The subsequent section briefly introduces the application domain, namely mathematical generalisation, and the ELE used, called ShapeBuilder, and discusses the challenges involved in performing learner modelling. Section 3 presents a conceptual framework for the learner modelling process and describes the case-based formulation. Section 4 illustrates the process with an example, while Section 5 concludes the paper and outlines future work.

2 EXPLORATORY LEARNING FOR MATHEMATICAL GENERALISATION

Mathematical generalisation (MG) is associated with algebra, as "algebra is, in one sense, the language of generalisation of quantity. It provides experience of, and a language for, expressing generality, manipulating generality, and reasoning about generality" [20].

However, students do not associate algebra with generalisation as the algebraic language is perceived as been separate from what it represents [15]. To address this problem the ShapeBuilder [8] system, which is an ELE under development in the context of the MiGen project [2], aims to facilitate the correspondence between the models, patterns and structures (visual representations) that the learners build, on one hand, and their numeric, iconic and symbolic representations, on the other hand. ShapeBuilder allows the construction of different shapes [9], e.g. rectangles, L-shapes, T-shapes and supports the three types of representations aforementioned: (a) *numeric representations* that include numbers (constants or variables) and expressions with numbers; (b) *iconic representations* which correspond to icon variables; (c) *symbolic representations* that are names or symbols given by users to variables or expressions. An icon variable has the value of a dimension of a shape (e.g. width, height) and can be obtained by double-clicking on the corresponding edge of the shape. It is represented as an icon of the shape with the corresponding edge highlighted (see Figure 1a).

Constants, variables and numeric expressions lead to specific constructions/models, while icon variables and expressions using them lead to general ones. Through the use of icon variables, ShapeBuilder encourages structured algebra thinking, connecting the visual with the abstract (algebraic) representation, as "each expression of generality expresses a way of seeing" [20] (see Figure 1b). It also uses the "messing up" metaphor [12] that consists of asking the learner to resize a construction and observe the consequences; the model will "mess up" only if it is not general (see Figures 1c and d), indicating learner's lack of generalisation ability.

When attempting to model the learner in an ELE for such a wide domain as MG, several challenges arise. The main and widely ac-

[1] The authors are with the London Knowledge Lab, Birkbeck College, University of London, UK; email: {mihaela;gmagoulas}@dcs.bbk.ac.uk

[2] Funded by ESRC, UK, under TLRP e-Learning Phase-II (RES-139-25-0381); http://www.tlrp.org/proj/tel/tel_noss.html.

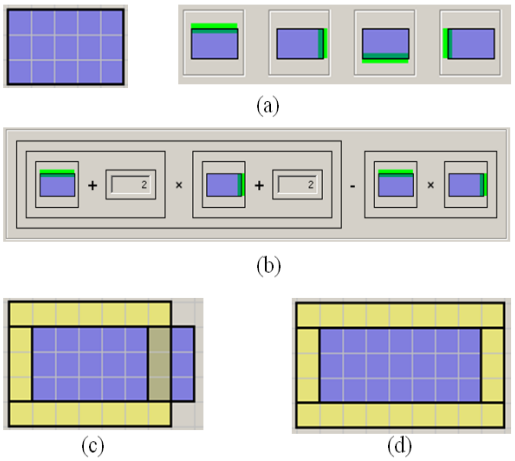


Figure 1. (a) A rectangular shape and its icon variable; (b) an expression using icon variables; (c) "messing up"; (d) general solution that does not "mess up".

knowledged challenge is to balance freedom with control: learners should be given enough freedom so that they can actively engage in activities but they should be offered enough guidance in order to assure that the whole process reflects constructivist learning and leads to useful knowledge [21]. This and some other challenges are illustrated in Table 1 with examples from the domain of MG.

Table 1. Applying learner modelling in ELEs for mathematical generalisation.

Challenge	Example
Balance between freedom and control	When a learner is trying to produce a general representation, for how long should he be left alone to explore and when does guidance become necessary?
What should be modelled?	Besides learner's knowledge of MG concepts (e.g. use of variables, consistency between representations, etc.), other aspects need to be modelled in order to support the learner during exploration: shapes constructed, relations between shapes, etc.
Do both correct and incorrect actions or behaviours have value?	In exploratory learning it is difficult to categorise actions or learner's explorations into "correct" and "incorrect". Moreover, actions that might lead to incorrect outcomes such as *resizing* can be more valuable for constructivist learning than "correct" actions.
Reasoning about abstract knowledge	Can consistency be inferred from the fact that a learner is checking the correspondence between various forms of representations? If so, is that always true? Are there any exceptions to this rule?
Underlying strategies	As it is neither realistic nor feasible to include all possible outcomes (correct or incorrect) to model the domain of MG, only key information with educational value could be stored, such as strategies in solving a task. The challenge is how to represent and detect them.

3 A CONCEPTUAL FRAMEWORK FOR LEARNER MODELLING

Given the challenges mentioned in Table 1 a conventional learner modelling approach does not fit the purposes of ELEs. Due to the exploratory nature of the activities and the diversity of possible trajectories, flexibility in the representation of information and handling of uncertainty are two important aspects for effectively supporting the learning process. As case-based reasoning offers flexibility of information representation and soft computing techniques handle uncertainty, a combination of the two is used. Moreover, previous research has proved the benefits of combining case-based reasoning with neural networks [23] and fuzzy quantifiers [30]. In the following subsections, the architecture of the system, the AI components and their role are described.

3.1 The Architecture

The architecture of the "Intelligent" ShapeBuilder is represented in Figure 2. As the learner interacts with the system through the interface, the actions of the learner are stored in the Learner Model (LM) and they are passed to the Interactive Behaviour Analysis Module (IBAM) where they are processed in cooperation with the Knowledge Base (KB); the results are fed into the LM. The Feedback Module (FM) is informed by the LM and the KB and feeds back to the learner through the interface.

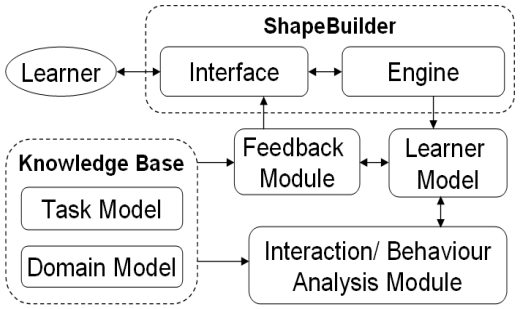


Figure 2. Schematic of an intelligent architecture for ShapeBuilder.

The KB includes two components (see Figure 2): a domain and a task model. The domain model includes high level learning outcomes related to the domain (e.g. using variables, structural reasoning, consistency, etc.) and considers that each learning outcome can be achieved by exploring several tasks. The task model includes different types of information: (a) strategies of approaching the task which could be correct, incorrect or partially correct; (b) outcomes of the exploratory process and solutions to specific questions associated with each (sub)task; (c) landmarks, i.e. relevant aspects or critical events occurring during the exploratory process; (d) contexts, i.e. reference to particular (sub)tasks.

The IBAM component combines case-based reasoning with soft computing in order to identify what learners are doing and be able to provide feedback as they explore a (sub)task. More specifically, as they are working in a specific subtask, which specifies a certain *context*, their actions are preprocessed, current cases are identified and matched to the cases from the Task Model (the case base). Prior to matching, local feature weighting [23] is applied in order to reflect the importance of the attributes in the current context.

In the FM component, multicriteria decision making [7] will be used to obtain priorities between several aspects that require feedback depending on the context.

3.2 Case-based Knowledge Representation

In case-based reasoning (CBR) [17] the knowledge is stored as cases, typically including the description of a problem and the corresponding solution. When a new problem is encountered, similar cases are

searched and the solution is adapted from one or more of the most similar cases.

Although CBR has been used successfully in applications for domains like legal reasoning [1], stock market prediction [5], recommender systems [19], and other areas, there is little research on using CBR for e-Learning environments. For example, [10] use CBR in the learner modelling process and call this approach case-based student modelling; while [13] use CBR and genetic algorithms to construct an optimal learning path for each learner. CBR is used also in [27] within a case-based instruction scenario rather than a method for learner modelling. We have not found any references in the literature to ELEs that use CBR or CBR combined with other intelligent methods.

The advantage of CBR for learning environments and especially for ELEs is that the system does not rely only on explicit representation of general knowledge about a domain, but it can also use specific knowledge previously experienced [10]. It also seems promising for improving the effectiveness of complex and unstructured decision making [13] in combination with other computing methods.

In our research, CBR is used in the learner modelling process. The cases contain information describing models that learners should construct using ShapeBuilder. Different strategies in approaching a problem (i.e. constructing a model to meet a particular learning objective) are represented as a series of cases that reflect possible exploratory trajectories of learners as they construct models during the various (sub-)tasks.

Table 2. The set of attributes (F_i) of a case.

Category	Name	Label	Possible Values
Shape	Shape type	α_{i_1}	Rectangle(/L-Shape/T-Shape)
Dimensions of shape	Width type	α_{i_2}	constant (c)/variable (v)/ icon variable (iv)/ numeric expression (n_exp)/ expression with iv(s) (iv_exp)
	Height type	α_{i_3}	c /v /iv /n_exp /iv_exp
	⋮	⋮	⋮
	Thickness type	α_{i_v}	c /v /iv /n_exp /iv_exp
	Width value	$\alpha_{i_{v+1}}$	numeric value
	Height value	$\alpha_{i_{v+2}}$	numeric value
	⋮	⋮	⋮
	Thickness value	α_{i_w}	c /v /iv /n_exp /iv_exp
Part of Strategy	$PartOfS_1$	$\alpha_{i_{w+1}}$	1
	$PartOfS_2$	$\alpha_{i_{w+2}}$	0
	⋮	⋮	⋮
	$PartOfS_r$	α_{i_N}	0

A case is defined as $C_i = \{F_i, RA_i, RC_i\}$, where C_i represents the case and F_i is a set of attributes. RA_i is a set of relations between attributes and RC_i is a set of relations between C_i and other cases respectively.

The set of attributes is represented as $F_i = \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_N}\}$. It includes three types of attributes: (a) numeric, (b) variables and (c) binary. Variables refer to different string values that an attribute can take, and binary attributes indicate whether a case can be considered in formulating a particular strategy or not. This could be represented as a "part of strategy" function: $PartOfS_u : C_i \rightarrow \{0, 1\}$,

$$PartOfS_u = \begin{cases} 1 & \text{if } C_i \in S_u \\ 0 & \text{if } C_i \notin S_u, \end{cases}$$

where S_u represents a strategy and is defined further on. The set of attributes of a generic case for ShapeBuilder is presented in Table 2. The first v attributes ($\alpha_{i_j}, j = \overline{1, v}$) are variables, the ones from $v+1$ to w are numeric ($\alpha_{i_j}, j = \overline{v+1, w}$) and the rest are binary ($\alpha_{i_j}, j = \overline{w+1, N}$).

The set of relations between attributes of the current case and attributes of other cases (as well as attributes of the same case) is represented as $RA_i = \{RA_{i_1}, RA_{i_2}, \ldots, RA_{i_M}\}$, where at least one of the attributes in each relation $RA_{i_m}, \forall m = \overline{1, M}$, is from the set of attributes of the current case F_i. Two types of binary relations are used: (a) a *dependency relation* (D_{i_s}) is defined as $(\alpha_{i_k}, \alpha_{j_l}) \in D_{i_s} \Leftrightarrow \alpha_{i_k} = DEP(\alpha_{j_l})$, where $DEP : \alpha_{i_k} \rightarrow \alpha_{j_l}$ for attributes α_{i_k} and α_{j_l} that are variables of cases i and j (where $i = j$ or $i \neq j$), and means that α_{i_k} depends on (is built upon) α_{j_l} (if $i = j$, $k \neq l$ is a condition as to avoid circular dependencies) (e.g. the width type of a case is built upon the height type of the same case; the width type of a case is built upon the width type of another case, an so on); (b) a *value relation* (V_{i_s}) is defined as $(\alpha_{i_k}, \alpha_{j_l}) \in V_{i_s} \Leftrightarrow \alpha_{i_k} = f(\alpha_{j_l})$, where α_{i_k} and α_{j_l} are numeric attributes and f is a function and could have different forms depending on context (e.g. the height of a shape is two times its width; the width of a shape is three times the height of another shape, etc.). The set of relations between attributes is presented in Table 3.

Table 3. The set of relations between attributes (RA_i) of cases.

Relation	Label	Example
Dependency relation	$D_{i_1}(RA_{i_1})$	$(\alpha_{i_k}, \alpha_{j_l}); k, l = \overline{2, v}; \forall j$
	⋮	⋮
	$D_{i_t}(RA_{i_t})$	$(\alpha_{i_k}, \alpha_{j_l}); k, l = \overline{2, v}; \forall j$
Value relation	$V_{i_1}(RA_{i_{t+1}})$	$(\alpha_{i_k}, \alpha_{j_l}); k, l = \overline{v+1, w}; \forall j$
	⋮	⋮
	$V_{i_z}(RA_{i_M})$	$(\alpha_{i_k}, \alpha_{j_l}); k, l = \overline{v+1, w}; \forall j$

The set of relations between cases is represented as $RC_i = \{RC_{i_1}, RC_{i_2}, \ldots, RC_{i_P}\}$, where one of the cases in each relation $RC_{i_j}, \forall j = \overline{1, P}$ is the current case (C_i). Two relations about order in time are defined: (a) $Prev$ relation indicates the previous case with respect to the current case: $(C_i, C_j) \in Prev$ if $t(C_j) < t(C_i)$ and (b) $Next$ relation indicates the next case with respect to the current case: $(C_i, C_k) \in Next$ if $t(C_i) < t(C_k)$. Each case includes at most one of each of these two relations ($p \leq 2$).

A strategy is defined as $S_u = \{N_u(C), N_u(RA), N_u(RC)\}$, $u = \overline{1, r}$, where $N_u(C_i)$ is a set of cases, $N_u(RA_i)$ is a set of relation between attributes of cases and $N_u(RC_i)$ is a set of relations between cases.

3.3 Comparing Cases, Exploiting Context and Modelling Learning Trajectories

In this section we present three distinctive features of the proposed framework: comparing cases, exploiting context and modelling of learning trajectories.

Comparing cases. The most common definition of similarity is a weighted sum of similarities of attributes of cases [17]:

$$S_{IR} = \frac{\sum_{i=1}^{N} o_i \times sim(f_i^I, f_i^R)}{\sum_{i=1}^{n} o_i},$$

where o_i represents the weight of each attribute, sim is a similarity function, and I and R stand for input and retrieved cases, respectively. In our case, four similarity measures are defined for comparing cases:

1. Euclidean distance is used for comparing numeric attributes: $D_{IR} = \sqrt{\sum_{j=v+1}^{w} o_j \times (\alpha_{I_j} - \alpha_{R_j})^2}$
2. The following metric is used for attributes that are variables: $V_{IR} = \frac{\sum_{j=1}^{v} g(\alpha_{I_j}, \alpha_{R_j})}{v}$, where g is defined as:

$$g(\alpha_{I_j}, \alpha_{R_j}) = \begin{cases} 1 & \text{if } \alpha_{I_j} = \alpha_{R_j} \\ 0 & \text{if } \alpha_{I_j} \neq \alpha_{R_j}, \end{cases}$$

3. In a similar way to [25], we define the following metric for comparing relations between attributes: $P_{IR} = \frac{|RA_I \cap RA_R|}{|RA_I \cup RA_R|}$. P_{IR} is the number of relations between attributes that the input and retrieved case have in common divided by the total number of relations between attributes of the two cases.
4. Similarity in terms of relations between cases is defined by $T_{IR} = \frac{|RC_I \cap RC_R|}{|RC_I \cup RC_R|}$, where T_{IR} is the number of relations between cases that the input and retrieved case have in common divided by the the total number of relations between cases of I and R.

In order to identify the closest strategy to the one employed by a learner, cumulative similarity measures are used for each of the four types of similarity:

1. Numeric attributes: $(\sum_{i=1}^{z} D_{I_i R_i})/z$.
2. Variables: $(\sum_{i=1}^{z} V_{I_i R_i})/z$.
3. Relations between attributes: $(\sum_{i=1}^{z} P_{I_i R_i})/z$.
4. Relations between cases. $(\sum_{i=1}^{z} T_{I_i R_i})/z$.

where z represents the minimum number of cases among the two compared strategies. The strength of similarity between the current strategy and the various stored strategies is defined as the maximum combined similarity of these four measures among the various strategies compared.

Exploiting context. Attributes and relations stored in cases have different relevance depending on the context, which in ShapeBuilder corresponds to different stages of the constructivist learning process that learners go through as they explore the various sub-tasks within a learning activity. Typically, a task includes several sub-tasks, and the activity is sequenced within the system so as to know at any time the current context. As the environment allows the learners to explore, they may "jump" to different stages in the activity sequence.

Context dependence can be taken into account by having different weights for attributes and relations depending on the stage of the learning process within a task or activity. The weights could be obtained through an approach called *local feature weighting* [23] that uses Neural Networks (NNs). The principle of the training algorithm is to reduce the distance between cases of the same class and increase the distance between cases of different classes [23], where the various classes in ShapeBuilder correspond to types of context (stages of the learning process) of the various (sub-)tasks. Thus, a neural network is trained in order to identify the context and several networks (one for each context) are used to provide the context-specific weights. This approach appears to be more robust than other weighting schemes due to the generalisation capacities of the NNs that can produce weights even in imprecise situations [23].

Learning trajectories. A string of cases connected with relations in time yields a knowledge structure that represents learner's explorations/learning trajectory in the ELE during a task or sub-task. Such a learning trajectory is constructed by successively applying $Prev$ and $Next$ relations to C_i in order to get cases previous in time to C_i and cases following C_i, respectively. Comparing trajectories in the KB to the current trajectory (this is useful to provide support and decide on scaffolding techniques) is done in two stages: comparing the past and evaluating the future.

Comparison of the past with respect to a reference point (e.g. a selected case) depends on the depth of the evaluation in terms of samples taken into account and rules than concern comparisons of the past, e.g. IF the actual trajectory is similar to a trajectory in the KB, indicated by a reference case representing a starting point in the past, THEN this trajectory is a past-matching trajectory.

When it comes to evaluating the future of a trajectory, comparison is based on the similarity between the future of a trajectory in the KB with a desired future for the current trajectory. This is expressed by rules of the general form: IF a piece of the future trajectory of a past-matching trajectory resembles the reference starting from a selected case, THEN the reference can be met by applying certain strategies.

As it is not possible to represent all learning trajectories in the KB of an ELE, similarity is measured in terms of convex fuzzy sets, whose width might change depending on the context and the amount of information available, i.e. the current trajectory can be interpreted in more vague way by increasing the width of the fuzzy set. Also if the distance between past and future is large for certain tasks, it does not make sense to evaluate the future carefully. Nevertheless, if the distance to a reference (desired outcome) is small, the future needs to be evaluated accurately. So the depth of the evaluation is measured by a fuzzy time distance set to evaluate both short and long time distances.

4 AN ILLUSTRATIVE EXAMPLE

To illustrate the combination of intelligent methods for learner modelling we use an example from the mathematical generalisation domain, and a task called "pond tiling", which is common in the English school curriculum and expects learners to produce a *general expression* for finding out how many tiles are required for surrounding any rectangular pond [8]. The high level learning objective in the Domain Model is to acquire the ability to perform structural reasoning [9]. In order to achieve this, sub-tasks can be explored in ShapeBuilder, e.g. construct a pond of fixed dimensions, surround the pond with tiles and determine how many are required; generalise the structure using icon variables.

Knowledge representation. The Task Model for pond tiling includes: (a) strategies identified in pilot studies [9], e.g. thinking in terms of areas (see Figure 3a) or in terms of width and height (see Figures 3b, c, d, e and f); (b) outcomes, e.g. model built, number of tiles for surrounding a particular pond, and solution, i.e. the general expression (see Figure 3 for the solutions corresponding to each strategy; for the "area strategy" the solution with icon variables is displayed in Figure 1b); (c) landmarks, e.g. for the area strategy: creating a rectangle with height and width greater by 2 than the pond; for the width and height strategies: using rows/column of tiles; slips: several correct actions followed by an incorrect one (e.g. correct surrounding of the pond, partially correct expression, but missing a 2 in the formula); (d) the context of each (sub-)task.

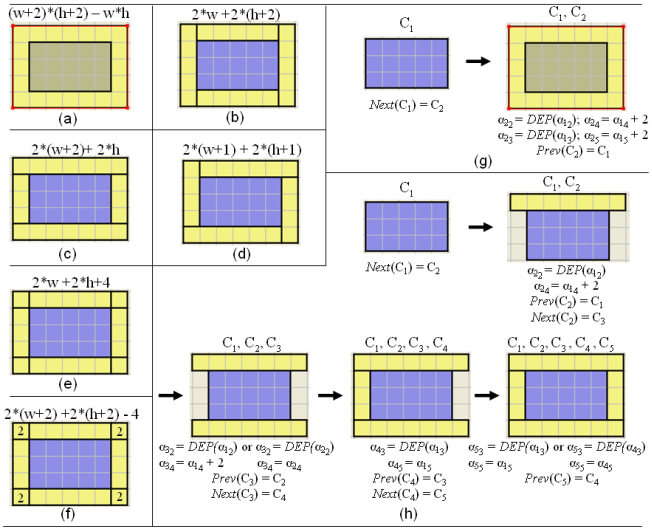


Figure 3. (a) "Area strategy"; (b) "H strategy"; (c) "I strategy"; (d) "Spiral strategy"; (e) "+4 strategy"; (f) "−4 strategy"; (g) Steps and relations of "area strategy"; (h) Steps and relations of "I strategy".

The six strategies and their associated solutions (the general expressions for surrounding any rectangular pond) are displayed in Figures 3(a–f). Two strategies are presented in detail: the "area strategy" (S_1) and the "I strategy" (S_3). The attributes of cases that are part of these two strategies are presented in Table 4 and Table 5, respectively. The steps and the sets of relations between attributes and between cases are displayed in Figure 3g and Figure 3h, respectively.

A particular order between cases is presented for the "I strategy" in Figure 3h. For the same strategy, the surrounding of the pond could be done in several other different orders; there are $4! = 24$ such possibilities (the pond is always first).

Table 4. The set of attributes (F_i) for the cases in the "area strategy".

Name	Label	C_1	C_2
Shape type	α_{i_1}	Rectangle	Rectangle
Width type	α_{i_2}	c/v/n_exp	iv/iv_exp
Height type	α_{i_3}	c/v/n_exp	iv/iv_exp
Width value	α_{i_4}	5	7
Height value	α_{i_5}	3	5
$PartOfS_1$	α_{i_6}	1	1
⋮	⋮	⋮	⋮
$PartOfS_2$	α_{i_7}	1	0
$PartOfS_6$	α_{i_8}	1	0

There are two types of strategies depending on the degree of generality: specific and general. Specific cases refer to surroundings that cannot be generalised and include value relations, but no dependency relations; the general cases refer to surroundings that can be generalised and are distinguished by the presence of the dependency relations and by the fact that the dimension type of at least one of the dimensions of the case is an icon variable or an expression using icon variable(s). The presence or absence of the abovementioned aspects apply to all cases that form the composite case with the exception of the first case representing the pond. The "area" and the "I strategy" presented previously fall into the category of general strategies.

The strategies displayed in Figure 3 are correct symmetrical "elegant" solutions, but trials with pupils have shown that not all of them

Table 5. The set of attributes (F_i) for the cases in the "I strategy".

Label	C_1	C_2	C_3	C_4	C_5
α_{i_1}	Rectangle	Rectangle	Rectangle	Rectangle	Rectangle
α_{i_2}	c/v/n_exp	iv /iv_exp	iv /iv_exp	c/v/n_exp	c/v/n_exp
α_{i_3}	c/v/n_exp	c/v/n_exp	c/v/n_exp	iv /iv_exp	iv /iv_exp
α_{i_4}	5	7	7	1	1
α_{i_5}	3	1	1	3	3
α_{i_6}	1	0	0	0	0
α_{i_7}	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮
α_{i_8}	1	0	0	1	1

use this type of approach [8, 9]. Some pupils surround the pond in a non–systematic manner and with variable degrees of symmetry. Such examples are illustrated in Figure 4.

Comparing cases. To illustrate the operation of similarity measures we use two non–symmetrical examples of surrounding the pond, displayed in Figure 4. The similarity measures are the ones presented in Section 3.3.

The first example (Figure 4a), has 4 cases in common with two strategies: the "I strategy" (C_1, C_3, C_4, C_5) and the "+4 strategy" (C_1, C_4, C_5, C_6). When comparing it with the "I strategy" $z = 5$ (minimum between 6 and 5) and the combined similarity is: $\frac{\sqrt{1}}{5} + \frac{5}{5} + \frac{7/4}{5} + \frac{10/4}{5} = 2.05$. When comparing with the "+4" strategy, $z = 6$ (minimum between 6 and 9) the combined similarity is: $\frac{\sqrt{5}}{6} + \frac{5+2/3}{6} + \frac{6/4}{6} + \frac{10/4+1/3}{6} = 2.04$. Thus, in this case the learner will be guided towards the "I strategy".

The second example (Figure 4b), has 3 cases in common with two strategies: the "spiral strategy" (C_1, C_3, C_4) and the "H strategy" (C_1, C_2, C_5). When comparing it with the "spiral strategy" as well as the "H strategy", $z = 5$ (minimum between 5 and 5), and the combined similarity is: $\frac{\sqrt{2}}{5} + \frac{4+2/3}{5} + \frac{8/4}{5} + \frac{10/4}{5} = 2.12$. In this situation, when the learner's construction is equally similar to two strategies, the following options could be offered: (a) present the learner with the two options and let him/her choose one of the two (an approach that appears more suitable for advanced learners than novices); (b) automatically suggest one of the two in a systematic way, e.g. present the one that occurs more/less often with other learners; (c) inform the teacher about the learner's trajectory and the frequency of strategies and let him/her decide between the two.

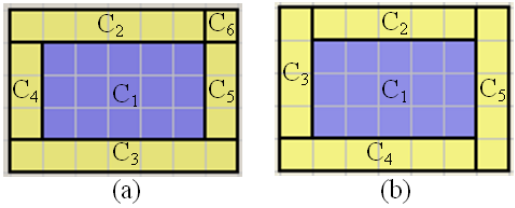


Figure 4. Non-symmetrical strategies: (a) combination of 'I' and '+4' strategies; (b) combination of 'spiral' and 'H' strategies.

Exploiting context. In the pond tiling task, when the learner is constructing a specific (as opposed to general) tiling of the pond, the value relation attribute is more relevant, while when dealing with a general tiling, the dependency relation attribute is more important.

Local feature weighting in this case involves two trained neural networks for each context and applying the weights delivered by the NNs before the matching process.

Learning trajectories. Lets consider the example in Figure 4b and a comparison after C_3. The current trajectory includes the creation of 3 rectangles corresponding to C_1, C_2 and C_3; this trajectory is considered to be far from the desired outcome (surrounding the pond), and thus, the future does not need to be evaluated accurately. At this point with respect to the past, two strategies partially match the learner's current trajectory: "I" (C_1, C_2) and "spiral" (C_1, C_3) strategy; the learner could be left to continue with his/her model construction without intervention. With respect to the future, the desired outcome can be obtained by following one of the two strategies previously identified.

If the comparison takes place after C_5, the trajectory would include the creation of 5 rectangles (C_1 to C_5) and thus it can be concluded that the learner has reached the desired outcome of surrounding the pond. However, in this process the learner did not use any of the desirable strategies, i.e. any of the six strategies presented in Figure 3. At this point in time two trajectories match the past and indicate the future, as before, but now it might be considered pedagogically important to intervene and guide the learner towards a trajectory that corresponds to one of the two identified desirable strategies.

5 CONCLUSIONS

In this paper a learner modelling process involving a combination of intelligent methods was presented for the domain of mathematical generalisation. Case-based reasoning is used in combination with soft computing (fuzzy sets and neural networks) in order to process the models that the learners construct and thus be able to provide feedback while learners work on the task.

Further work includes expanding the conceptual framework by defining a strength as the maximum combined similarity measure (similarity of the past and similarity of the future at a particular distance) for various evaluated trajectories and a reliability index that will reflect the extent to which the similarities can be relied upon to provide the right support.

REFERENCES

[1] V. Aleven, 'Using background knowledge in case-based legal reasoning: A computational model and an intelligent learning environment', *Artificial Intelligence*, **150**(1-2), 183–237, (2003).

[2] J.E. Beck and B.P. Woolf, 'Using a Learning Agent with a Student Model', in *Intelligent Tutoring Systems*, eds., B.P. Goettl, H.M. Halff, C.L. Redfield, and V.J. Shute, volume 1452 of *Lecture Notes in Computer Science*, pp. 6–15. Springer, (1998).

[3] P. Brusilovsky, A.T. Corbett, and F. de Rosis, eds. *User Modeling 2003, 9th International Conference, UM 2003, Johnstown, PA, USA, June 22-26, 2003, Proceedings*, volume 2702 of *Lecture Notes in Computer Science*. Springer, 2003.

[4] A. Bunt and C. Conati, 'Probabilistic Student Modelling to Improve Exploratory Behaviour', *User Modelling and User-Adaptive Interaction*, **13**(3), 269–309, (2003).

[5] S-H. Chun and Y-J. Park, 'Dynamic adaptive ensemble case-based reasoning: application to stock market prediction', *Expert Systems with Application*, **28**(3), 435–443, (2005).

[6] C. Conati, A.S. Gertner, and K. VanLehn, 'Using Bayesian Networks to Manage Uncertainty in Student Modeling', *User Modelling and User-Adaptive Interaction*, **12**(4), 371–417, (2002).

[7] G. Ghinea, G.D. Magoulas, and C. Siamitros, 'Multicriteria decision making for enhanced perception-based multimedia communication', *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, **35**(6), 855–866, (2005).

[8] S. Gutiérrez, M. Mavrikis, and D. Pierce, 'A Learning Environment for Promoting Structured Algebraic Thinking in Children', in *Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies*, (2008). forthcoming.

[9] S. Gutiérrez, D. Pierce, E. Geraniou, and M. Mavrikis, 'Supporting Reasoning and Problem-Solving in Mathematical Generalisation with Dependency Graphs', in *Proceedings of the 5th International Conference on the Theory and Application of Diagrams*, (2008). forthcoming.

[10] S-G. Han, S-G. Lee, and S. Jo, 'Case-based tutoring systems for procedural problem solving on the www', *Expert Systems with Applications*, **29**(3), 573–582, (2005).

[11] I. Hatzilygeroudis and J. Prentzas, 'Using a hybrid rule-based approach in developing an intelligent tutoring system with knowledge acquisition and update capabilities', *Expert Systems with Applications*, **26**(4), 477–492, (2004).

[12] L. Healy, Hoelzl R., Hoyles C., and R. Noss, 'Messing Up', *Micromath*, **10**(1), 14–16, (1994).

[13] M-J. Huang, H-S. Huang, and M-Y. Chen, 'Constructing a personalized e-learning system based on genetic algorithm and case-based reasoning approach', *Expert Systems with Applications*, **33**(3), 551–564, (2007).

[14] C.D. Hulshof, T.H.S. Eysink, S. Loyens, and T. de Jong, 'Zaps: Using interactive programs for learning psychology', *Interactive Learning Environments*, **13**, 39–53, (2005).

[15] J. Kaput, *Technology and Mathematics education*, 515–556, D. Grouws (ed.) Handbook of Research on Mathematics Teaching and Learning, New York: Macmillan, 1992.

[16] P. Kirschner, J. Sweller, and R.E. Clark, 'Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential and inquiry-based teaching', *Educational Psychologist*, **41**(2), 75–86, (2006).

[17] J.L. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann Publishers, Inc., 2nd edn., 1993.

[18] C. Koutsojannis, G. Beligiannis, I. Hatzilygeroudis, C. Papavlasopoulos, and J. Prentzas, 'Using a hybrid ai approach for exercise difficulty level adaptation', *International Journal of Continuing Engineering Education and Life-Long Learning*, **17**(4-5), 256–272, (2007).

[19] P. Kumar, S. Gopalan, and V. Sridhar, 'Context enabled multi-CBR based recommendation engine for e-commerce', in *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE)*, pp. 237 – 244. IEEE Press, (2005).

[20] J. Mason, *Generalisation and algebra: Exploiting children's powers*, 105–120, L. Haggarty, Aspects of Teaching Secondary Mathematics: Perspectives on Practice, Routledge Falmer and the Open University, 2002.

[21] R. Mayer, 'Should There Be a Three-Strikes Rule Against Pure Discovery Learning? The Case for Guided Methods of Instruction', *American Psychologist*, **59**(1), 14–19, (2004).

[22] R. Morales Gamboa, *Exploring Participative Learner Modelling and Its Effects on Learner Behaviour*, PhD thesis, Univ. of Edinburgh, 2000.

[23] J.H. Park, K.H. Im, C-K. Shin, and S-C. Park, 'MBNR: Case-Based Reasoning with Local Feature Weighting by Neural Network', *Applied Intelligence*, **21**(3), 265–276, (2004).

[24] C. Romero, S. Ventura, P. de Bra, and C. de Castro, 'Discovering Prediction Rules in AHA! Courses', In Brusilovsky et al. [3], pp. 25–34.

[25] Y. Seo, D. Sheen, and T. Kim, 'Block assembly planning in shipbuilding using case-based reasoning', *Expert Systems with Applications*, **32**(1), 245–253, (2007).

[26] R. Stathacopoulou, M. Grigoriadou, G.D. Magoulas, and D. Mitropoulos, 'A Neuro-fuzzy Approach in Student Modeling', In Brusilovsky et al. [3], pp. 337–341.

[27] R.H. Stottler and S. Ramachandran, 'A Case-Based Reasoning Approach to Internet Intelligent Tutoring Systems (ITS) and ITS Authoring', in *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, pp. 181–186. AAAI Press, (1999).

[28] J. Swaak, W.R. van Joolingen, and T. de Jong, 'Supporting simulation-based learning; the effects of model progression and assignments on definitional and intuitive knowledge', *Learning and Instruction*, **8**, 235–253, (1998).

[29] W.R. van Joolingen, 'Cognitive tools for discovery learning', *International Journal of Artificial Intelligence and Education*, **10**, 385–397, (1999).

[30] R.R. Yager, 'Soft Aggregation Methods in Case Based Reasoning', *Applied Intelligence*, **21**(3), 277–288, (2004).

Belief Propagation in Fuzzy Bayesian Networks

Christopher Fogelberg[1] and **Vasile Palade** and **Phil Assheton**[2]

Abstract. Fuzzy Bayesian networks are a generalisation of classic Bayesian networks to networks with fuzzy variable state. This paper describes our formalisation and outlines how belief propagation can be conducted. Fuzzy techniques can lead to more robust inference. A key advantage of our formalisation is that it can take advantage of all existing network inference and Bayesian network algorithms. Another key advantage is that we have developed several techniques to control the algorithmic complexity. When these techniques can be applied it means that fuzzy Bayesian networks are only a small linear factor less efficient than classic Bayesian networks. With appropriate pre-processing they may be substantially more efficient.

1 Introduction

Modern machine learning research frequently uses *Bayesian networks* (BNs)[6; 7; 15; 16]. However, BN inference is NP-complete due to cycles in the undirected graph[4], and belief propagation is exponential in the *tree-width* of the network. This makes them difficult to use for large problems.

Fuzzy[3] and hybrid fuzzy systems[11; 13] are also frequently used. In a fuzzy system, a variable's *state* is represented by a set of *fuzzy values* (FVs). Because fuzzy systems do not force a model to artificially discretise a continuous underlying state they are often more robust in the face of noise.

To date there has been very little research into BNs with fuzzy variable states. What there is has centred around the use of fuzzy approximations to perform inference and belief propagation in a *hybrid BN*[1; 12]. A hybrid BN is one where the parameters are a mix of continuous and multinomial distributions.

This paper's key contribution is a formal generalisation of classic BNs to *fuzzy Bayesian networks* (FBNs). In a FBN the variables can have *fuzzy states*. The paper also describes tractable belief propagation over FBNs. An important advantage of the presented formalisation is that existing inference algorithms (e.g. MCMC, simulated annealing) can be used without modification. Furthermore, FBNs may be only a small linear constant less efficient than classic BNs of the same size, and appropriate pre-processing may make some problems which were intractable for classic BNs tractable for FBNs.

The paper is structured as follows. Section 2 presents a fuzzy Bayesian network which will be used as an example. Section 3 introduces some notation (subsection 3.1), and presents belief propagation for variables with one parent (subsection 3.3) and for variables with multiple parents (subsection 3.4). Section 4 analyses the algorithmic efficiency of FBNs and how it can be controlled. Section 5 outlines an important bioinformatic domain where FBNs may be especially useful, and section 6 concludes the paper.

[1] Supported by the ACU and CSCUK

[2] Computing Laboratory, University of Oxford; Contact email: christopher.fogelberg@comlab.ox.ac.uk

2 A Fuzzy Bayesian Network

The structure of the FBN that is used as an example in this paper is shown in figure 1. Call this FBN $G = \langle \eta, \theta \rangle$, where η denotes the structure of G and θ its parameters.

For clarity of presentation, G is a multinomial (discrete) BN. However, the formalisation generalises easily and transparently to continuously-valued FBNs and hybrid FBNs.

The relevant conditional distributions of G are shown in figure 2. D's distribution is not shown and we will later assume a state for D with no loss of generality.

Because we have restricted the differences between BNs and FBNs to belief propagation, the specification of a FBN and a BN are identical.

3 Belief Propagation

Belief propagation in a Bayesian network involves calculating the updated probability distributions of variables in the network, given θ and the observed states of other variables.

3.1 Some Notation

The terminology and notation is as follows. A variable has a state, either a fuzzy state (FS) or a *discrete state* (DS).

A fuzzy state is made up of one or more *components*, and each component is annotated with the variable's degree of membership (μ) in that component. For example, equation 1 is an example of a variable (S) with two components. It has membership 0.7 in the component *hi* and membership 0.3 in the component *mid*. *hi* and *mid* are examples of *values* that the variable can take. When annotated with μ they are referred to as fuzzy values (FV). The set of all possible values (fuzzy values) that a variable can take is the *range* of that variable, e.g. *hi*, *mid*, *lo*.

$$S = [hi_{0.7}, mid_{0.3}] \tag{1}$$

In general, the components of a variable's state are enclosed in square brackets. A discrete state is just a special case of a fuzzy state. Discrete states have just one component with $\mu = 1$, and the square brackets and μ subscript can be omitted in this situation. We assume that $\sum_{c \in C} \mu_c = 1$ for a FS with C components and have not considered other situations.

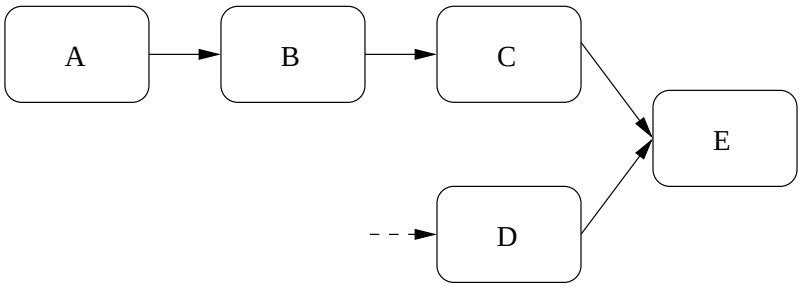


Figure 1. The fuzzy Bayesian network G, used as an example in this paper.

$\rightarrow A$	$A = lo$	$A = mid$	$A = hi$
	0.7	0.1	0.2

(a) θ_A, A's prior distribution

$A \rightarrow B$	$B = lo$	$B = mid$	$B = hi$
$A = lo$	0.6	0.2	0.2
$A = mid$	0.1	0.1	0.8
$A = hi$	0.1	0.2	0.7

(b) θ_B, B's conditional distribution

$B \rightarrow C$	$C = lo$	$C = mid$	$C = hi$
$B = lo$	0.1	0.1	0.8
$B = mid$	0.1	0.8	0.1
$B = hi$	0.7	0.2	0.1

(c) θ_C, C's conditional distribution

$C, D \rightarrow E$		$E = lo$	$E = mid$	$E = hi$
$C = lo$	$D = lo$	0.6	0.2	0.2
$C = lo$	$D = mid$	0.1	0.1	0.8
$C = lo$	$D = hi$	0.1	0.1	0.8
$C = mid$	$D = lo$	0.6	0.2	0.2
$C = mid$	$D = mid$	0.1	0.6	0.3
$C = mid$	$D = hi$	0.1	0.6	0.3
$C = hi$	$D = lo$	0.1	0.2	0.7
$C = hi$	$D = mid$	0.1	0.2	0.7
$C = hi$	$D = hi$	0.8	0.1	0.1

(d) θ_E, E's conditional distribution

Figure 2. θ for G. The conditional distributions of A, B, C and E.

Just as a component can be a value from the range of a variable, e.g. hi, a component can also be a probability distribution (PD). PD are denoted with curly brackets, e.g. $\{hi_{0.3}, mid_{0.2}, lo_{0.5}\}$. Because the value associated with each subscripted probability is implicit in the tuple order the value names can be omitted: $\{0.3, 0.2, 0.5\}$.

An example of a fuzzy state which mixes values and probability distributions is shown in equation 2.

$$T = [\{0.2, 0.1, 0.7\}_{0.2}, \{0.1, 0.8, 0.1\}_{0.6}, mid_{0.2}] \qquad (2)$$

A PD which is annotated (subscripted) with μ is called a *fuzzy probability distribution* (FPD). Samples are drawn from a FPD in the same way that they are drawn from a PD. However, a variable with membership μ in a FPD can only have μ proportion of its state determined by that FPD; a sample from a FPD will have the same μ as the FPD does. For example, a sample from $\{0.2, 0.1, 0.7\}_{0.2}$ will be one of $lo_{0.2}$, $mid_{0.2}$ or $hi_{0.2}$, and each of these components will be drawn with probability 0.2, 0.1 and 0.7 respectively.

This means that the state of a sample from the uncertain variable T (equation 2 will be some member from the set $[lo_{[0..0.8]}, mid_{[0.2..1]}, hi_{[0..0.8]}]$ and the distribution over members in this set is determined by the two FPD and one FV which make up the fuzzy state of T.

3.2 Assumptions

The full and general analysis of FBNs would also consider unrestricted interactions amongst components in a fuzzy state, allowing $\sum \mu \neq 1$ and so forth. In this article we make a number of linearising assumptions which make FBN belief propagation cheap, relative to the cost of full general propagation. They also greatly aid the clarity of the presentation in the space available. Furthermore, these assumptions are reasonable and do not restrict the general utility of FBNs. However it is important to make them explicit. The assumptions (and consequently the nature of full general propagation) will be briefly summarised in this section; a more general discussion is forthcoming.

3.2.1 Assumption: Total Membership

As noted above and in subsection 3.1, we assume that $\sum \mu = 1$. This is our first linearising assumption, and it can be conceptualised as follows. A variable's degree of membership in each of its $|C|$ components forms a $|C|$-dimensional *fuzzy state space*. If the variable has no uncertainty (no component is an FPD) then $|C|$ will be the same as the variable's range. Even if a FS has 0 membership in some of its range those values are still part of the state's FSS. By assuming that $\sum_{c \in C} \mu_c = 1$, we restrict our attention to a smaller $|C| - 1$ dimensional subspace. This subspace constrains the degrees of membership in each component in a cyclically conditional way on the degrees of membership in the other components. This assumption simplifies the combination of components of fuzzy states in subsection 3.4.

For example, if a FS has membership 0.5 in the value hi its membership in the values lo and mid in the FSS are constrained to be in the range $[0, 0.5]$. Furthermore, its membership in lo and mid mutually constrain (in this case define, as

there are no other values in the range) each other. Figure 3 illustrates the impact of the first assumption on the FSS for a fuzzy state with two components.

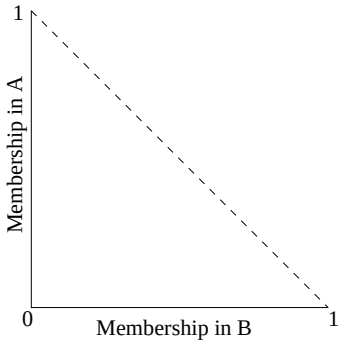


Figure 3. Imagine a fuzzy state with two components, A and B. Such a state would have a two-dimensional FSS, as in this figure. Without restriction, the state's degree of membership in each component could be specified by any point in the FSS. However, we assume that $\sum_{c \in C} \mu_c = 1$. Therefore its membership in components A and B must be specified by some point on the dashed line.

3.2.2 Assumption: Component Independence

We also make two further assumptions about FBN during belief propagation. The first is that components are independent. This means that when a variable has only one parent then its state will have one component for each component its parent has, and these components will have the same μ as the corresponding parent's component. For example, the children which have S (equation 1) as their only parent will have two components in their FS, one with $\mu = 0.7$ and one with $\mu = 0.3$.

When a variable has more than one parent then the components of the parents will be mixed and combined before propagation. This is described in subsection 3.4. Because we assume independence, the child's fuzzy state will have one component for each component in the mixed and combined parent set, and each of the child's component will have the same μ as the corresponding component in the parent set.

It will also be clear that we assume independence when we describe how the parents' components are mixed and combined in subsection 3.4.

3.2.3 Assumption: FPD Samples

The third assumption implicit in this model is the assumption that a sample from an FPD with $\mu = x$ will be a single fuzzy component with $\mu = x$. Although natural and intuitive we do not believe that this is automatically entailed, thus we explicitly assume it.

Consider an uncertain variable with a range of r in a standard (discrete) Bayesian network. Its state will be a probability distribution which specified a single point in the r dimensional *probability space* (p-space).

Now consider this variable in a FBN. Any uncertainty in its state will be represented by an FPD component with some μ. As described in subsection 3.1, a FPD is just PD with a fixed (0 dimensional) μ associated with it.

This definition of a FPD could be generalised so that μ could vary independently but was fixed for each of the r possible samples that could be drawn from the FPD. Call this a *slightly general FPD* (SGFPD) and call the FPD defined in subsection 3.1 a *standard FPD*. An SGFPD would specify a single point in an $r + r$ dimensional space, where r of the dimensions are the probability of each value and the other r dimensions specify the μ of a sample of each value. Just as the first r dimensions specify a p-space, the second r dimensions specify a *μ-space*. An example of such a space is given diagrammatically in figure 4, and it is used to contrast a SGFPD with a standard FPD.

An example of an SGFPD might be "there is a 0.2 probability of drawing a sample of *hi*, and any sample of *hi* will have $\mu = 0.3$, and there is a 0.3 probability of drawing a sample of *mid*, and any sample of *mid* will have $\mu = 0.5$, and..." and so forth. The assumption that $\sum_{c \in C} \mu_c = 1$ for a state could be relaxed if SGFPD were used.

After considering figure 4 it will be clear that SGFPD could be further generalised so that the μ of any sample also varied probabilistically, conditional on the value (*hi*, *mid*, etc.) of the sample. Such a *general FPD* (GFPD) would be an $r + r$ dimensional probability distribution over the joint μ and range of the variable. We believe that this represents the most general kind of inference and belief propagation in a FBN. Such inference is intractable and we do not consider it in this paper.

In summary, the assumptions which we have made substantially reduce the dimensionality of belief propagation and are necessary for it to be tractable. However, more general FBNs with GFPD do not have these restrictions; their utility will be considered in a forthcoming publication.

3.3 Single-Parent Belief Propagation

Assume that observations indicate $A = [mid_{0.2}, hi_{0.8}]$ in G. With this information we can calculate the updated distributions on B and C.

Because A has an observed (certain) FS and is B's only parent the components of B's updated FS can be read from θ_B. This shows that:

$$B = [\{0.1, 0.1, 0.8\}_{0.2}, \{0.1, 0.2, 0.7\}_{0.8}] \tag{3}$$

The FS over C is calculated similarly. Just as each of the fuzzy values in A lead to a weighted FPD in the FS of B the same occurs for C, and $C = [\alpha_{0.2}, \beta_{0.8}]$. The weighted distributions α and β are calculated using standard BN belief propagation, based on the conditional distribution of B. This is shown in equations 4, 5 and 6.

$$\begin{aligned} p(C|B = lo) &= \{0.1, 0.1, 0.8\} \\ p(C|B = mid) &= \{0.1, 0.8, 0.1\} \\ p(C|B = hi) &= \{0.7, 0.2, 0.1\} \end{aligned} \tag{4}$$

$$\begin{aligned} \alpha &= \{0.1, 0.1, 0.8\} \times 0.1 + \{0.1, 0.8, 0.1\} \times 0.1 + \\ &\qquad \{0.7, 0.2, 0.1\} \times 0.8 \\ &= \{0.01, 0.01, 0.08\} + \{0.01, 0.08, 0.01\} + \\ &\qquad \{0.56, 0.16, 0.08\} \\ &= \{0.58, 0.25, 0.17\} \end{aligned} \tag{5}$$

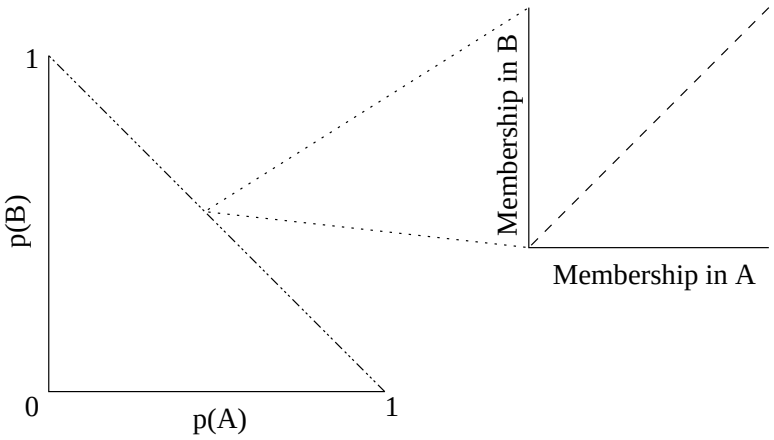


Figure 4. Assume a variable with a range (r) of 2 (A and B). Each point in the 2 dimensional p-space (left hand side) could be considered an index into a 2 dimensional μ-space (right hand side), as diagrammed. All proper probability distributions ($\sum p = 1$) fall on the dotted dashed line in the p-space. The μ-space that is indexed by some point in the p-space could be unique to that point. In a standard FPD, the μ-space is reduced to a single point on the dashed line and that point on the line in the μ-space is specified precisely by the μ of the FPD. Because any sample from a standard FPD, regardless of its value, will have the same μ, r of the dimensions are eliminated. In addition, we assume that an FPD has a proper probability distribution. In total, these reduces the dimensionality from $r + r$ to $r - 1$. In a SGFPD the μ-space would also be reduced to a single point, but any point in the μ-space would be a valid reduction, so an SGFPD with a proper distribution over the values still has $r + r - 1$ dimensions.

$$\begin{aligned}\beta &= \{0.1, 0.1, 0.8\} \times 0.1 + \{0.1, 0.8, 0.1\} \times 0.2 + \\ &\quad \{0.7, 0.2, 0.1\} \times 0.7 \\ &= \{0.01, 0.01, 0.08\} + \{0.02, 0.16, 0.02\} + \\ &= \{0.49, 0.14, 0.07\} \\ &= \{0.52, 0.31, 0.17\} \end{aligned} \quad (6)$$

The calculated FS for C is shown in equation 7.

$$C = [\{0.58, 0.25, 0.17\}_{0.2}, \{0.52, 0.31, 0.17\}_{0.8}] \quad (7)$$

3.4 Multi-Parent Belief Propagation

Subsection 3.3 illustrated belief propagation in a FBN when a variable has only one parent. This subsection shows *naive FBN belief propagation* in the case of a variable with multiple parents. Section 4 outlines several more nuanced approaches which address the problems with naive propagation.

Take the calculated value of C, and assume a fuzzy state for D (equation 8). What is the updated fuzzy state of E?

$$\begin{aligned} C &= [\{0.58, 0.25, 0.17\}_{0.2}, \{0.52, 0.31, 0.17\}_{0.8}] \\ D &= [\{0.45, 0.30, 0.25\}_{0.3}, \{0.1, 0.8, 0.1\}_{0.7}] \end{aligned} \quad (8)$$

Any *combination* of components, one from each parent, can be used to calculate an updated probability distribution for a variable. However, this raises the question of how to combine and weight each combination of component distributions in the parent FSs to calculate an updated FS for the child.

Because the parents are conditionally independent given the variable being updated[3], any particular combination of PD and observations can be summed over, as one was in each of equations 5 and 6. The summed over combinations became components of C's updated distribution.

[3] And also given their updated state and the acyclic nature of the graph

In the naive approach to belief propagation the Cartesian product of the parents' FS is used to find all possible combinations of components. μ for each one of these combinations is calculated using the *product t-norm*[2]. Any other fuzzy conjunction (normalising μ where necessary) could also be used. Because we assumed that $\sum \mu = 1$ holds for each of the parents though, using this fuzzy conjunction guarantees that $\sum \mu$ over the child's components will also equal 1 and no normalisation is necessary.

For example, if we use the first components of C and D ($\{0.58, 0.25, 0.17\}_{0.2}$ and $\{0.45, 0.30, 0.25\}_{0.3}$, respectively, equation 8) then standard Bayesian propagation and using the product t-norm to calculate μ shows that one member of E's updated FS is:

$$\alpha = \{0.3165, 0.2189, 0.4647\}_{0.06} \quad (9)$$

The full FPD for E will have four members, one for each member of $C \times D$ (equation 10, below). For clarity, the calculated α from equation 9 has not been substituted into this equation.

$$E = [\alpha_{0.06},\ \beta_{0.14},\ \gamma_{0.24},\ \delta_{0.56}] \quad (10)$$

In general a variable with k parents that each have an FS with m components will have an updated FS of size m^k. Assuming all variables have k parents, the grand-children will have updated FS of size m^{k^k}, and so forth. This is the *fuzzy state size explosion* (FSSE), and it makes naive belief propagation in a FBN intractable.

4 Dealing with Complexity

There are several ways that the explosion in the complexity can be controlled by approximating the FS. This section discusses four kinds of control. The bimodal fuzzy state $X = [\{0.9_\alpha, 0.1_\beta\}_{0.5}, \{0.1_\alpha, 0.9_\beta\}_{0.5}]$ is used as an example in several places in this section. Such a variable could represent

a committee of two in which the committee members (components) hold diametrically opposite beliefs about the outcome of some future event.

4.1 Linear Collapse

A first approximation that addresses the FSSE is to linearly collapse a FS that is made up only of FPDs, immediately after they are calculated. Each component can be weighted by its fuzzy membership and they can be summed to calculate a single, discrete, PD. For example, B (equation 3) can be collapsed as shown in equation 11. Collapsed FS are denoted with a prime.

$$\begin{aligned} B &= [\{0.1, 0.1, 0.8\}_{0.2}, \{0.1, 0.2, 0.7\}_{0.8}] \\ \therefore B' &= \{0.1, 0.18, 0.72\} \end{aligned} \tag{11}$$

However, this approximation is unsatisfactory: it conflates probability with fuzziness and may change the *expected value* of the variable. Although it may be approximately correct in some circumstances, a simple thought experiment will show why it is insufficient.

Consider the bimodal FS X. The expected sample from X is $X' = [\alpha_{0.5}, \beta_{0.5}]$. Although this sample does not reflect any of the uncertainty in X it does reflect the bi-modality (indecision) of the variable (committee) as a whole. Subsection 4.4 returns to this approach.

If X is linearly collapsed though then $X' = \{0.5, 0.5\}$. No sample drawn from this PD can be half α and half β. Important information in X has been lost. Although further belief propagation will not be biased if this variable is summed over[4], there is no way to compare the linearly collapsed value X' with any observed value for X when trying to evaluate the quality of an inferred network.

Other approximations to the naive approach have been developed. They are discussed in the next three subsections.

4.2 Strict and Dynamic Top Fuzzy Combinations

Consider again the full (naive) FS of E, reproduced in equation 12.

$$E = [\alpha_{0.06},\ \beta_{0.14},\ \gamma_{0.24},\ \delta_{0.56}] \tag{12}$$

Some of the components barely contribute to the overall state and will not have a substantial influence on any children either. Such components could be ignored, and the remaining components could have their μ normalised. For example, if just the top three components of E were used then the updated FS would take the form:

$$E = [\beta_{0.149}, \gamma_{0.255}, \delta_{0.596}] \tag{13}$$

The number of components retained could be either *k-component strict selected* or *ϕ-dynamically selected*. In the former case, the top k components would be selected. In the latter, the $|C|$ components with greatest μ would be selected so that $\sum_{c \in C} \mu_c > \phi$. Strict selection would mean that FBNs were only a small linear factor less efficient than classic BNs of the same size. However, the top k components may not be an accurate reflection of the full FS, thus ϕ-dynamic selection may be more appropriate in some cases.

[4] Due to the use of the product t-norm.

4.3 Clustering the Fuzziness

Another way of controlling the FSSE is to calculate the full FS of each variable during belief propagation. However, before using the full FS to update the state of its children, its components could be clustered so that FPD which specified similar distributions were combined together.

For example, the FS $[\ldots, \{0.7, 0.2, 0.1\}_{0.3}, \{0.6, 0.3, 0.1\}_{0.2}, \ldots]$ might cluster to $[\ldots, \{0.66, 0.24, 0.1\}_{0.5}, \ldots]$.

Because the clustering problem would only have as many dimensions as the range of each FPD, we speculate that a simple fixed-k clustering algorithm like k-means would work very well.

Although this approach is more complex than selection or linear collapse, the total increase in complexity in belief propagation would be related to and bound by the maximum in-degree and range of a variable.

4.4 Expected Values

A fourth kind of control is inspired by particle filtering and the Condensation algorithm[9]. The general *sequential Monte Carlo* (SMC) method will be outlined first. Although this approach is not as efficient as others it is applicable in all cases and is strictly correct.

Consider X. An infinite sequence of independent samples drawn from this uncertain fuzzy state will take something like the form $[\alpha_{0.5}, \beta_{0.5}]$, $[\alpha_{0.5}, \beta_{0.5}]$, $\ldots$, $[\alpha_1]$, $[\alpha_{0.5}, \beta_{0.5}] \ldots$ and so forth.

The properties of this sequence are identical to those of the fuzzy state, and a long-enough finite sequence will be a good approximation to it. For example, 100 samples could be drawn from X. Each of these samples could then be used to propagate the uncertain state of X to X's children. The relative efficiency of this technique compared to clustering depends on the range and k_{max} of the variables, but in certain situations it may also be better.

As noted in subsection 4.1, the expected value of a variable is easily calculated analytically. For example, the expected value of X is $[\alpha_{0.5\times0.9+0.5\times0.1}, \beta_{0.5\times0.1+0.5\times0.9}] = [\alpha_{0.5}, \beta_{0.5}]$.

Doing this expectation calculation is analogous to summing over or numerically integrating a probability distribution, and we call it *fuzzy integration*. Like clustering the impact on efficiency of fuzzy integration depends on the range and k_{max}.

This approach is very similar to linear collapse but it has a number of key advantages. Firstly, like linear collapse, it does not bias any further belief propagation. This is untrue of selection and clustering. Secondly, the expected value X' which is the result of this fuzzy integration can be meaningfully compared with observed values of X when performing network inference. In many cases, users are only interested in the expected (integrated) value. In these cases the expected value of a FS is ideal.

5 A Bioinformatic Domain

Inference of large *genetic regulatory networks* (GRN) is a central problem in modern bioinformatics. However, algorithmic complexity has limited detailed inference using BNs[6;

15; 16] to $N \lessapprox 100$ genes[5]. Approaches which can be applied to larger numbers of genes include modern clustering methods[10] and the inference of *graphical Gaussian models* over clustered gene expression data[8; 14].

FBNs suggest a novel approach to detailed exploration of large GRN. Such a methodology generalises to the inference of other large causal networks as well.

If the data is pre-processed by using a *fuzzy cover* algorithm the dimensionality of the problem may be reduced by an order of magnitude or more. This could lead to an exponential reduction in the algorithmic complexity which would more than offset any increase caused by the fuzzy state size explosion and its collapse.

A fuzzy cover is a clustering algorithm which *covers* the data, rather than clusters it. In a fuzzy cover, a variable (gene) can have $\sum_{c \in C} \mu_c > 1$, where C is the set of covers that the algorithm finds.

Inference over the covers is performed using a standard algorithm to find a virtual GRN. Using the retained μ_c for each $n \in N$ and $c \in C$, most of the original fidelity can be recovered after the inference has been performed by linearly devolving and normalising the network of covers back down to a network of genes. The synergistic use of dimension-reduction and FBNs are what we believe will be most useful.

The authors are using this approach (fuzzy covering, FBN inference, FBN devolution) to infer and explore large *genetic regulatory networks*. With fuzzy clustering and FBNs we expect to be able to perform more detailed exploratory inference for $N \approx 1000$.

6 Contributions and Future Work

This paper has presented a new formalisation which combines fuzzy theory and Bayesian networks. Because of the way that it extends classic BNs, all existing algorithms, tools and machine learning techniques for classic BNs can be used immediately with FBNs.

Several techniques for tractably propagating fuzzy beliefs across a FBN are also described. Using these techniques, previously used BNs can be assigned fuzzy variable states and updated accordingly. This means that existing networks, often learnt only after substantial effort, can be easily reused.

Furthermore, the difference in BN and FBN efficiency with sensible fuzziness collapse may be as little as a small linear constant in some circumstances. This means that there are few disadvantages to using FBNs instead of BNs.

The possibility of integrating FBNs into a machine learning pipeline which involves dimension-reduction and network devolution also suggests that the inference of larger causal networks will be possible using FBNs.

Future research may uncover more efficient methods for integrating, clustering or otherwise collapsing a FS. In addition, the authors plan to present an even more generalised formalisation which relaxes the assumptions made in subsection 3.2.

REFERENCES

[1] Jim F. Baldwin and Enza Di Tomaso, 'Inference and learning in fuzzy Bayesian networks', in *FUZZ'03: The 12th IEEE International Conference on Fuzzy Systems*, volume 1, pp. 630–635, (May 2003).

[2] F. Bobillo and U. Straccia, 'A fuzzy description logic with product t-norm', in *Proceedings of the 16th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pp. 652–657, London (United Kingtom), (July 2007).

[3] Y. Cao, P. Wang, and A. Tokuta, 'Reverse engineering of NK boolean network and its extensions — fuzzy logic network (FLN)', *New Mathematics and Natural Computation*, **3**(1), 68–87, (2007).

[4] David M. Chickering, 'Learning Bayesian networks is NP-Complete', in *Learning from Data: Artificial Intelligence and Statistics V*, eds., D. Fisher and H. J. Lenz, 121–130, Springer-Verlag, (1996).

[5] Christopher Fogelberg and Vasile Palade, 'Machine learning and genetic regulatory networks: A review and a roadmap', Technical Report CS-RR-08-04, Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1-3QD, (April 2008).

[6] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young, 'Combining location and expression data for principled discovery of genetic regulatory network models.', *Pacific Symposium on Biocomputing*, 437–449, (2002).

[7] David Heckerman, 'A tutorial on learning with Bayesian networks', Technical report, Microsoft Research, Redmond, Washington, (1995).

[8] Katsuhisa Horimoto and Hiroyuki Toh, 'Statistical estimation of cluster boundaries in gene expression profile data.', *Bioinformatics*, **17**(12), 1143–1151, (2001).

[9] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking, 1998.

[10] Sara C. Madeira and Arlindo L. Oliveira, 'Biclustering algorithms for biological data analysis: a survey', *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **1**(1), 24–45, (2004).

[11] Daniel Neagu and Vasile Palade, 'A neuro-fuzzy approach for functional genomics data interpretation and analysis', *Neural Computing and Applications*, **12**(3-4), 153–159, (2003).

[12] Heping Pan and Lin Liu, 'Fuzzy Bayesian networks - a general formalism for representation, inference and learning with hybrid Bayesian networks', *IJPRAI*, **14**(7), 941–962, (2000).

[13] Romesh Ranawana and Vasile Palade, 'Multi-classifier systems: Review and a roadmap for developers', *International Journal of Hybrid Intelligent Systems*, **3**(1), 35–61, (2006).

[14] Hiroyuki Toh and Katsuhisa Horimoto, 'Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling.', *Bioinformatics*, **18**(2), 287–297, (2002).

[15] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis, 'Advances to Bayesian network inference for generating causal networks from observational biological data.', *Bioinformatics*, **20**(18), 3594–3603, (2004).

[16] Yu Zhang, Zhingdong Deng, Hongshan Jiang, and Peifa Jia, 'Dynamic Bayesian network (DBN) with structure expectation maximization (SEM) for modeling of gene network from time series gene expression data.', in *BIOCOMP*, eds., Hamid R. Arabnia and Homayoun Valafar, pp. 41–47. CSREA Press, (2006).

Combining Goal Inference and Natural-Language Dialogue for Human-Robot Joint Action

Mary Ellen Foster[1] and **Manuel Giuliani**[1] and **Thomas Müller**[1] and **Markus Rickert**[1] and **Alois Knoll**[1]
Wolfram Erlhagen[2] and **Estela Bicho**[2] and **Nzoji Hipólito**[2] and **Luis Louro**[2]

Abstract. We demonstrate how combining the reasoning components from two existing systems designed for human-robot joint action produces an integrated system with greater capabilities than either of the individual systems. One of the systems supports primarily non-verbal interaction and uses dynamic neural fields to infer the user's goals and to suggest appropriate system responses; the other emphasises natural-language interaction and uses a dialogue manager to process user input and select appropriate system responses. Combining these two methods of reasoning results in a robot that is able to coordinate its actions with those of the user while employing a wide range of verbal and non-verbal communicative actions.

1 INTRODUCTION AND MOTIVATION

As robot systems become increasingly sophisticated, their role is moving from one where the robot is essentially an intelligent tool to one where the robot is able to participate as a full team member in collaborative tasks. Supporting this type of human-robot cooperation requires that the robot system be able to produce and understand a wide range of natural communicative cues in order to allow humans to cooperate with it easily. For example, [15] experimentally demonstrated the contribution of anticipatory action to the fluency of human-robot interaction; similarly, natural-language dialogue has been shown to be an effective means of coordinating actions between a human and a robot [7].

A number of previous systems have also addressed the task of human-robot cooperation, using a variety of communicative styles. The Leonardo robot [2], for example, is able to learn simple action sequences and to execute them jointly with the user. The Ripley system [24] is able to manipulate objects in response to spoken requests from a human partner; a more recent robot from the same group [16] increases the responsiveness of the system and allows the action planner to adapt flexibly to a rapidly-changing world. The BARTHOC [26] and ARMAR [27] humanoid robots both support multimodal dialogue to interact with a human user in a variety of settings and domains. The experiments described in [15] demonstrated that understanding and anticipating the user's actions produces a robot that can cooperate more smoothly with a human user.

Since an intelligent robot system must both process continuous sensor data and reason about discrete concepts such as plans, actions, and dialogue moves, this type of system is often made up of components drawing from an assortment of representation and reasoning paradigms. The robot system described in [22], for example, combines low-level robot control and vision systems with a high-level planner, using connectionist kernel perceptron learning to learn the effects of different domain actions. Integration among the different components of this system is achieved through a common representation of actions and their effects. Such hybrid architectures are also particularly common when the robot is designed to cooperate with a human partner; recent examples include [13, 17, 32].

In this paper, we present two robot systems designed to cooperate with humans on mutual tasks and then show how combining reasoning components from these systems results in a more powerful integrated system. Both of the robot systems have been developed in the context of the JAST[3] project ("Joint Action Science and Technology"). The two main goals of this project are to investigate the cognitive, neural, and communicative aspects of jointly-acting agents, both human and artificial, and to build jointly-acting autonomous systems that communicate and work intelligently on mutual tasks. The common task across the project is *joint construction*—that is, multiple agents working together to assemble objects from their components.

The two JAST human-robot systems support intelligent cooperation with humans on this joint construction task. Although both systems address the same basic task and incorporate similar input- and output-processing components, the reasoning components are implemented using very different techniques and they support very different styles of interaction. The *goal inference* system is implemented using dynamic neural fields and concentrates on inferring the user's intended domain actions based on their non-verbal behaviours and on selecting appropriate domain actions for the system to perform in response. For example, if the user picks up a bolt in a way that indicates that they intend to use it themselves, the system might pick up the corresponding wheel and hold it out to the user. The *dialogue* system, on the other hand, concentrates on understanding and generating multimodal natural-language utterances to support cooperation between the human and the robot, using a dialogue manager. Sections 2–3 present the details of these two systems and show a typical interaction with each.

Since the two JAST human-robot systems address the same task, using complementary forms of reasoning, it is possible to combine the two forms of reasoning into a single system. This integrated system is able both to intelligently infer the user's actions and suggest appropriate responses, and also to engage in dialogue with the user to support coordination and to discuss situations when the system is unable to infer the user's goal. In Section 4, we present this integrated system and show a sample of the interactions that it can support that are not possible with either of the individual systems; this section also gives some technical details of how the components of the two

[1] Technische Universität München, Germany, contact: foster@in.tum.de
[2] University of Minho, Portugal, contact: wolfram.erlhagen@mct.uminho.pt

[3] http://www.euprojects-jast.net/

systems are combined in practice. Finally, in Section 5, we compare the integrated system with other similar systems and summarise the contributions of the system and the areas for future work.

2 GOAL INFERENCE BASED ON DYNAMIC NEURAL FIELDS

The first of the JAST human-robot systems concentrates on giving the robot the ability to predict the consequences of observed actions, using an implementation inspired by neurocognitive mechanisms underlying this capacity in humans and other social species. Many contemporary theories of intention understanding in familiar tasks rely on the notion that an observer uses their own motor repertoire to simulate an observed action and its effect ([4], for a review see [25]). The selection of an appropriate complementary behaviour in a joint action task depends not only on the inferred goal of the partner, but also on the integration of additional information sources such as shared task knowledge (e.g., a construction plan) and contextual cues.

The cognitive control architecture for action coordination in the joint construction scenario is formalized by a coupled system of dynamic fields representing a distributed network of local but connected neural populations [3]. Different pools of neurons encode task-relevant information about action means, action goals, and context in the form of activation patterns that are self-sustained through recurrent interactions.

The motor simulation idea is implemented by the propagation of activity through interconnected neural populations that constitute a learned chain of motor primitives directed towards a specific goal [6]. Typical examples in the context of the construction scenario are reaching-grasping-placing/plugging sequences. The chains are automatically triggered by an observed motor act (e.g., reaching or grasping) whenever additional input from connected dynamic fields (e.g., representing the currently available subgoals) pre-activates the neural populations. As a consequence of the motor simulation, the robot is able to react to the partner's action sequences well ahead of their completion. This anticipation capacity has been shown to be crucial for a fluent team performance [1, 15].

In the layer of the control architecture linked to motor execution, neural populations represent the decision about the most appropriate complementary behaviour. The behaviour is selected as a consequence of a competition process between all response alternatives getting input from connected layers (for details see [1]).

A system based on this dynamic field architecture was implemented to support human-robot cooperation on the JAST joint construction task. This system constructs a toy vehicle (Figure 1) with the user. The vehicle is composed of several components which are initially distributed in the separated working areas of the two teammates; this ensures that neither of the agents is able to reach all of the required components on its own and must rely on the partner to retrieve them, making joint action essential to a successful interaction. The robotics platform we are currently using consists of a torus on which are mounted a 7 DOFs AMTEC arm (Schunk GmbH & Co.KG) with a 3-fingered BARRET hand (Barrett Technology Inc.) and a stereo vision system. The system uses synthesised speech to communicate its reasoning process to the human partner.

To control the arm-hand system, we applied a global planning method in posture space that facilitates the integration of optimization principles derived from experiments with humans [5]. For the object recognition as well as for the classification of object-directed hand postures and communicative gestures such as pointing or demanding an object, a combination of feature- and correspondence-based pattern recognizers were used [30]. As a software development platform we haven chosen YARP [19]. This open-source project supports inter-process communication, image processing and a class hierarchy to ease code reuse across different hardware platforms.

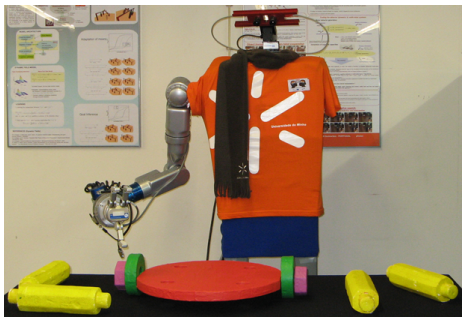


Figure 1. The JAST goal-inference robot together with the toy vehicle that the human and the robot jointly construct.

Figure 2 illustrates a typical example of the goal inference and action selection capacities in this domain. In the top image, the human reaches his open hand towards the robot teammate. By activating the respective action chain in its repertoire, the robot interprets this gesture as a request for a bolt to fix the wheel. Since the human has already mounted the wheel on his side of the construction, this inferred goal describes a currently active subtask. A logical complementary action sequence would be that the robot grasps a bolt to place it in the teammate's hand. However, the human has seemingly overlooked a bolt in his own working area. In this situation, the representation of the inferred goal together with the representation of the bolt in the work space of the human trigger the decision to make a pointing gesture directed towards the object. In addition, the robot uses speech to explain the type of error the human is making.

3 DIALOGUE-BASED HUMAN-ROBOT INTERACTION

Like the system described in the preceding section, the JAST human-robot dialogue system [23] also supports multimodal human-robot collaboration on a joint construction task. In this case, the user and the robot work together to assemble wooden construction toys on a common workspace, coordinating their actions through speech, gestures, and facial displays. The robot (Figure 3) consists of a pair of Mitsubishi manipulator arms with grippers, mounted in a position to resemble human arms, and an animatronic talking head [29] capable of producing facial expressions, rigid head motion, and lip-synchronised synthesised speech. The input channels consist of speech recognition, object recognition, robot sensors, and face tracking; the outputs include synthesised speech, head motions, and robot actions. The components of the system communicate with each other using the Ice distributed object middleware system [14].

The robot is able to manipulate objects in the workspace and to perform simple assembly tasks. The primary form of interaction with the current version of the system is one in which the robot instructs the user on building a particular compound object, explaining the necessary assembly steps and retrieving pieces as required, with the user performing the actual assembly actions. As with the dynamic-field system, the workspace is divided into two areas—one belonging to the robot and one to the human—in order to make joint action necessary for success in the overall task.

Input on each of the channels is processed using a dedicated module for that channel. To process the speech, we use a Java Speech

Figure 2. Example of the goal inference (top) and action selection (bottom) capacities which are implemented by the dynamic field architecture. The robot uses speech to communicate the results of its reasoning about the behaviour of the teammate.

Figure 3. The JAST dialogue robot with a selection of wooden construction-toy components.

API interface to the commercial Dragon NaturallySpeaking speech recogniser [21]. A camera mounted above the common work area provides two-dimensional images of the contents of the workspace. The information from this camera is pre-processed to extract regions of interest (ROIs). The extracted ROIs are then processed in parallel by a template-based object-recognition module [20] and a module that performs static hand-gesture recognition [33].

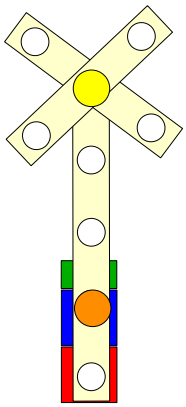
Input received on all of the input sensors is continuously processed by the corresponding modules and broadcast through Ice, using the built-in *IceStorm* publish-subscribe mechanism. All of the input messages are received by a multimodal fusion component [11, 12], which parses the recognized speech into logical forms using the OpenCCG grammar formalism [31] and combines it with the recognised non-verbal behaviour to produce multimodal hypotheses representing user requests. The fusion hypotheses are then sent to the dialogue manager, which selects an appropriate response.

The dialogue manager is implemented using the TrindiKit dialogue management toolkit [18]. This toolkit uses the well-known *information-state update* approach to dialogue management [28], which allows a dialogue to be modelled declaratively. When the dialogue manager receives a new set of fusion hypotheses, it selects the appropriate system response using information from three sources: the inventory of objects in the world, a representation of the current assembly state, and the history of the dialogue. When the system is jointly following an assembly plan with the user, the dialogue manager is able to select from different strategies for traversing the plan: it may use a *postorder* strategy, in which it proceeds directly to describing the concrete assembly actions, or it may use a *preorder* strategy, in which the structure of the plan is described before giving specific assembly actions. More details on the dialogue manager and on the description strategies are given in [10].

Once the dialogue manager has selected a response to the user's multimodal utterance, it sends the specification of the response to the output planner. This module in turn sends commands to select appropriate output on each of the individual channels to meet the specification: linguistic content including appropriate multimodal referring expressions [9], facial expressions and gaze behaviours of the talking head [8], and actions of the robot manipulators. The user then responds to the system utterance by speaking or performing actions in the world, and the interaction continues until the target object has been assembled.

An excerpt from a typical interaction between a user and the JAST dialogue system is shown in Figure 4. In this excerpt, the robot knows the full plan for building the target object: a "railway signal", which has sub-components called a "snowman" and a "flower". The assembled object is shown in Figure 4(a). In the excerpt, the robot instructs the user on how to build the target object, using a preorder strategy, and the user learns to make particular sub-components along the way. We are currently carrying out a system evaluation based on this robot-as-instructor scenario. The evaluation is designed to compare the two description strategies in terms both of user satisfaction and in success in the overall joint-construction task.

We will shortly extend the dialogue system to handle scenarios where the user also knows the assembly plan. In such situations, the main goal of the interaction is no longer instruction, but rather—as with the goal-inference system described previously—coordination between the partners, and the user will be able to take much more initiative in the dialogue than is currently possible. We will return to the details of this extended scenario in the following section.



(a) "Railway signal"

- SYSTEM: To build a railway signal, we need to make a snowman and a flower.
- USER: How do you make a snowman?
- SYSTEM: *[Picking up a red cube and holding it out]* To make a snowman, insert the green bolt into your blue cube and fasten it with this cube.
- USER: *[picks up bolt and cube; takes other cube; assembles "snowman"]* Okay.
- SYSTEM: Well done, you have made a snowman. Now we need to make a flower. To make a flower, [. . .]

(b) Dialogue excerpt

Figure 4. A sample object and an excerpt from an interaction where the robot instructs the user on how to construct this object.

4 INTEGRATING GOAL INFERENCE AND NATURAL-LANGUAGE DIALOGUE

There are a number of similarities between the two human-robot systems described above. Both support the same basic task—joint construction—and view the goals and subgoals of this task in a similar way. Also, the input and output channels used by the two systems are very similar: both include object and gesture recognition in the input and produce speech and robot-manipulator actions as part of their output. On the other hand, the reasoning processes used by the two systems are very different: the former uses dynamic neural fields to perform goal inference and action selection based entirely on non-verbal input, while the latter uses techniques from issue-based dialogue management to engage in natural-language conversation with some multimodal components. The strengths of the two systems are also complementary: the dynamic-field system is good at detecting and reasoning about the user's non-verbal actions, but uses language only for a limited form of canned output; the dialogue system supports advanced linguistic interaction, but has no mechanism to infer the user's intention from their actions in the world.

Motivated by the above similarities and complementary features, we have combined components from the two individual human-robot systems into a single, integrated architecture. The hardware platform for the integrated system is the robot from the dialogue system (Figure 3), while the scenario is an extended version of the scenarios used by each of the individual systems. As in the dynamic-field scenario, the user and the robot are both assumed to know the assembly plan for the target object and are able to infer the partner's intentions based on their behaviour, and the main goal of the interaction is for the two participants to coordinate their actions. As in the dialogue system, this coordination is accomplished through natural-language dialogue incorporating both verbal and non-verbal communication.

Figure 5 shows the high-level architecture of the integrated system. Messages on all of the multimodal input channels (speech, gestures, and recognised objects) are sent to both of the input-processing components, each of which—just as in the individual systems—reasons about the meaning of the user's actions in the current context, each drawing information from the same set of state modules (plan state, object inventory, interaction history). The inferred goals and suggested system responses from the goal-inference system are then passed to the the dialogue manager, which incorporates this information along with the processed messages from the fusion system into the (extended) information state of the integrated system. The dialogue manager then uses enhanced update rules to select an appropriate system response to the input. Finally, just as in the individual systems, the selected response is sent to the output system for realisation on the output channels.

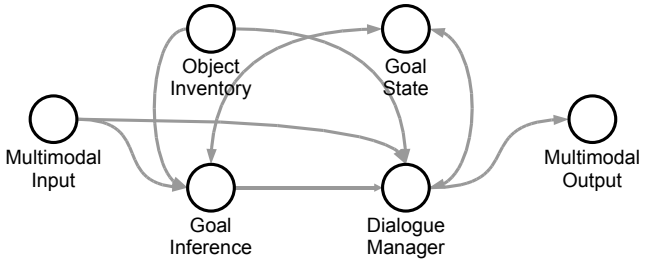


Figure 5. The architecture of the integrated system.

This integrated system supports interaction patterns that would not be possible with either of the individual systems. Most importantly, it is able both to detect unexpected actions from the user (i.e., actions that do not meet what it believes to be the current subgoals) and to engage the user in dialogue to discuss how to deal with the unexpected action. When both forms of reasoning work together, the system is able to detect such user actions and to produce a variety of responses, including correcting the user, asking for clarification as to the user's intentions, or attempting to silently revise its representation of the goal state. Varying the system's response to this situation is able to produce systems with different interactive "personalities", ranging from one that always makes the user follow the plan selected by the system to one where the user has full control over the course of the interaction.

Figure 6 shows a sample interaction between a user and the integrated system, where the role of each of the reasoning components is shown throughout. In this interaction, the user and the robot are jointly building the "railway signal" object (Figure 4(a)). At the start, the robot system has assumed that the user is building the "snowman" sub-component. When the user grasps a medium slat, which is not needed for that subgoal, the goal inference system detects this (just as in the sample interaction described at the end of Section 2) and sends a message to the dialogue manager that the user's action cannot be integrated into the current plan.

At this point, the system has several options to deal with the mismatch between its beliefs about the current subgoals and the recent action of the user. It might silently revise its view of the current subgoals, for example, or it might—as in Figure 2—correct the user's apparent "error". In the example, the system uses a third strategy, and one that is only available because of the integration of the dialogue components: it asks the user in words to clarify their intentions. After the user provides the needed clarification, also verbally, the dialogue manager updates the system's subgoals and informs the goal-inference system of the change. The goal-inference system then anticipates that, to meet this new subgoal, the user will need the nut that is lying on the robot's side of the table. The system therefore picks up the nut and offers it to the user without being asked.

As can be seen by the right-hand columns in Figure 6, this type of interaction would not be possible with either of the individual systems. The dialogue system does not have the necessary mechanism to infer the user's goals from their actions, while the goal-inference system would only have been able to respond to the user's unexpected

Actions	Dialogue Manager	Goal Inference
User grasps a medium slat		
		Notices that action does not meet current subgoal
	Tells output planner to ask for clarification	
SYSTEM: "We don't need a medium slat for the snowman"		
USER: "Yes, but I want to build the flower now"		
	Interprets response and updates subgoals	
		Suggests system response
	Sends message to output planner	
Robot picks up a nut and holds it out SYSTEM: "Then you'll need this nut"		

Figure 6. A sample interaction with the integrated system, showing the role of each individual reasoning component in the decision-making process.

action by treating it as an error rather than discussing the user's goals as in the example. Only when these two components are combined is this rich interaction made possible.

4.1 Technical Details

The two individual systems use the same basic information in their reasoning (task goals and subgoals, object inventory, input events); however, due to the different implementations, they represent this information quite differently. Also, at the implementation level, the components of the dynamic-field system use YARP to communicate with one another, while the dialogue system uses Ice as an integration platform. A specific goal of the integration has been to make as few changes as possible to the individual systems. An important aspect of creating the integrated system has therefore been coming up with a common representation for all of the relevant information, where the representation is compatible with both of the systems and both of the integration platforms.

To support the integration, we have defined generic interfaces to represent recognised gestures and objects, as well as inferred and proposed domain actions. These representations include the following information:

- The **Gestures** representation includes the type of gesture recognised (pointing, grasping, holding-out, unknown) and if necessary, the object indicated.
- The **Objects** representation includes the classification of the object, a 3D position and a flag indicating whether the object can be reached by the robot.
- The **Action** representation consists of the type of action (grasp-and-give, demand-and-receive, speak, undefined) and a string containing further specifications (e.g. the object-id for grasp-and-give or the sentence to speak out loud).

Internal communication between YARP and Ice is implemented via a connector module that translates Ice messages to YARP messages and vice versa.

5 DISCUSSION

We have presented two human-robot systems, each of which is designed to support the same joint construction task. One system uses dynamic neural fields to perform non-verbal goal inference and action selection, while the other uses a dialogue manager to support multimodal natural-language interaction. We have then shown how a system integrating the reasoning components of the two individual systems is able to take advantage of the complementary strengths of each to support interactions that neither system is able to support on its own. In particular, this integrated system is able both to detect the user's intentions and anticipate their needs, and to use natural-language dialogue to manage the joint activity. The integration of these two systems is made possible through well-defined interfaces that allow the two sets of reasoning components to share information about world state, task goals, and input events.

In contrast to the other systems mentioned in the introduction, the integrated JAST system is unique in that it combines methods and techniques taken from two separate, fully-implemented, existing systems—a neuro-inspired perception-action system and a symbolic, multimodal-dialogue system—to produce an integrated robot system that is able both to communicate with its human partner using language and to intelligently understand and anticipate the partner's intentions. As demonstrated by the example in the preceding section, the integrated system is able to go beyond the capabilities of either of the individual systems to support intelligent human-robot cooperation on the joint construction task.

The integrated system is currently under development: the necessary interfaces have been specified as described in Section 4.1, and the reasoning modules from the two systems are being adapted to use the common interfaces. When the is completed, we will run a user evaluation of the full system similar to that currently under way for the dialogue system to demonstrate the contribution of both forms of reasoning to natural human-robot joint action.

ACKNOWLEDGEMENTS

This work was supported by the EU FP6 IST Cognitive Systems Integrated Project "JAST" (FP6-003747-IP), `http://www.euprojects-jast.net/`. Thanks to the CIMA workshop reviewers for their useful comments and suggestions.

REFERENCES

[1] E. Bicho, L. Louro, N. Hipólito, and W. Erlhagen, 'A dynamic neural field architecture for flexible and fluent human-robot interaction.', in *Proceedings of the 2008 International Conference on Cognitive Systems*, pp. 179–185, (2008).

[2] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Chilongo, 'Tutelage and collaboration for humanoid robots', *International Journal of Humanoid Robotics*, **1**(2), 315–348, (2004).

[3] W. Erlhagen and E. Bicho, 'The dynamic neural field approach to cognitive robotics', *Journal of Neural Enginering*, **3**, R36–R54, (2006).

[4] W. Erlhagen, A. Mukovskiy, and E. Bicho, 'A dynamic model for action understanding and goal-directed imitation.', *Brain Research*, **1083**, 174–188, (2006).

[5] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. van Schie, and H. Bekkering, 'Goal-directed imitation for robots: a bio-inspired approach to action understanding and skill learning', *Robotics and Autonomous Systems*, **54**, 353–360, (2006).

[6] W. Erlhagen, A. Mukovskiy, F. Chersi, and E. Bicho, 'On the development of intention understanding for joint action tasks', in *Proceedings of the 6th IEEE International Conference on Development and Learning*, (July 2007).

[7] T. Fong, C. Thorpe, and C. Baur, 'Collaboration, dialogue, and human-robot interaction', in *Robotics Research*, volume 6 of *Springer Tracts in Advanced Robotics*, 255–266, Springer, (2003).

[8] M. E. Foster, 'Roles of a talking head in a cooperative human-robot dialogue system', in *Proceedings of the 7th International Conference on Intelligent Virtual Agents (IVA 2007)*, (September 2007).

[9] M. E. Foster, E. G. Bard, R. L. Hill, M. Guhe, J. Oberlander, and A. Knoll, 'The roles of haptic-ostensive referring expressions in cooperative, task-based human-robot dialogue', in *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction (HRI 2008)*, (March 2008).

[10] M. E. Foster and C. Matheson, 'Following assembly plans in cooperative, task-based human-robot dialogue', in *Proceedings of Londial 2008*, (June 2008).

[11] M. Giuliani and A. Knoll, 'Integrating multimodal cues using grammar based models', in *Proceedings of HCI International 2007*, (July 2007).

[12] M. Giuliani and A. Knoll. MultiML – A general purpose representation language for multimodal human utterances, 2008. Submitted.

[13] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G.-J. Kruijff, M. Brenner, G. Berginc, and D. Skočaj, 'Towards an integrated robot with multiple cognitive functions', in *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI 2007)*, (2007).

[14] M. Henning, 'A new approach to object-oriented middleware', *IEEE Internet Computing*, **8**(1), 66–75, (Jan–Feb 2004).

[15] G. Hoffman and C. Breazeal, 'Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team', in *Proceedings of the 2nd ACM/IEEE International Conference on Human Robot Interaction (HRI 2007)*, (2007).

[16] K.-y. Hsiao, S. Vosoughi, S. Tellex, R. Kubat, and D. Roy, 'Object schemas for responsive robotic language use', in *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction (HRI 2008)*, (2008).

[17] W. G. Kennedy, M. D. Bugajska, M. Marge, W. Adams, B. R. Fransen, D. Perzanowski, A. C. Schultz, and J. G. Trafton, 'Spatial representation and reasoning for human-robot collaboration', in *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI 2007)*, (2007).

[18] S. Larsson and D. R. Traum, 'Information state and dialogue management in the TRINDI dialogue move engine toolkit', *Natural Language Engineering*, **6**, 323–340, (2000).

[19] G. Metta, P. Fitzpatrick, and L. Natale, 'YARP: Yet another robot platform', *International Journal of Advanced Robotics Systems*, **3**(1), 43–48, (2006).

[20] T. Müller, P. Ziaie, and A. Knoll, 'A wait-free realtime system for optimal distribution of vision tasks on multicore architectures', in *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics*, (May 2008).

[21] Nuance Communications. Dragon NaturallySpeaking 9. `http://www.nuance.com/naturallyspeaking/`.

[22] R. Petrick, D. Kraft, K. Mourão, C. Geib, N. Pugeault, N. Krüger, and M. Steedman, 'Representation and integration: Combining robot control, high-level planning, and action learning', in *Proceedings of the International Cognitive Robotics Workshop (CogRob 2008) at ECAI 2008*, (July 2008).

[23] M. Rickert, M. E. Foster, M. Giuliani, T. By, G. Panin, and A. Knoll, 'Integrating language, vision and action for human robot dialog systems', in *Proceedings of HCI International 2007*, (July 2007).

[24] D. Roy, K.-Y. Hsiao, and N. Mavridis, 'Mental imagery for a conversational robot', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **34**(3), 1374–1383, (June 2004).

[25] N. Sebanz, H. Bekkering, and G. Knoblich, 'Joint action: bodies and minds moving together', *Trends in Cognitive Sciences*, **10**, 70–76, (2006).

[26] T. Spexard, M. Hanheide, and G. Sagerer, 'Human-oriented interaction with an anthropomorphic robot', *IEEE Transactions on Robotics*, **23**(5), 852–862, (October 2007).

[27] R. Stiefelhagen, H. Ekenel, C. Fugen, P. Gieselmann, H. Holzapfel, F. Kraft, K. Nickel, M. Voit, and A. Waibel, 'Enabling multimodal human-robot interaction for the Karlsruhe humanoid robot', *IEEE Transactions on Robotics*, **23**(5), 840–851, (October 2007).

[28] D. Traum and S. Larsson, 'The information state approach to dialogue management', in *Current and New Directions in Discourse and Dialogue*, eds., J. C. J. Van Kuppevelt and R. W. Smith, 325–353, Kluwer Academic Publishers, (2003).

[29] A. J. N. van Breemen, 'iCat: Experimenting with animabotics', in *Proceedings of the AISB 2005 Creative Robotics Symposium*, (2005).

[30] G. Westphal, C. von der Malsburg, and R. Würtz, 'Feature-driven emergence of model graphs for object recognition and categorization', in *Applied Pattern Recognition*, eds., A. Kandel, H. Bunke, and M. Last, 155–199, Springer Verlag, (2008).

[31] M. White, 'Efficient realization of coordinate structures in Combinatory Categorial Grammar', *Research on Language and Computation*, **4**(1), 39–75, (2006).

[32] H. Zender, P. Jensfelt, Ó. Martinez Mozos, G.-J. M. Kruijff, and W. Burgard, 'An integrated robotic system for spatial understanding and situated interaction in indoor environments', in *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI 2007)*, (July 2007).

[33] P. Ziaie, T. Müller, M. E. Foster, and A. Knoll, 'Using a naïve Bayes classifier based on k-nearest neighbors with distance weighting for static hand-gesture recognition in a human-robot dialog system', in *Proceedings of CSICC 2008*, (March 2008).

A Tool for Evolving Artificial Neural Networks

Efstratios F. Georgopoulos[1, 3], Adam V. Adamopoulos[2, 3] and Spiridon D. Likothanassis[3]

Abstract. A hybrid evolutionary algorithm that combines genetic programming philosophy, with localized Extended Kalman Filter (EKF) training method is presented here. This algorithm is used for the topological evolution and training of Multi-Layered Neural Networks. It is implemented as a visual software tool in C++ programming language. The proposed hybrid evolutionary algorithm is applied on two bio-signal modeling tasks: the Magneto Encephalogram (MEG) of epileptic patients and the Magneto Cardiogram (MCG) of normal subjects, exhibiting very satisfactory results.

1 INTRODUCTION

One of the main problems that are faced when Artificial Neural Networks (ANN) and especially Multilayer Perceptrons, are applied on some tasks, is finding the network architecture or topology that is best suited for the task at hand. A small network for the problem might causes poor learning ability, while a large one might cause poor generalization. Until now the common method to determine the architecture of a neural network is by trial and error. However, in the last years there have been many attempts, in the direction of designing the architecture of a neural network automatically, that have led to a variety of methods.

Two subcategories of such methods are a) the constructive and b) pruning (destructive) algorithms, [28], [29]. Roughly speaking, a constructive algorithm starts with a minimal network, that is an ANN with a minimal number of hidden layers, hidden neurons and connections, and adds new layers, neurons or connections, if it is necessary, during the training phase. On the opposite, a pruning (destructive) algorithm does the opposite, starts with a maximal network and deletes the unnecessary layers, nodes and connections during training.

Another approach to this problem is by using Genetic Algorithms. Genetic Algorithms are a class of optimization algorithms, which are good in exploring a large and complex space in an intelligent way in order to find values close to the global optimum (see [12], [15], [20] and [22] for details). The design of a near optimal topology can be formulated as a search problem in the architecture space, where each point in the space represents network architecture. The training can be formulated as a search problem in the weight space. Since the end of the last decade, there have been several attempts to combine the technology of neural networks with that of genetic algorithms. Given some performance criteria, for example error, generalization ability, learning time, architectural complexity etc, for the architecture, the performance level of all architectures forms a surface in the space. The optimal architecture design equals to finding the optimum point on this surface.

The first attempts, described in [10], [23], [25] and [27], focused mainly on the problem of training the networks and not in the topology design. They used neural networks with fixed architecture and genetic algorithms in order to search the weight space for some near optimum weight vector that solves the problem of network training. That is, they used genetic algorithms instead of some classical training algorithm. Soon the main research interest moved from the training, to the search for the optimal architectural (or topological) design of a neural network. Some first works used genetic algorithms in order to imitate the pruning algorithms. They start with a network larger than necessary for the task and then use a specially designed genetic algorithm to define which combination of connections is sufficient to, quickly and accurately, learn to perform the target task, using back propagation. Miller et al. [21] did that for some small nets. The same problem, but for larger networks, was faced by Whitley and Bogard in [26]. Bornholdt and Graudenz in [9], used a modified GA in order to evolve a simplified model of a biological neural network and then applied the algorithm to some toy Boolean functions. A different approach to the design and evolution of modular neural network architectures is presented in [13]. Billings and Zheng in [8] used a GA for the architectural evolution of radial basis function (RBF) networks. The most recent approach and maybe the most successful one, to the problem of finding the near optimum architecture is presented in [28]. There, Yao and Liu propose a new evolutionary system, the EPNet, for evolving artificial neural networks' behavior.

The last couple of years, there is an increasing interest in the use of multi-objective optimization methods and especially evolutionary multi-objective techniques for neural network training and structure optimization. Two very interesting approaches are presented in [31] and [32].

The present work is the sequence of a series of efforts concerning the application of evolutionary algorithms for the optimization of neural networks. In [17] a neural network model with binary neurons was evolved by a modified genetic algorithm in order to learn some Boolean functions. In [1], [2], [3], [4], [5], [6], [7], [11], [18] and [19] genetically evolved artificial neural networks were successfully used for a variety of problems.

In this paper we present a hybrid evolutionary method that looks like more to a genetic programming technique for the evolution of a population of feed-forward Multi Layered Perceptrons [14]. This hybrid algorithm combines a genetic programming technique (for details see [16]) for the evolution of the architecture of a neural network, with a training method based on the localized Extended

[1] Technological Educational Institute of Kalamata, Greece, e-mail: sfg@teikal.gr
[2] Dept. of Medicine, Democritus University of Thrace, Greece, e-mail: adam@med.duth.gr
[3] Dept. of Computer Engineering & Informatics, University of Patras, Greece, e-mail: likothan@cti.gr

Kalman Filter (EKF), known as Multiple Extended Kalman Algorithm (MEKA). The MEKA is described in detail in [24]. The novelty of this effort depends on, apart from the combination of evolution techniques with MEKA, the capability of the proposed method to search, not only for the optimal number of hidden units, but also, for the number of inputs needed for the problem at hand; of course this stands only for time series prediction problems where the number of needed past values, which represent the network's inputs, is unknown. This hybrid algorithm is an evolved and heavily enriched version of an older algorithm that was developed by the authors and presented in [4], [7] and [19]. Furthermore this evolutionary neural network system has been implemented as a visual tool in C++ with a graphical user interface. In order to test the ability of this algorithm to produce networks that perform well, we apply the system on two biosignals, namely the Magneto Encephalogram (MEG) recordings of epileptic patients and Magneto Cardiogram (MCG) of normal subjects. The algorithm produces networks with small sizes that perform well.

The rest of the paper is organized as follows. Section 2 describes the hybrid evolutionary algorithm, while the numerical experiments are presented in section 3. Finally, section 4 discusses the concluding remarks.

2 THE HYBRID EVOLUTIONARY ALGORITHM

2.1 THE MULTIPLE EXTENDED KALMAN ALGORITHM - MEKA

Consider a network characterized by a weight vector w. The average cost function that should be minimized during the training phase is defined in terms of *N* input-output patterns as follows:

$$E_{av}(w)=\frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C}\left[d_j(n)-y_j(n)\right]^2 \tag{1}$$

Where $d_j(n)$ is the desired response and $y_j(n)$ the actual response of output neuron *j* when input pattern *n* is presented, while the set *C* includes all the output neurons of the network. The cost function $E_{av}(w)$ depends on the weight vector *w* due to the fact that $y_j(n)$ itself depends on *w*.

Concentrating on an arbitrary neuron *i*, which might be located anywhere in the network, its behavior during the training phase may be viewed as a non-linear dynamic system, which in the context of Kalman filter theory may be described by the following state-measurement equations [14], [24]:

$$w_i(n+1)=w_i(n) \tag{2}$$

$$d_i(n)=y_i(n)+e_i(n) \tag{3}$$

$$y_i(n)=\varphi\left(x_i^T(n),w_i(n)\right) \tag{4}$$

Where the iteration n corresponds to the presentation of the nth input pattern, $x_i(n)$ and $y_i(n)$ are the input and output vector of neuron i respectively and $e_i(n)$ is the measurement error at the output of neuron i, the instantaneous estimate of which is given by:

$$e_i(n)=-\frac{\partial E(n)}{\partial y_i(n)} \tag{5}$$

$$E(n)=\frac{1}{2}\sum_{j\in C}\left[d_j(n)-y_j(n)\right]^2 \tag{6}$$

The differentiation in equation (5) corresponds to the back-propagation of the global error to the output of neuron *i*. The activation function $\varphi(\bullet)$ is responsible for the non-linearity in the neuron. The weight vector w_i of the optimum model for neuron *i* is to be "estimated" through training with examples. The activation function is assumed to be differentiable. Accordingly, we can use Taylor series to expand equation (3) about the current estimate of the weight vector and thereby linearize the equation as follows [14]:

$$\phi\left(x_i^T(n)w_i(n)\right)\cong q_i^T(n)w_i(n)+\left[\phi\left(x_i^T(n)\hat{w}_i(n)\right)-q_i^T(n)\hat{w}_i(n)\right] \tag{7}$$

where

$$q_i(n)=\left[\frac{\partial\phi\left(x_i^T(n)w_i(n)\right)}{\partial w_{i(n)}}\right]_{w_i(n)=\hat{w}_i(n)}=\hat{y}_i(n)\left[1-\hat{y}_i(n)\right]x_i(n) \tag{8}$$

$\hat{y}_i(n)$ is the output of neuron *i* that results from the use of the weight estimate. In equation (8) we have assumed the use of the logistic function; other sigmoid functions, like the hyperbolic tangent, can be used as well. The first term of the right hand side of equation (7) is the desired linear term while the remaining term represents a modeling error. Thus substituting equation (7) and (4) in (3) and ignoring the modeling error we obtain:

$$d_i(n)=q_i^T(n)w_i(n)+e_i(n) \tag{9}$$

Where $e_i(n)$ and $q_i(n)$ are defined in equations (5) and (8) respectively.

Equations (2) and (9) describe the linearized behavior of neuron *i*. Given the pair of equations (2) and (9), we can make use of the standard Recursive Least Squares (RLS) algorithm equations [14], which is a special case of the Kalman filter, to make an estimate of the weight vector $w_i(n)$ of neuron *i*. The resulting solution is defined by the following system of recursive equations [14] that describe the Multiple Extended Kalman Algorithm (MEKA) [24]:

$$r_i(n)=(\lambda-1)\cdot P_i(n-1)\cdot q_i(n) \tag{10}$$

$$k_i(n)=r_i(n)\cdot\left(1+r_i^T(n)\cdot q_i(n)\right)-1 \tag{11}$$

$$w_i(n+1)=w_i(n)+e_i(n)\cdot k_i(n) \tag{12}$$

$$P_i(n+1)=(\lambda-1)\cdot P_i(n)-k_i(n)\cdot r_i^T(n) \tag{13}$$

Where, *n=1,…,N* is the iteration number and *N* is the total number of examples.

The vector $q_i(n)$ represents the linearized neuron activation function given in equation (6), $P_i(n)$ is the current estimate of the inverse of the covariance matrix of $q_i(n)$ and $k_i(n)$ is the Kalman gain. The parameter λ is a forgetting factor which takes values in the range (0,1], and $e_i(n)$ is the localized measure of the global error. Equation (13) is called the Riccatti difference equation.

Each neuron in the network perceives its own effective input $q_i(n)$, hence it has to maintain its own copy of $P_i(n)$ even in the

case in which it may share some of its inputs with other neurons in the network.

2.2 THE EVOLUTIONARY ALGORITHM

The proposed evolutionary algorithm is an improved version of a modified genetic algorithm that was used aforetime by the authors. It maintains the basic working philosophy of evolutionary algorithms and resembles genetic programming (see [16] for details) since it evolves complicated structures like linked lists and not simple bit strings as genetic algorithms do.

The algorithm evolves, using a number of genetic operators, a population of artificial neural networks (multilayered perceptrons) that are represented as linked lists of network layers and neurons; thus it is used the direct encoding scheme. The basic steps of the algorithm are as follow:

1. Initialization: An initial population of neural networks (called individuals) is created. Every individual has a random number of neurons (or nodes) and connections (synapses). The connection weights are initialized to some random values within a specific range.

2. Training: Every individual (neural network) in the population is trained using MEKA for a small number of training epochs. For populations other than initial, training occurs only for those networks that have been changed by the application of genetic operators.

3. Fitness Evaluation: As fitness function it is used a function that combines the performance of the network in the training and/or validation set with the size of the network. The performance is evaluated using the Mean Squared Error (MSE) or the Mean Relative Error (MRE). While, the size is the number of neurons and/or the number of active synapses in the network. So the fitness function for the case of MRE has a formula of the type:

$$Fitness(i) = \frac{1}{1 + MRE(i) + sp \cdot MRE(i) \cdot SIZE(i)} \quad (14)$$

Where *sp* is a parameter that controls the weight of the network size in the evaluation of fitness, *MRE(i)* is the value of MRE of individual *i*, *SIZE(i)* is the size of individual i which can be calculated as the number of active connections or the number of neurons and *i* is an index taking values in the range 1 to population size.

4. Selection: Selection operator is been used in order to create a new, intermediate, population from the old one, by selecting individuals based on their fitness. This can be done using any of the following three different selection schemes that have been implemented, namely:
   - The Elitism Roulette Wheel Selection Operator, with variable elitist pressure (for more details see [12], [16], [20] and [22]).
   - The Rank Based Selection (for more details see [12], [16], [20] and [22]).
   - The Tournament Selection with variable tournament size (for more details see [12], [16], [20] and [22]).

5. *Mutation:* It works on the members of Three different mutation operators are implemented:
   - *Input Mutation:* it selects randomly a neural network from the population and changes its number of inputs. This operator works only on time series modeling and prediction problems, where the number of past values (network inputs) needed to predict future values is not usually known a priory.
   - *Hidden mutation:* it selects randomly a neural network from the population and changes the structure of its hidden region by adding or deleting a random number (selected uniformly from a given interval) of hidden neurons.
   - *Non Uniform Weight mutation:* it is responsible for the fine tuning capabilities of the system. It selects randomly a number of connection weights and changes their values to new ones as follows: Let suppose that w is the old weight value then the new one is given by the formula:

$$w(n+1) = w(n) \pm \Delta w\left(t, ub - w(n)\right) \quad (15)$$

Where *lb* and *ub* are the lower and upper bounds of the weight values, *t* is the generation number, and $\Delta(t,y)$ is a function that returns a value in the range [0,y], such that the probability of $\Delta(t,y)$ being close to 0 increases as t increases. This property causes this operator to search the solution space initially uniformly (while t is small) and very locally at the later stages. In our experiments the following function, [20] was used:

$$\Delta(t,y) = y \cdot \left(1 - r^{\left(1 - t/T\right)^b}\right) \quad (16)$$

Where *r* is a random number on [0,1], *T* is the maximal generation number (a parameter of the algorithm), and *b* is a system parameter determining the degree of non-uniformity.

   - *Gaussian weight mutation:* it works like the *Non Uniform Weight mutation* operator with the difference that the new weight value is calculated by the formula:

$$w(n+1) = w(n) + \Delta w(n) \quad (17)$$

Where, Δw is a small random number following Gaussian distribution.
   - *Uniform weight mutation: it works like the Gaussian mutation operator with the difference that, Δw is a small random number following Uniform distribu*tion.

6. *Crossover:* It selects two parents (neural networks) and generates one or two offspring by recombining parts of them. The offspring take the place of their parents in the new population. In the presented algorithm crossover recombines whole neurons with their incoming connections. But since we have to deal with networks with different structures, the new connections that might have to be produced are initialized with random weight values as in the initialization phase. Herein crossover works more like a mutation operator, like in most genetic programming systems, than as the recombination operator of genetic algorithms

Therefore the presented hybrid evolutionary algorithm works in brief as follows: it starts with a population of randomly constructed Neural Networks (step 1). Networks undergo some training for a couple of epochs with MEKA, using the training set (step 2). Performance is measured with the fitness function (step 3) using the validation set, in order to improve generalization. Then a new, intermediate, population is created, by selecting the more fit individuals according to their fitness (step 4) using any of the three selection schemes. Some members of this intermediate population undergo transformations by means of genetic operators to form the members (individuals) of the new population: mutation (step 5) and

crossover (step 6) operators. The new population that is created is trained again (step 2); new members are trained for a couple of epochs, while the members that have survived and passed from the old population may be trained with MEKA for some more epochs, or may not be trained at all. This is the new generation. This whole process continues until a predefined termination condition is fulfilled; the termination condition might be a maximum number of generation or a minimum error (MSE or MRE) value. Once terminated the algorithm is expected to have reached a near-optimum solution, i.e. a trained network with near optimum architecture.

2.3 THE TOOL

This hybrid evolutionary algorithm has been implemented as a visual tool in C++ programming language, having a graphical user interface (GUI). Specifically, it was used the Borland C++ version 6.0 IDE for Windows. Figure 1 and 2 depict two of the basic forms of the program, for the two main categories of problems that it can be used for, classification and time series prediction. Figure 3 is the "statistics" form that illustrates the evolutionary process and prints useful information about it.

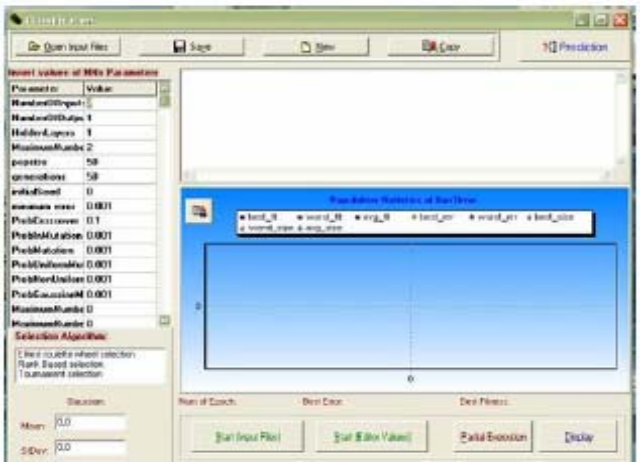


Figure 1. The main form for classification problems of the evolutionary neural network system

Figure 2. The main form for prediction problems of the evolutionary neural network system

The user can select between this two problem categories. Then he/she can insert the values of the various genetic parameters, the training, validation and test files, as well as the output log files. The user can observe the evolutionary process using some real time graphical display of the error, the performance of the best ever network and other parameters.

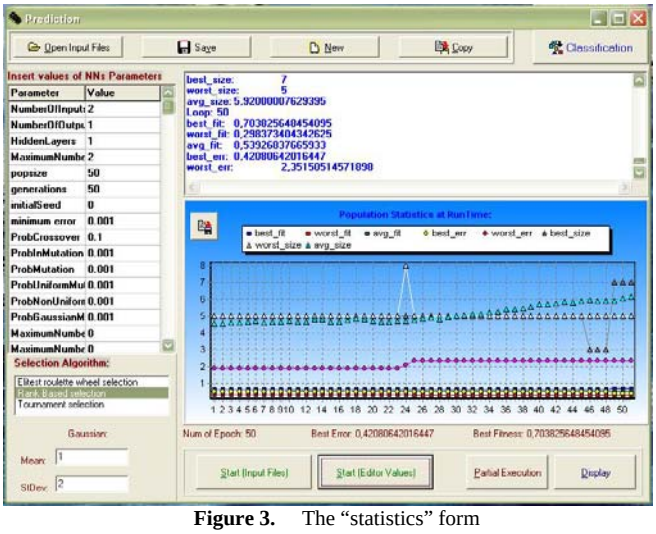


Figure 3. The "statistics" form

3 NUMERICAL EXPERIMENTS

In order to examine the ability of the algorithm to produce networks that learn and generalize well we have tested it on two real world problems: the modeling of the MEG recordings of epileptic patients and the modeling MCG recordings of normal subjects.

Brain dynamics can be evaluated by recording the changes of the neuronal electric voltage, either by the electroencephalogram (EEG), or by the MEG. The EEG recordings represent the time series that match up to neurological activity as a function of time. On the other hand the MEG is generated due to the time varying nature of the neuronal electric activity, since time-varying electric currents generate magnetic fields. EEG and MEG are considered to be complementary, each one carrying a part but not the whole of the information related to the underlying neural activity. Thus, it has been suggested that the EEG is mostly related to the inter-neural electric activity, whereas the MEG is mostly related to the intra-neural activity. The MEG recordings of epileptic patients were obtained using a Super-conductive QUantum Interference Device (SQUID) and were digitized with a sampling frequency of 256Hz using a 12-bit A/D Converter. SQUID is a very sensitive magnetometer, capable to detect and record the bio-magnetic fields produced in the human brain due to the generation of electrical micro-currents at neural cellular level [30].

The same stands for the MCG recordings which are magnetic recordings of the heart operation of normal subjects. MEG and MCG data were provided by the Laboratory of Medical Physics of the Democritus University of Thrace, Greece, where a one-channel DC SQUID is operable. Both biosignal data were normalized in the interval [0,1] in order to be processed by the neural networks.

In all the experiments we used, for comparison reasons, the same parameter values, which are depicted in table 1. For the case of the MEG modeling, as training set where used 1024 data samples (corresponding to a four seconds epoch of the MEG) while for the testing was used 512 data samples (corresponding to a two seconds epoch of the MEG). For the case of the MCG modeling, as training set where used 1024 data samples and for the test set was used 1024 data samples The algorithm was left to run over 1000 generations.

In order to evaluate the forecasting capability of the produced networks we used three well-known error measures, the Normalized Root Mean Squared Error (NRMSE), the Correlation Coefficient (CC) and the Mean Relative Error (MRE). The performance of the hybrid algorithm for the case of MEG modeling is depicted in tables 2 and 3 and in figure 4, while for the case of MCG in tables 4 and 5 and in figure 5.

Table 1. Parameters used for the cases of MEG and MCG modeling

Parameter	*Value*
Population	50
Max number of Generations	1000
Crossover Probability	0,1
Input Mutation Probability	0,1
Weight Mutation Probability	0,1
Uniform Mutation Probability	0,1
nonUniform Mutation Probability	0,1
Gaussian Mutation Probability	0,1
MeanGaussian	0,1
StDev Gauss	0,25
Predicting Horizon	1

Table 2. MEG forecasting - Errors on the Training Set

Architecture	NRMSE	C.C.	MRE
4-9-1	0.2540	0.9674	0.0350
3-3-1	0.2913	0.9581	0.0386
4-5-1	0.2564	0.9672	0.0357
3-2-1	0.2967	0.9557	0.0411
3-3-1	0.2705	0.9656	0.0369
3-4-1	0.2655	0.9645	0.0361

Table 3. MEG forecasting - Errors on the Test Set

Architecture	NRMSE	C.C.	MRE
4-9-1	0.1971	0.9805	0.0403
3-3-1	0.2189	0.9757	0.0438
4-5-1	0.2063	0.9786	0.0434
3-2-1	0.2309	0.9733	0.0466
3-3-1	0.2177	0.9765	0.0446
3-4-1	0.2111	0.9775	0.0429

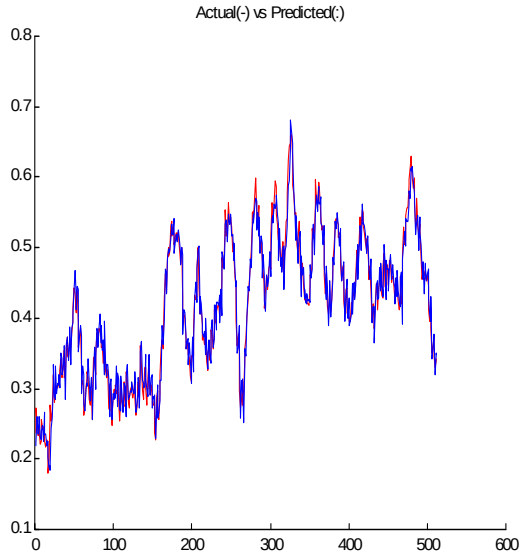


Figure 4. MEG forecasting, performance on the test set.

Table 4. MCG forecasting - Errors on the Training Set

Architecture	NRMSE	C.C.	MRE
12-10-1	0.1918	0.9815	0.8421
9-20-1	0.1840	0.9829	0.8073
10-11-1	0.1790	0.9839	0.6790
13-10-1	0.1869	0.9824	0.7875
13-12-1	0.2213	0.9761	0.8983
10-9-1	0.2274	0.9740	0.9702
4-4-1	0.3593	0.9441	0.9822
3-1-1	0.6481	0.8262	0.9061

Table 5. MCG forecasting - Errors on the Test Set

Architecture	NRMSE	C.C.	MRE
12-10-1	0.2419	0.9704	1.4222
9-20-1	0.2081	0.9781	1.1874
10-11-1	0.2214	0.9752	1.3642
13-10-1	0.1858	0.9827	0.9617
13-12-1	0.2163	0.9774	0.8677
10-9-1	0.2193	0.9760	0.9882
4-4-1	0.3586	0.9445	0.9817
3-1-1	0.6497	0.8241	0.9915

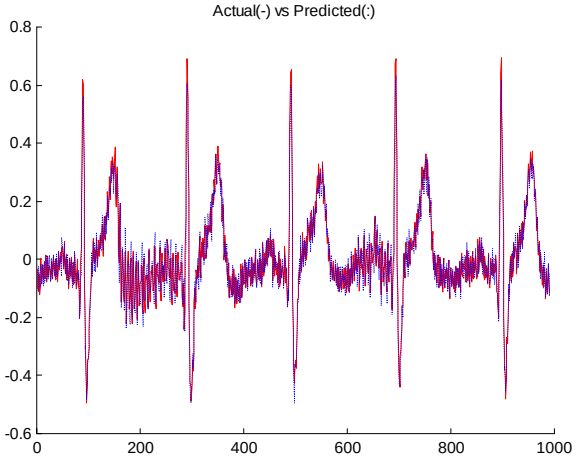


Figure 5. MCG forecasting, performance on the test set.

4 CONCLUSIONS

In the current paper it was presented a hybrid biological inspired evolutionary algorithm that combines a genetic programming technique with a training method based on the Multiple Extended Kalman Algorithm. This hybrid algorithm is implemented in C++ as a software system with a graphical user interface.

The main novelties of the proposed hybrid algorithm are the combination of genetic programming technique with MEKA, the use of a fitness function that combines performance with network size, the ability to evolve not only the structure of the hidden layers but the number of inputs as well, and the large number of different genetic operators and especially mutation operators that have been implemented. Another novelty is the representation used for neural networks. As said before, every network in the population is represented as a link list of layers and neurons, using the direct encoding scheme. The use of link lists has some certain advantages that have to do mainly with the memory management; you use only the memory that is needed every time and you don't have to allocate a maximum memory size, for maximum network size like other representation schemes. Moreover link lists are dynamic data structures, which it means that the neural network architecture can change dynamically during run time in contrast with other data structures like matrices that in C++ can not change during run time.

This hybrid algorithm was used for the modeling of two biological time series, namely the Magneto Encephalogram (MEG) recordings of epileptic patients and Magneto Cardiogram (MCG) of normal subjects. All the reported cases refer to predictions on recordings of the dynamics of nonlinear systems. In all the performed experiments the algorithm was able to find a near optimum network architecture that gave small prediction errors. Therefore we can conclude that the algorithm is able to produce small and compact networks that learn and generalize well.

The algorithm has only tested on time series prediction problems and it is in our intention to test it on some difficult classification problems as well.

One of the main drawbacks of this kind of algorithms, namely the evolutionary algorithms, hybrid or not, is that they are computational expensive in terms of computer memory and CPU time. Even though the proposed algorithm belongs to this category, the use of MEKA for just a couple of epochs for the training phase of the neural networks and the representation where each member of the population is a network represented as a link list so that there is no need to use encoding and decoding functions for the calculation of network's performance, makes the algorithm less computational expensive than other approaches to the same problem of neural networks evolution.

The algorithm could be further improved by adding some more genetic operators for better and faster local search both to the architecture and weight space and this is going to be one of our future research targets. Furthermore, in the integrated software system there are already implemented a large number of genetic operators whose influence to the performance of the hybrid algorithm needs to be appraised; we need to see which of the three selection schemes, or the many mutation operators give better results. Another future research direction will be the combination of MEKA with other evolutionary techniques like Particle Swarm Optimization and Differential Evolution for neural network evolution.

REFERENCES

[1] Adamopoulos, A., Andreou, A., Georgopoulos, E., Ioannou, N. and Likothanassis, S., "Currency Forecasting Using Recurrently RBF Networks Optimized by Genetic Algorithms", Computational Finance 1997 (CF'97), London Business School, 1997.

[2] Adamopoulos, A., Anninos, P., Likothanassis, S., and Georgopoulos, E. "On the Predictability of MEG of Epileptic Patients Using RBF Networks Evolved with Genetic Algorithms", BIOSIGNAL'98, Brno, Czech Republic, June 23-25, 1998a.

[3] Adamopoulos, A., Georgopoulos E., Manioudakis, G. and Likothanassis, S. "An Evolutionary Method for System Structure Identification Using Neural Networks" *Neural Computation '98,* 1998b.

[4] Adamopoulos, A., G. Georgopoulos, S. Likothanassis and P. Anninos, "Forecasting the MagnetoEngephaloGram (MEG) of Epileptic Patient Using Genetically Optimized Neural Networks", *Genetic and Evolutionary Computation Conference (GECCO'99)*, Orlando, Florida USA, July 14-17, 1999

[5] Andreou, A., Georgopoulos, E., and Likothanassis, S., and Polidoropoulos, P., "Is the Greek foreign exchange-rate market predictable? A comparative study using chaotic dynamics and neural networks", Proceedings of the *Fourth International Conference on Forecasting Financial Markets*, Banque Nationale de Paris and Imperial College, London, 1997.

[6] Andreou, A., Georgopoulos, E., Zombanakis, G. and Likothanassis, S., "Testing Currency Predictability Using An Evolutionary Neural Network Model", Proceedings of the *fifth International Conference on Forecasting Financial Markets*, Banque Nationale de Paris and Imperial College, London, 1998.

[7] Andreou A., Georgopoulos E. and Likothanassis, S. "Exchange Rates Forecasting: A Hybrid Algorithm Based On Genetically Optimized Adaptive Neural Networks", *Computational Economics Journal*, Kluwer Academic Publishers, vol. 20, issue 3, pp. 191 – 210, December 2002

[8] Billings, S. A., and Zheng, G. L. Radial basis function network configuration using genetic algorithms. *Neural Networks*, Vol. 8, pp. 877-890, 1995.

[9] Bornholdt S. and Graudenz, D. General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks*, Vol. 5, pp327 – 334, 1992.

[10] Davis, L. Mapping classifier systems into neural networks. *Proceedings of the 1988 Conference on Neural Information Processing Systems*, Morgan Kaufmann, 1988.

[11] Georgopoulos E., Likothanassis S. and Adamopoulos A., "Evolving Artificial Neural Networks Using Genetic Algorithms", *Neural Network World*, 4/00, pp. 565 – 574, 2000.

[12] Goldberg, D. *Genetic Algorithms in Search Optimization & Machine Learning*, Addison-Wesley 1989.

[13] Happel, B., et al. Design and evolution of modular neural network architectures. *Neural Networks*, Vol. 7, pp. 985 – 1004, 1994.

[14] Haykin, S., "Neural Networks - A Comprehensive Foundation", McMillan College Publishing Company, ch. 6, p.213, New York, 1994.

[15] Holland, J. *Adaptation in Natural and Artificial Systems,* MIT press 1992.

[16] Koza J.R., Genetic programming: on the programming of computers by means of natural selection, MIT Press, Cambridge, MA, 1992.

[17] Likothanassis S. Georgopoulos E. and Fotakis D. (1997). Optimizing the Structure of Neural Networks Using Evolution Techniques. *5th International Conference on Applications of High - Performance Computers in Engineering*, pp. 157-168, Santiago de Compostela, Spain, July.

[18] Likothanassis, S. D., Georgopoulos, E. F., Manioudakis, G. D. and Adamopoulos, A.V., "Currency Forecasting Using Genetically Optimized Neural Networks", *HERCMA* Athens September 1998.

[19] Likothanassis, S. D., Georgopoulos, E. F. "A Novel Method for the Evolution of Neural Networks", *3rd IMACS/IEEE International Conference on Circuits Systems and Computers (CSC'99)*, July 1999.

[20] Michalewicz, Z., "*Genetic Algorithms + Data Structures = Evolution Programs*", Springer-Verlag, 1996.

[21] Miller, G., et al. Designing neural networks using genetic algorithms. *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann 1989.

[22] Mitchell M. *An Introduction to Genetic Algorithms*, MIT Press 1996.

[23] Montana, D. and Davis, L. Training feedforward neural networks using genetic algorithms. *BBN Systems and Technologies*, Cambridge, MA 1989.

[24] Shah, S., Palmieri, F. and Datum, M., "Optimal Filtering Algorithms for Fast Learning in Feed-Forward Neural Networks", *Neural Networks*, Vol. 5, pp. 779-787, 1992.

[25] Whitley, D. Applying genetic algorithms to neural network problems, *International Neural Networks Society*, p.230 1988.

[26] Whitley, D., and Bogart, C. The evolution of connectivity: Pruning neural networks using genetic algorithms. *International Joint Conference on Neural Networks, Washington D.C.,* 1. Hillsdale, NJ: Lawpence Erlbaum, pp. 134-137, 1990.

[27] Whitley, D., and Hanson, T. Optimizing neural networks using faster, more accurate genetic search. *3rd Intern. Conference on Genetic Algorithms, Washington D.C.,* Morgan Kaufmann, pp. 391-396, 1989.

[28] Yao, X. & Liu, Y. A New Evolutionary System for Evolving Artificial Neural Networks, *IEEE Transactions on Neural Networks*, vol. 8, no. 3, 1997.

[29] Yao, X. "Evolving Artificial Neural Networks", *Proceedings of the IEEE*, 87(9):1423:1447, September 1999.

[30] Anninos, P. Jacobson, J. Tsagas, N. Adamopoulos, A. "Spatiotemporal Stationarity of Epileptic Focal Activity Evaluated by Analyzing Magneto Encephalographic (MEG) data and the Theoretical Implications". *Panminerva Med*. 39, 189-201, 1997.

[31] Graning, L.; Yaochu Jin; Sendhoff, B.: Generalization Improvement in Multi-Objective Learning, Neural Networks, IJCNN apos;06. International Joint Conference on Volume , Issue , 0-0 0 Page(s):4839 – 4846, 2006.

[32] Vieira, D. A. G. and J. A. Vasconcelos and W. M. Caminhas: Controlling the parallel layer perceptron complexity using a multiobjective learning algorithm, Neural Computing and Applications, vol. 16, n. 4, p.p. 317—325, 2007.

Intelligently Raising Academic Performance Alerts

Dimitris Kalles[1], Christos Pierrakeas and Michalis Xenos

Abstract. We use decision trees and genetic algorithms to analyze the academic performance of students and the homogeneity of tutoring teams in the undergraduate program on Informatics at the Hellenic Open University (HOU). Based on the accuracy of the generated rules, we examine the applicability of the techniques at large and reflect on how one can deploy such techniques in academic performance alert systems.

1 INTRODUCTION

Student success is a natural performance indicator in universities. However, if that success is used as a criterion for tutor assessment (and subsequent possible contract renewal), and if students must evaluate their own teachers, then tutors may tend to lax their standards. This paper is about dealing with this issue in the context of the Hellenic Open University (HOU); we focus on the undergraduate Informatics program (about 2,500 students). We ask whether we can detect regularities in distance tutoring, then, we try to associate them with measures of students' success in an objective way and, subsequently, reflect on how to effectively disseminate this information to all interested parties.

The measurement strategy we have developed to-date in HOU has been progressively refined to deal with two closely linked problems: that of predicting student success in the final exams and that of analyzing whether some specific tutoring practices have any effect on the performance of students. Each problem gives rise to the emergence of a different type of user model. A student model allows us, in principle, to explain and maybe predict why some students fail in the exams while others succeed. A tutor model allows us to infer the extent to which a group of tutors diffuses its collective capacity effectively into the student population they advise. However, both types of models can be subsequently interpreted in terms of the effectiveness of the educational system that the university implements.

The rest of this paper is organised in five sections. The next section presents the educational background. Section 3 then reviews the fundamental features of the AI techniques that we have used. Following that we report the experimental results for the undergraduate programme that we have analysed, as well as a short evaluation of the individual module results that seem to signify an interesting deviation. Section 5 presents a discussion from the point of view of how one can generalise our approach as well as how one can substitute other intelligent techniques for data analysis; finally we conclude and describe directions for future development.

2 THE EDUCATIONAL BACKGROUND

A module is the basic educational unit at HOU. It runs for about ten months and is the equivalent of about 3-4 conventional university semester courses. A student may register with up to three modules per year. For each module, a student is expected to attend five plenary class meetings throughout the academic year. A typical class contains about thirty students and is assigned to a tutor (tutors of classes of the same module collaborate on various course aspects). Class face-to-face meetings are about four hours long and are structured along tutor presentations, group-work and review of homework. Furthermore, each student must turn in some written assignments (typically four or six), which contribute towards the final grade, before sitting a written exam. That exam is delivered in two stages: you only need sit the second if you fail or miss the first.

Students fail a module and may not sit the written exam if they do not achieve a pass grade in the assignments they turn in; these students must repeat that module afresh. A student who only fails the written exam may sit it on the following academic year (without having to turn in assignments); such "virtual" students are also assigned to student groups but the tutor is only responsible for marking their exam papers.

3 GENETIC ALGORITHMS AND DECISION TREES FOR PREDICTION

In our work we have relied on decision trees to produce performance models. Decision trees can be considered as rule representations that, besides being accurate, can produce comprehensible output, which can be also evaluated from a qualitative point of view [1, 2]. In a decision tree nodes contain *test attributes* and leaves contain *class descriptors*.

A decision tree for the (student) exam success analysis problem could look like the one in Figure 1 and tells us that a mediocre grade at the second assignment (root) is an indicator of possible failure (left branch) at the exams, whereas a non-mediocre grade refers the alert to the fourth (last) assignment.

Decision trees are usually produced by analyzing the structure of examples (*training instances*), which are given in a tabular form. An excerpt of a training set that could have produced such a tree is shown in Table 1. Note that the three examples shown are consistent with the decision tree. As this may not always be the case, there rises the need to measure accuracy, even on the training set, in order to compare the quality of two decision trees which offer competing explanations for the same data set.

[1] All authors are with Hellenic Open University, www.eap.gr. Contact address is dkalles@acm.org.

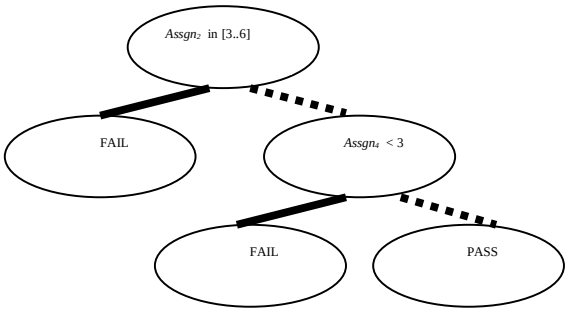


Figure 1. A sample decision tree [3].

Note that the sample decision tree does not utilize data neither on the first nor the third assignments, but such data is shown in the associated table. Such *dimensionality reduction* information is typical of why decision trees are useful; if we consistently derive trees on some problem that seem to not use some data column, we feel quite safe to not collect measurements for that data column. Of course, simple correlation could also deliver such information, however it is the visual representation advantages of decision trees that have rendered them as very popular data analysis tools.

Table 1. A sample decision tree training set (adapted from [3]).

$Assgn_1$	$Assgn_2$	$Assgn_3$	$Assgn_4$	Exam
...	...	...	...	...
4.6	7.1	3.8	9.1	PASS
9.1	5.1	4.6	3.8	FAIL
7.6	7.1	5.8	6.1	PASS

Analyzing the performance of high-risk students is a goal towards achieving tutoring excellence. It is, thus, reasonable to assert that predicting a student's performance can enable a tutor to take early remedial measures by providing more focused coaching, especially in issues such as priority setting and time management.

Initial experimentation at HOU [4] consisted of using several machine learning techniques to predict student performance with reference to the final examination. The scope of the experimentation was to investigate the effectiveness and efficiency of machine learning techniques in such a context. The WEKA toolkit [5] was used because it supports a diverse collection of techniques. The key result was that learning algorithms could enable tutors to predict student performance with satisfying accuracy long before final examination. The key finding that lead to that result was that success in the initial written assignments is a strong indicator of success in the examination. Furthermore, our tutoring experience corroborates that finding.

We then employed the GATREE system [6] as the tool of choice for our experiments, to progressively set and test hypotheses of increasing complexity based on the data sets that were available from the university registry. The formation and development of these tests is the core content of this chapter and is presented and discussed in detail in the following sections. GATREE is a decision tree builder that employs genetic algorithms to evolve populations of decision trees; it was eventually used because it produces short comprehensible trees. Of course, GATREE was first used [3] to confirm the qualitative validity of the original findings experiments [4], also serving as result replication, before advancing to more elaborate experiments [7, 8, 9].

GATREE [6] evolves populations of trees according to a fitness function that allows for fine-tuning decision tree size vs. accuracy on the training set. At each *generation*, a certain *population* of decision trees is generated and sorted according to *fitness*. Based on that ordering, certain *genetic operators* are performed on some members of the population to produce a new population. For example, a mutation may modify the test attribute at a node or the class label at a leaf, while a cross-over may exchange parts between decision trees.

The fitness function is $fitness_i=Correct_i^2*x/(size_i^2+x)$, for tree i. The first part of the product is the actual number of training instances that i classifies correctly. The second part of the product (the size factor) includes a factor x which regulates the relative contribution of the tree size into the overall fitness; thus, the payoff is greater for smaller trees

When using GATREE, we used the default settings for the genetic algorithm operations and set cross-over probability at 0.99 and mutation probability at 0.01. Moreover, *all but the simplest* experiments (explicitly so identified in the following sections) were carried out using 10-fold cross-validation, on which all averages are based (i.e. one-tenth of the training set was reserved for testing purposes and the model was built by training on the remaining nine-tenths; furthermore, ten such stages were carried out by rotating the testing one-tenth.

4 DATA ANALYSIS AT A PROGRAMME LEVEL

Before advancing, we first review some aggregate statistics of the undergraduate informatics programme at HOU.

First, Table 2 presents the success rates for the modules that we have analysed.

Table 2. Success (percentage) rates of modules.

	2004-5	2005-6	2006-7
INF10	35%	38%	33%
INF11	55%	52%	55%
INF12	39%	34%	35%
INF20	56%	44%	44%
INF21	37%	44%	37%
INF22	71%	61%	55%
INF23	N/A	83%	97%
INF24	70%	64%	58%
INF30	81%	85%	84%
INF31	93%	92%	85%
INF35	N/A	98%	93%
INF37	N/A	98%	98%
INF42	N/A	N/A	100%

Next, Table 3 presents the enrolment numbers for these modules. Note that, as we advance from junior to senior years, the overall enrolment is dramatically reduced and the success rates increase.

Table 3. Enrollment numbers at modules.

	2004-5	2005-6	2006-7
INF10	987	1.247	1.353
INF11	492	517	642
INF12	717	818	925
INF20	362	389	420
INF21	322	363	383
INF22	321	291	321
INF23	N/A	52	73
INF24	157	167	221
INF30	156	198	199
INF31	149	200	144
INF35	N/A	101	58
INF37	N/A	106	132
INF42	N/A	N/A	109

The above statistics are all drawn from the university registry and none is subject to any further processing. However, all results presented from now on, refer to experiments carried out totally using the GATREE system, with the occasional help of some post-processing automation scripts.

4.1 Detecting a shift in exam grades

There is a straightforward way to attempt to answer this question. One can build a model that attempts to answer the success question for the first stage of the final exam. Then, one can build a model that attempts to answer the success question for the overall student grade. A gross disparity in these numbers should be indicative of an issue that merits investigation.

The simplest data to consider as input for this problem consists of exercise and exam grades, as in Table 1, omitting any other information (for example, which tutor was responsible for a student). The results reported are based on re-classification (we reserve a cross-validation like mechanism for the more detailed experiments later on) and are shown in Table 4.

What does a difference signify? To answer that, one can take a step backwards and try to answer a simpler question: what does a large difference signify? We have elected to brand a difference as large when the re-classification accuracy of the same module for the same year differs by at least 20 percentage points when we compare the model predicting the pass/fail result of the first stage of the final exam and the corresponding model after a possible second stage (which is the actual pass/fail grade for the module). In Table 4 such differences are shown in **bold**.

There are two issues that become apparent when one views Table 4. The first is that whenever we observe an increase in the model accuracy when switching from the first exam (E) to the final grade (F), this is associated with senior modules where eventual success rates (see Table 2) are substantial. The only decrease is observed in a junior year module where success rates are considerably reduced compared to senior year modules.

Table 4. Model accuracies omitting tutor data.

	2004-5		2005-6		2006-7	
	E	F	E	F	E	F
INF10	83	84	84	82	83	82
INF11	75	76	76	78	75	80
INF12	74	76	86	74	78	74
INF20	76	70	76	59	**87**	**60**
INF21	83	78	76	72	77	73
INF22	68	80	68	76	63	70
INF23	N/A		**46**	**78**	89	99
INF24	67	67	68	66	69	70
INF30	77	82	**64**	**85**	**71**	**94**
INF31	**65**	**95**	86	93	**68**	**91**
INF35	N/A		**72**	**97**	80	92
INF37	N/A		95	100	95	98
INF42	N/A		N/A		96	100

It is straightforward to attribute the increase in senior year modules to the fact that, eventually, students have to focus on their exam and pass the test, regardless of how well they did along the year. The large discrepancy, however, suggests that the exercises do not serve well their goal, which is to keep the students engaged in the learning process. One could say that exercises are less of learning opportunities and more of necessary evils.

The dramatic decrease in the 2006-7 year results of the INF20 module are quite interesting. They reflect, basically, a huge fail rate in the first stage of the exam, which is well served by a small model that predicts failure all around.

When seen from that viewpoint, however, the relatively narrow margins of the junior year modules seem quite impressive, since they are also associated with low overall pass rates. The difference, however, is that the junior modules also report significant dropout rates which skews pessimistically the rates reported in Table 2.

4.2 Detecting tutor influence

If we take the data sets that were used in section 4.1 and put back in the information on which tutor was responsible for each student group, we can run the same experiments and try to see whether the tutor attribute will surface in some models (sample data are shown in Table 5).

In principle, observing models where the tutor attribute appears near the decision tree root would not be a good thing, suggesting that a crucial factor in student success is not the educational system itself but the tutor. As a matter of fact we can opt to not look for this information at all in the resulting trees; comparing the accuracies to the ones reported in Table 4 should suffice. These results are now shown in Table 6.

Table 5. An expanded sample training set (see Group).

$Assgn_1$	$Assgn_2$	$Assgn_3$	$Assgn_4$	Group	Exam
...	...	...	...	...	...
4.6	7.1	3.8	9.1	Athens-1	PASS
9.1	5.1	4.6	3.8	Patras-1	FAIL
7.6	7.1	5.8	6.1	Athens-2	PASS

Table 6. Model accuracies including tutor data.

	2004-5		2005-6		2006-7	
	E	F	E	F	E	F
INF10	82	83	80	79	82	81
INF11	75	77	76	78	75	80
INF12	75	77	81	72	80	72
INF20	76	72	76	62	87	61
INF21	84	77	74	74	75	72
INF22	66	80	68	74	62	75
INF23	N/A		52	82	90	99
INF24	63	69	69	69	66	74
INF30	75	82	60	88	75	94
INF31	67	94	85	93	**89**	**91**
INF35	N/A		72	98	76	90
INF37	N/A		96	100	94	98
INF42	N/A		N/A		96	100

This time we observe that the relative difference between the models which utilise the tutor attribute and the ones that do not are quite small. There are some very interesting cases, however.

For example, the INF11 module demonstrates near zero differences throughout. It is interesting to note that this module utilizes a plenary exam marking session, which means that tutors get to mark exam papers drawn from all groups at random. This places only marginal administrative overhead and, when viewed from the point of model consistency, seems to be well worth it.

Another example is the INF31 module (shown in **bold**), which demonstrated a year where the tutor attribute seemed to be of paramount importance. In that year, the gap between the first exam stage and the final grade seems to be influenced by the tutors. It is now very narrow (89 to 91) while it was quite wide (68 to 91). This could suggest a relative gap in tutor homogeneity.

There is one other way to view the importance of the tutor attribute. One can derive a model for one module group and then attempt to use that model as a predictor of performance for the other module groups (within the same module). This approach, while suppressing the tutor attribute, essentially tests its importance by specifically segmenting the module data set along groups. The overall accuracy is then averaged over all individual tests. This is the lesion comparison; its results are shown in Table 7.

We highlight (**in bold**) the main difference from the results in Table 4, where it now seems that the gap has been shortened a while. Surprisingly, it suggests an erratic intra-group consistency. Note also, that this particular result in Table 4 was the only one not to pass the binary choice (50%) level, which it only just did in Table 6.

Table 7. Lesion study model accuracies including tutor data.

	2004-5		2005-6		2006-7	
	E	F	E	F	E	F
INF10	78	78	75	75	77	77
INF11	70	74	72	75	71	74
INF12	71	68	77	69	75	71
INF20	70	65	72	60	82	61
INF21	79	68	69	65	69	64
INF22	57	74	61	68	60	65
INF23	N/A		**65**	**74**	83	98
INF24	62	70	66	69	63	66
INF30	70	82	64	84	65	89
INF31	63	91	79	91	59	83
INF35	N/A		66	97	72	91
INF37	N/A		95	98	91	95
INF42	N/A		N/A		93	100

Furthermore, we tried to summarise the results from a further point of view: that of consistency between the results reported for the *E* and *F* columns of both tables. Essentially we computed the quantity $(F_5\text{-}E_5)\text{-}(F_6\text{-}E_6)$ for each module for each year, where the subscript indicates which table that particular number was drawn from. Not surprisingly, the two singularities observed were module INF23 for year 2005-6 (with a value of about 20%) and module INF31 for year 2006-7 (with a value of about -22%).

4.3 Observing the accuracy-size trade-off

It is interesting to investigate whether the conventional wisdom on model characteristics is valid. In particular, we analysed the results in Table 6 and in Table 7 with respect to whether an increase (or decrease, accordingly) in model accuracy for a particular module for a year was associated with a reduction in model size. We say that the model accuracy increases if the accuracy for the *E* column of that year is less that the corresponding number in the *F* column. For the 68 pairs of numbers reported in Table 6 and in Table 7 we observed that only in 4 of them did we see the same direction in model accuracy and model size. So, conventional wisdom was confirmed in nearly 95% of the cases.

5 DISCUSSION

HOU has been the first university in Greece to operate, from its very first year, a comprehensive assessment scheme (on tutoring and administrative services). Despite a rather hostile political environment (at least in Greece), quite a few academic departments have lately been moving along the direction of introducing such schemes, though the practice has yet to be adapted at a university level. Still, however, there is quite a mentality shift required when considering the subtle differences between “measuring” and “assessing”.

The act of measuring introduces some error in what is being measured. If indices are interpreted as assessment indices, then people (actually, any “assessed” subject where people are involved – groups of people, for example) will gradually skew their behaviour towards achieving good measurements. Such behaviour is quite predictably human, of

course; the problem is that it simply educates people in the ropes of the measurement system while sidelining the real issue of improving the educational service.

By shifting measurement to quantities that are difficult to "tweak", one hopes that people whose performance is assessed will gradually shift form fine-tuning their short-term behaviour toward achieving longer-term goals. Indeed, if people find out that the marginal gains from fine-tuning their behaviour are too small for the effort expended to achieve them, it will be easier to convince them to improve more fundamental attitudes towards tutoring (as far as tutors are concerned) or studying (as far as students are concerned).

In our application, this is demonstrated two-fold.

First, by disseminating tutor group homogeneity indices, one hopes that, regardless how we call these indices, these tutor groups will be motivated by peer pressure to consider their performance vis-a-vis other tutor groups. Even if that may not be really required, that introspection itself will quite likely improve how hat particular tutor group co-operates; at least it will focus their decisions with respect to why such decisions might influence their overall ranking.

For students, a similar argument applies. Realising that one fits a model which predicts likely failure, even if one knows that the particular model is known to err quite some times, is something that will most likely motivate that person to take a more decisive approach to studying. For adult students, such a decisive approach might even mean to drop a course of studying or defer studying. This is not necessarily negative, however; knowing how to better utilise one's resources is a key skill in life long learning.

We have selected decision trees because we want to generate models that can be effectively communicated to tutors and students alike. We have also selected genetic algorithms to induce the decision trees because we have shown [7] that, for the particular application domain, we can derive small and easy to communicate yet accurate trees. We thus need a hybrid approach: rule-based output to be comprehensible and grounded and evolutionary computing to derive this output.

Which other techniques should one utilise to develop the models? We cannot fail to note that conventional statistics can be cumbersome to disseminate to people with a background on humanities or arts, and this could have an adverse impact on the user acceptance of such systems. In that sense, the decision of whether the models are computed centrally or in a decentralized fashion (by devolving responsibility to the tutors, for example) is a key factor. In any case, deploying our measurement scheme in an organization-wide context would also lend support to our initial preference for short models. At the same time, the possibility of a decentralized scheme also suggests that we should strive to use tools that do not demand a steep learning curve on the part of the tutors.

Of course, one can take an alternative course and drop the requirement that a model has to be communicated. If we only focus on the indices then any technique can be used, from neural networks to more conventional ones, such as naive Bayes or logistic regression [4]. As in all data mining application contexts, it is the application that must drive the techniques to use; for our problem, suffice to note that the comparisons reported (Table 4, Table 6 and Table 7) are essentially technique independent, yet the GATREE approach has proven to-date to be the best method for prototyping the measurement exercise that we are developing.

6 CONCLUSION

We have shown how we have used a combination of genetic algorithms and decision trees in the context of experimenting with how one might setup a quality control system in an educational context.

Quality control should be a core aspect of any educational system but setting up a system for quality control entails managerial and administrative decisions that may also have to deal with political side-effects. Deciding how to best and as early as possible defuse the potential stand-offs that a quality measurement message might trigger calls for the employment of techniques that not only ensure a basic technical soundness in the actual measurement but also cater to the way the results are conveyed and subsequently exploited. This is particularly so when the application context for the large scale suggests that data and models will freely flow amongst thousands of tutors and tens of thousands of students.

We have earlier [9] expressed the view that our approach is applicable to any educational setting where performance measurement can be cast in terms of test (and exam) performance. In the proposed paper we have scaled up our analysis to cover several modules and years and still believe that taking the sting out of individual performance evaluation but still being able to convey the full message is a key component of tutoring self-improvement. Scaling our approach to other programmes, other institutions and, even, obtaining the approval of our own university for official and consistent reporting of such indices is, however, less of a technical nature and more of a political exercise. After all we need to persuade people that some innovations are less of a threat and more of an opportunity.

ACKNOWLEDGEMENTS

Thanassis Hadzilacos (now at the Open University of Cyprus) has contributed to this line of research while at Hellenic Open University.

Anonymized data can be available on request for research purposes only, on a case-by-case basis.

We acknowledge the advice, from an anonymous reviewer of the CIMA-ECAI08 workshop, on how to improve the presentation of this work to reflect the combination of AI techniques used.

REFERENCES

[1] Mitchell, T. (1997). Machine Learning. McGraw Hill.

[2] Quinlan, J.R (1993). C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.

[3] Kalles, D., & Ch. Pierrakeas (2006). Analyzing student performance in distance learning with genetic algorithms and decision trees. Applied Artificial Intelligence, 20(8), pp. 655-674.

[4] Kotsiantis, S., Pierrakeas, C., & Pintelas, P. (2004). Predicting students' performance in distance learning using Machine Learning techniques. Applied Artificial Intelligence, 18:5, 411-426.

[5] Witten, I., & Frank, E. (2000). Data mining: practical machine learning tools and techniques with Java implementations. San Mateo, CA: Morgan Kaufmann.

[6] Papagelis, A., & D. Kalles (2001). Breeding decision trees using evolutionary techniques. Proceedings of the International Conference on Machine Learning, Williamstown, Massachusetts, pp. 393-400, Morgan Kaufmann.

[7] Kalles, D., & Ch. Pierrakeas. (2006). Using Genetic Algorithms and Decision Trees for a posteriori Analysis and Evaluation of Tutoring Practices based on Student Failure Models. Proceedings of the 3rd IFIP conference on Artificial Intelligence Applications and Innovations, Athens, Greece, pp. 9-18, Springer.

[8] Hadzilacos, Th., Kalles, D. Pierrakeas, Ch., & M. Xenos (2006). On Small Data Sets Revealing Big Differences. Proceedings of the 4th Panhellenic conference on Artificial Intelligence, Heraklion, Greece, Springer LNCS 3955, pp. 512-515.

[9] Hadzilacos, Th., & D. Kalles (2006). On the Software Engineering Aspects of Educational Intelligence. Proceedings of the 10th International Conference on Knowledge-Based Intelligent Information & Engineering Systems, Bournemouth, UK, Springer LNCS 4252, pp. 1136-1143.

[10] Xenos, M., Pierrakeas, C., & Pintelas, P. (2002). A survey on student dropout rates and dropout causes concerning the students in the Course of Informatics of the Hellenic Open University. Computers & Education, 39, 361-377.

Recognizing predictive patterns in chaotic maps

Nicos G. Pavlidis[1], **Adam Adamopoulos**[2] and **Michael N. Vrahatis**[3]

Abstract. We investigate the existence of rules (in the form of binary patterns) that allow the short-term prediction of highly complex binary sequences. We also study the extent to which these rules retain their predictive power when the sequence is contaminated with noise. Complex binary sequences are derived by applying two binary transformations on real-valued sequences generated by the well known tent map. To identify short-term predictors we employ Genetic Algorithms. The dynamics of the tent map depend strongly on the value of the *control parameter*, r. The experimental results suggest that the same is true for the number of predictors. Despite the chaotic nature of the tent map and the complexity of the derived binary sequences, the results reported suggest that there exist settings in which an unexpectedly large number of predictive rules exists. Furthermore, rules that permit the risk free prediction of the value of the next bit are detected in a wide range of parameter settings. By incorporating noise in the data generating process, the rules that allow the risk free prediction of the next bit are eliminated. However, for small values of the variance of the Gaussian noise term there exist rules that retain much of their predictive power.

1 Introduction

In this paper we consider the problem of identifying rules, in the form of binary patterns, that are perfect, or in the worst case good, short-term predictors of complex binary sequences. A binary pattern of length L is defined as *perfect short-term predictor* if its presence in any place of the binary sequence is declarative of the value of the next bit. By definition, perfect predictors, enable the risk-free prediction of the next bit. Similarly, *good short-term predictors*, are binary patterns whose appearance in any position of the binary sequence renders the value of the next bit highly predictable.

Complex binary sequences are derived through the application of binary transformations on real–valued data sequences obtained from the tent map. The tent map is a piecewise-linear, continuous map on the unit interval $[0, 1]$ into itself:

$$f_r(x) = \begin{cases} rx, & x \in [0, 1/2] \\ r(1-x), & x \in (1/2, 1] \end{cases}, \qquad (1)$$

where r is a *control parameter* that assumes values in the interval $[0, 2]$. We consider a discrete process generated by:

$$x_{n+1} = f_r(x_n) = \underbrace{f_r\left(f_r\left(\ldots\right)\right)}_{(n+1) \text{ times}} = f_r^{(n+1)}(x_0), \quad n = 0, 1, \ldots, \qquad (2)$$

where $f_r^{(n)}$ denotes the nth iterate of f_r. The Lyapunov exponent is given by:

$$\lambda_r(x) = \lim_{n\to\infty} \frac{1}{n} \ln \left| \frac{d}{dx} f_r^{(n)}(x) \right| = \ln r,$$

everywhere in $[0, 1]$. For $r \in (0, 1)$, the orbit, $f_r^{(n)}(x_0)$, for any $x_0 \in [0, 1]$ converges to the unique fixed point 0, as n increases. For $r = 1$, every point $x \in [0, 1/2]$ is a fixed point. The chaotic region is $1 < r \leqslant 2$, in which $\lambda_r > 0$ [5]. For $r > 1$ the map has two unstable fixed points, one at 0 and the other at $x^*(r) = r/(r+1)$. Using the notation in [5], we write, $x_n(r) \equiv f_r^{(n)}(1/2)$. Then $x_1(r) = r/2$ and $x_2(r) = r(1-r/2)$. The intervals, $(0, x_2(r))$ and $(x_1(r), 1)$ are transient for f_r, and we have $f_r A = A$ for $A = [x_2(r), x_1(r)]$. If $r \in \left(\sqrt{2}, 2\right]$, then A is an attractor. At $r = \sqrt{2}$, the attractor A splits into two bands, A_0 and A_1, at the position $x = x^*(r)$. For $r \in \left(1, \sqrt{2}\right]$ we have $f_r(A_0) = A_1$ and $f_r(A_1) = A_0$. Similarly, at $r^2 = \sqrt{2}$, each of the two bands splits into two bands, $A_{ij}(i, j = 0, 1)$. In this manner, as r decreases, band splitting occurs successively at $r = \overline{r}_1, \overline{r}_2, \ldots, \overline{r}_m, \ldots$, where $\overline{r}_m = 2^{1/2^m}$, and $m = 1, 2, \ldots$. By setting $\overline{r}_0 = 2$, then, for $\overline{r}_{m+1} < r < \overline{r}_m$, there exist 2^m disjoint intervals $A_{i_1,i_2,\ldots,i_m}$, $i_k = (0, 1)$ in which the invariant density is positive (the 2^m-band regime). Defining, $l = 1+i_1+2i_2+\cdots+2^{m-1}i_m$, and $J_l \equiv A_{i_1,i_2,\ldots,i_m}$, it is shown in [5] that $f_r(J_l) = J_{l+1}$ for $1 \leqslant l \leqslant 2^m - 1$, and $f_r(J_M) = J_1$, where $M = 2^m$. Therefore, if r lies in the interval $\left(1, \sqrt{2}\,\right]$, f_r maps a set of intervals between $r - r^2/2$ and $r/2$ to themselves. If, on the other hand, $r > \sqrt{2}$ these intervals merge. This is illustrated in the bifurcation diagram of Fig. 1.

Real-world time series are frequently contaminated by noise. To this end, we investigate the resilience of the predictors to the presence of noise in the data generating process. We include an additive Gaussian noise term with zero mean, to Eq. (1), and study the extent to which the predictors detected in the original sequences retain their predictive power for different values of the variance of the distribution.

2 Methods

The tent map, described in Eq. (1), was employed to generate raw data sequences $x_n(x_0, r)$. To generate the raw data

[1] Institute for Mathematical Sciences, Imperial College London, South Kensington Campus, London SW7 2AZ, United Kingdom, email: n.pavlidis@imperial.ac.uk
[2] Medical Physics Laboratory, Department of Medicine, Democritus University of Thrace, Alexandroupolis GR-68100, Greece, email: adam@med.duth.gr
[3] Department of Mathematics, University of Patras, Patras GR-26110, Greece, email:vrahatis@math.upatras.gr

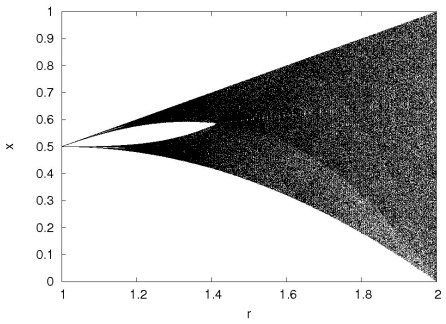


Figure 1. Bifurcation diagram of the steady states of the tent map with respect to r.

from the tent map the GNU Multiple Precision Arithmetic Library (GMP) [1] was utilized to generate floating point numbers with precision of at least 5000 bits. Subsequently, binary data sequences $b_n(x_0, r)$ were produced by applying the simple, threshold, binary transformation originally proposed for the logistic equation in [4]:

$$b_n(x_0, r) = \begin{cases} 0, & \text{if } x_n \leqslant 0.5, \\ 1, & \text{if } x_n > 0.5. \end{cases} \quad (3)$$

Eq. (3) produces a bit with value '1' when the value of the tent map is greater that 0.5 and a bit with value '0' otherwise. To avoid transient phenomena, the first 10^4 iterations of the map were discarded. A number of real-valued sequences $x_n(x_0, r)$ were generated through Eq. (1) for different values of the control parameter, r, and starting points, x_0. Binary sequences, $b_n(x_0, r)$, of 10^6 bits were produced by applying Eq. (3) on the raw data, $x_n(x_0, r)$.

A second binary transformation, also proposed in [4] for the logistic equation, was applied on the raw data. This transformation is also a simple, linear, threshold binary transformation, but with a variable threshold. The threshold value is the previous value of the raw data of the tent map. Hence, the second transformation is formulated as:

$$b_n(x_0, r) = \begin{cases} 0, & \text{if } x_n \leqslant x_{n-1}, \\ 1, & \text{if } x_n > x_{n-1}. \end{cases} \quad (4)$$

The number of all possible patterns of length L, 2^L, increases exponentially with respect to L. For large values of L, therefore, it is infeasible to perform exhaustive search, and more efficient search methods, such as Genetic Algorithms (GAs), are required [2, 3]. To this end, a simple GA with binary representation was implemented and utilized. The GA population consisted of L–bit patterns. The fitness of a pattern p, was the number of times p was encountered in the binary sequence $b_n(x_0, r)$. The selection process used was *roulette wheel selection*. As crossover operator the well–known *one–point crossover operator* was employed. Finally, the mutation operator utilized was the *flip bit mutation operator*. GAs were applied for several values of L and a number of binary sequences, $b_n(x_0, r)$. Consequently, patterns that can account as perfect, or good, predictors can be identified by comparing the obtained results for L–bit and $(L+1)$–bit patterns.

3 Presentation of Results

3.1 Fixed threshold

In the following, we present indicative results for binary sequences of length 10^6, obtained by applying the transformation of Eq. (3). In Fig. 2 the distribution of bits with value '1' and '0' for different values of r is plotted. Evidently, an equal distribution of the two occurs only as r tends to 2.

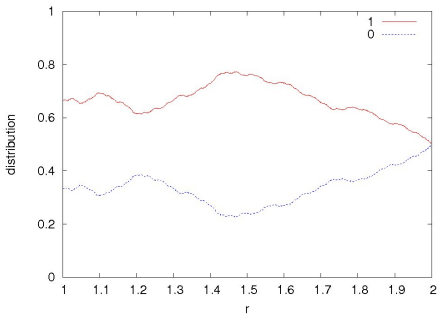


Figure 2. Distribution of ones (dashed) and zeros (solid) according to the transformation of Eq. (3) for $b_n(0.1, r)$ and $r \in [1, 2]$ with stepsize 10^{-3}.

The number of distinct patterns of length L that appear for different values of r, is reported in Table 1. In detail, the first column of Table 1 reports the value of r; the second column indicates the length of the binary patterns L; the third column corresponds to the number of different patterns of length L identified in each binary sequence (#f); and finally the fourth column reports the ratio of the number of patterns of length L found (#f) to the number of possible binary patterns of this length (2^L). The lower the ratio shown in the last column of the table the fewer the patterns that appear in the binary sequence and hence the higher the predictability.

An inspection of Table 1 suggests that increasing the value of r, gradually increases the number of patterns that are encountered for each value of L and hence degrades predictability. This effect becomes clear by comparing the results for $r = 1.44$ and $r = 1.999$. For $r = 1.44$ and $L = 2$, already the ratio of appearing to all possible patterns is 0.75 suggesting that one out of the four possible patterns is absent. This ratio decreases as L increases to reach 0.091 for $L = 9$ indicating that less than 10% of all possible patterns of this length are present in the sequence $b_{10^6}(0.1, 1.44)$. On the contrary, for $r = 1.999$ all possible patterns appear for all the different values of L up to and including $L = 9$. It should be noted that for $r = 1.999$ and $L = 10$ the ratio of column four becomes less than unity, but still its value is very close to that, 0.999, suggesting that even in this case increasing L reduces the ratio but this effect takes place very slowly.

Next, the impact of introducing noise to the data generating process is investigated. A normally distributed, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, additive noise term was included in the tent map equation, yielding $x_{n+1} = f_r(x_n) + \varepsilon$, where $f_r(x_n)$ is given by Eq. (1). It should be noted that we enforced the resulting raw data series to lie in the interval $[0, 1]$ by rejecting realizations of the noise term that would result in $x_{n+1} \notin [0, 1]$. The obtained experimental results for the most predictable binary sequence

Table 1. Number of patterns in $b_{10^6}(0.1, r)$ obtained through the transformation of Eq. (3) for different values of r.

r	L	#f	#f/2^L
1.44	2	3	0.750
	3	5	0.625
	4	7	0.437
	5	11	0.343
	6	15	0.234
	7	23	0.179
	8	31	0.121
	9	47	0.091
1.5	2	3	0.750
	3	5	0.625
	4	7	0.437
	5	11	0.343
	6	16	0.250
	7	25	0.195
	8	37	0.144
	9	57	0.111
1.6	2	3	0.750
	3	5	0.625
	4	8	0.500
	5	13	0.406
	6	21	0.328
	7	34	0.265
	8	55	0.214
	9	88	0.171
1.7	2	4	1.000
	3	7	0.875
	4	12	0.750
	5	21	0.656
1.7	6	36	0.562
	7	61	0.476
	8	105	0.410
	9	179	0.349
1.8	2	4	1.000
	3	7	0.875
	4	13	0.812
	5	24	0.750
	6	43	0.671
	7	78	0.609
	8	141	0.550
	9	253	0.494
1.9	2	4	1.000
	3	8	1.000
	4	15	0.937
	5	29	0.906
	6	55	0.859
	7	105	0.820
	8	199	0.777
	9	379	0.740
1.999	2	4	1.000
	3	8	1.000
	4	16	1.000
	5	32	1.000
	6	64	1.000
	7	128	1.000
	8	256	1.000
	9	512	1.000

when no noise is included, $b_{10^6}(0.1, 1.44)$, are summarised in Table 2. The first column of the table corresponds to the pattern length L; the second lists all the possible binary patterns of length L (due to space limitations, only patterns of length up to four are included); while columns three to six report the number of occurrences of each pattern for different values of the variance, σ^2, starting with the case of no noise ($\sigma^2 = 0$).

Starting from the case of no noise, we observe that more than three quarters of the binary sequence consists of bits with value '1'. Furthermore, from the patterns with length two, the pattern '00' is missing, indicating that a '0' is always followed by a '1'. This fact renders the unit length pattern '0' (and consequently all patterns of any length ending with a '0') a perfect predictor, and hence approximately 23% of the sequence is perfectly predictable. The inclusion of the additive noise term distorts these findings gradually as the variance increases. For $\sigma^2 = 0.01$ findings are marginally altered as the length two pattern '00' appears only 17 times in the length 10^6 binary sequence. Thus, the probability of a '1' following a bit with value '0' is 0.99993. For $\sigma^2 = 0.1$ and $\sigma^2 = 0.5$ this probability becomes 0.56109 and 0.44146 respectively. In the case of $\sigma^2 = 0.5$, therefore, the impact of noise is so large that the original finding is reversed and a '0' is more likely to be followed by a '0'. The fact that increasing the variance of the noise term deteriorates the predictability of the binary sequence is also evident from the fact that patterns that did not appear in the not contaminated with noise sequence, appear frequently in the contaminated series. The predictive power of the binary pattern '0' (perfect predictors in the noise-free binary sequence) with respect to the value of the variance of the additive noise term, σ^2 is illustrated in Fig. 3. To generate Fig. 3, σ^2 assumed values in the interval $[0, 0.5]$ with stepsize 10^{-3}.

Table 2. Patterns in $b_{10^6}(0.1, 1.44)$ obtained through the transformation of Eq. (3) and different values of σ^2.

L	patterns	$\sigma^2 = 0.0$	$\sigma^2 = 0.01$	$\sigma^2 = 0.1$	$\sigma^2 = 0.5$
1	0	230307	246757	434808	552264
	1	769693	753243	565192	447736
2	00	0	17	190252	308874
	01	231742	246799	243209	244133
	10	231742	246799	243209	244133
	11	536514	506383	323328	202858
3	000	0	0	93237	173043
	001	0	17	97015	135831
	010	112119	119901	108060	133112
	011	119623	126897	135149	111021
	100	0	17	97015	135831
	101	231742	246782	146194	108301
	110	119623	126898	135148	111021
	111	416890	379485	188179	91837
4	0000	0	0	50612	96905
	0001	0	0	42625	76138
	0010	0	17	43068	74254
	0011	0	0	53947	61577
	0100	0	9	44101	74116
	0101	112119	119892	63959	58995
	0110	0	2915	55812	60753
	0111	119622	123982	79336	50268
	1000	0	0	42625	76138
	1001	0	17	54390	59693
	1010	112119	119884	64992	58858
	1011	119623	126897	81202	49443
	1100	0	8	52914	61715
	1101	119623	126890	82234	49306
	1110	119623	123983	79336	50268
	1111	297267	255502	108843	41569

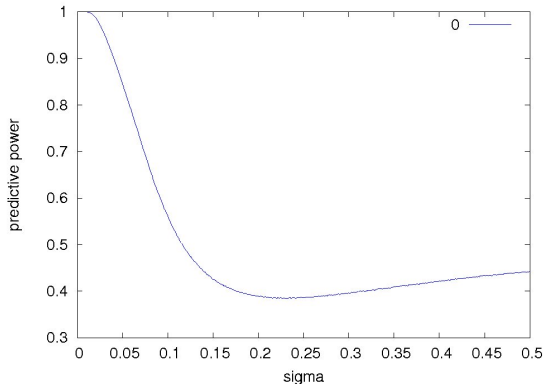


Figure 3. Predictive power of the unit length binary pattern '0' in the sequences obtained through the transformation of Eq (3) with respect to the variance σ^2 of the noise term.

3.2 Variable threshold

In this subsection we present results from the analysis of binary sequences derived by applying the transformation of Eq. (4), according to which the threshold is equal to the previous value of the tent map. The distribution of bits with value '1' and '0' for different values of the control parameter r is illustrated in Fig. 4. Comparing Figs. 2 and 4 it is evident that the two transformations yield substantially different distributions of ones and zeros. For the second transformation, the proportion of ones exceeds that of zeros for r marginally larger than 1. As shown in Fig. 4 the two proportions are

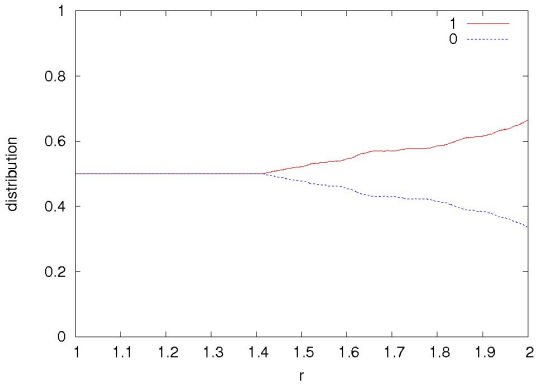


Figure 4. Distribution of ones (dashed) and zeros (solid) according to the transformation of Eq. (4) for $b_n(0.1, r)$ and $r \in [1, 2]$ with stepsize 10^{-3}.

equal until r becomes equal to $\sqrt{2}$. This finding is attributed to the band splitting phenomenon, briefly described in Section 1, that occurs for $r \in (1, \sqrt{2}]$ [5]. From that point and onward their difference increases.

The number of patterns of different length L that appear in the binary sequences of length 10^6, are reported in Table 3 for different values of the control parameter r. More specifically, the first column of Table 3 reports the value of r; the second column corresponds to the length L of the binary patterns; the third column reports the number of different patterns of length L that were identified in the sequence (#f); and lastly, column four depicts the proportion of the patterns encountered (#f) to the number of possible binary patterns of length L (2^L).

As in the case of the fixed threshold binary transformation, increasing the value of the control parameter r increases the number of patterns that appear in the derived binary sequences. However, this effect is more pronounced for the fixed threshold transformation of Eq. (3) than for the variable threshold transformation of Eq. (4). Even for $r = 1.999$, Table 3 reports that the number of binary patterns of length two is three, suggesting that one pattern of length two does not appear, and hence a unit length perfect binary predictor exists. In contrast, for the fixed threshold binary transformation, Table 1, all four length two binary patterns are present in the sequences that are generated with $r \geqslant 1.7$. Moreover, the ratio of the patterns of length L found to the number of possible patterns of this length decreases more rapidly in the sequences generated by the variable threshold transformation. For instance, for $r = 1.44$, the number of patterns of length nine is 47 for the fixed threshold transformation, while for the variable threshold transformation this number is 10.

The impact of introducing noise on the short-term predictors is studied next. Table 4 reports the patterns of length two to four that were encountered in the binary sequence $b_{10^6}(0.1, 1.44)$ that was obtained through the second transformation, for different values of σ^2. In detail, the first column of Table 4 corresponds to the length L of the patterns; the second column lists all possible binary patterns of this length; while columns three to six report the number of occurrences of each pattern in the binary sequences obtained for different values

Table 3. Number of patterns in $b_{10^6}(0.1, r)$ obtained through the transformation of Eq. (4) for different values of r.

r	L	#f	#f/2^L
1.44	2	3	0.750
	3	4	0.500
	4	5	0.312
	5	6	0.187
	6	7	0.109
	7	8	0.062
	8	9	0.035
	9	10	0.019
1.5	2	3	0.750
	3	4	0.500
	4	5	0.312
	5	6	0.187
	6	7	0.109
	7	8	0.062
	8	9	0.035
	9	11	0.021
1.6	2	3	0.750
	3	4	0.500
	4	5	0.312
	5	7	0.218
	6	9	0.140
	7	12	0.093
	8	15	0.058
	9	19	0.037
1.7	2	3	0.750
	3	4	0.500
	4	5	0.312
	5	7	0.218
1.7	6	9	0.140
	7	12	0.093
	8	16	0.062
	9	21	0.041
1.8	2	3	0.750
	3	5	0.625
	4	7	0.437
	5	10	0.312
	6	14	0.218
	7	19	0.148
	8	27	0.105
	9	38	0.074
1.9	2	3	0.750
	3	5	0.625
	4	8	0.500
	5	12	0.375
	6	18	0.281
	7	27	0.210
	8	40	0.156
	9	59	0.115
1.999	2	3	0.750
	3	5	0.625
	4	8	0.500
	5	13	0.406
	6	21	0.328
	7	34	0.265
	8	55	0.214
	9	89	0.173

of the variance of the additive noise term $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Note that as in the previous case, the resulting raw data sequence $\{x_n\}_{n=0}^{10^6}$ was restrained in the interval $[0, 1]$ by rejecting realizations of the noise term that would result in $x_n \notin [0, 1]$.

Starting from the case of no noise, we observe that zeros and ones are approximately equally distributed in the binary sequence. As in the case of the first transformation, pattern '00' is missing from the patterns of length two, a finding which implies that a '0' is always followed by a '1', and hence all the binary patterns of any length that end with '0' are perfect predictors. Furthermore, the pattern '111' was not encountered, implying that '11' is always followed by a '0'. From the inspection of the findings for patterns of length three we also obtain a good predictor of length two, namely the pattern '01', for which the probability of appearance of '0' immediately after this pattern is 0.96919. Comparing the aforementioned findings for the case of no noise, with the corresponding ones obtained by the first, fixed threshold, transformation we conclude that the binary sequence obtained through the variable threshold transformation is more predictable.

As expected, the introduction of noise eliminates all the perfect predictors identified in the original binary sequence. For a low value of the variance, $\sigma^2 = 0.01$, the findings are marginally distorted. The previously not encountered pattern '00' appears 5979 times yielding a probability of encountering a bit with the value '1' following a bit with a value of '0' equal to 0.987688. This probability is marginally lower than the corresponding probability for the first transformation. On the other hand, when the variance of the noise term increases to $\sigma^2 = 0.1$ and $\sigma^2 = 0.5$ this probability becomes 0.867348 and 0.698495, respectively. Both these probabilities

Table 4. Patterns in $b_n(0.1, 1.44)$ obtained through the transformation of Eq. (4) and different values of σ^2.

L	patterns	$\sigma^2 = 0.0$	$\sigma^2 = 0.01$	$\sigma^2 = 0.1$	$\sigma^2 = 0.5$
1	0	492207	485787	399771	471981
	1	507793	514213	600229	528019
2	00	0	5979	52973	142274
	01	492415	479647	346368	329606
	10	492414	479647	346368	329606
	11	15169	34725	254289	198512
3	000	0	628	7257	32975
	001	0	5351	45716	109299
	010	477246	447216	186610	187262
	011	15169	32430	159758	142343
	100	0	5351	45716	109299
	101	492414	474296	300652	220307
	110	15168	32430	159758	142343
	111	0	2295	94530	56169
4	0000	0	71	979	6175
	0001	0	557	6278	26800
	0010	0	4840	24665	57459
	0011	0	511	21051	51840
	0100	0	3964	24580	59905
	0101	477246	443252	162030	127357
	0110	15168	30378	98071	98715
	0111	0	2052	61686	43628
	1000	0	557	6278	26800
	1001	0	4794	39438	82499
	1010	477245	442376	161945	129803
	1011	15169	31919	138707	90503
	1100	0	1387	21136	49394
	1101	15168	31043	138622	92949
	1110	0	2052	61686	43628
	1111	0	243	32844	12541

are higher than the corresponding ones for the case of the transformation of Eq. (3). Moreover, note that for the case of the first transformation and $\sigma^2 = 0.5$ a bit with value '0' is more likely to be followed by a bit with the same value (probability equal to 0.55854); a phenomenon that does not occur at present. For the pattern '11' the probability of encountering a zero immediately after it becomes 0.933909, 0.628256, and 0.717049, for σ^2 equal to 0.01, 0.1, and 0.5, respectively. Finally, for the pattern '01' the probability of zero after its appearance is 0.932387, 0.538762, and 0.568140 for σ^2 equal to 0.01, 0.1, and 0.5, respectively. The predictive power of the binary patterns, '0', '11', (perfect predictors in the noise-free binary sequence) and '01' (good predictor in the noise-free binary sequence), with respect to the value of the variance of the additive noise term, σ^2 is illustrated in Fig. 5. To generate Fig. 5, σ^2 assumed values in the interval $[0, 0.5]$ with a stepsize of 10^{-3}.

4 Conclusions

Despite the chaotic nature of the tent map and the resulting complexity of the binary sequences that were derived after the application of two threshold, binary, transformations a large number of short-term predictors was detected. The reported experimental results indicate that the binary sequences generated through the variable threshold binary transformation are more predictable than those obtained through the fixed threshold transformation. This finding is clearer for values of the control parameter, r, close to its upper bound, 2. Indeed for $r = 1.999$ all the patterns of length up to nine appear in the binary sequences obtained through the first transformation,

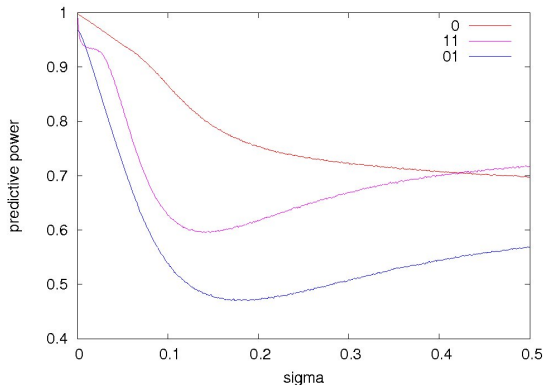


Figure 5. Predictive power of binary patterns identified in the sequences obtained through the transformation of Eq (4) with respect to the variance σ^2 of the noise term.

suggesting that there is no perfect predictor. On the contrary, for the sequences generated through the second transformation with the same value of r, only three out of the four possible patterns of length two are encountered, suggesting that there is a perfect short-term predictor of length one. The inclusion of an additive Gaussian noise term with zero mean in the tent map equation eliminated all perfect predictors. However, for small values of the variance of the Gaussian noise binary patterns with high predictive power were identified.

Future work on the subject will include the investigation of multiplicative noise, as well as, the application of this methodology to real–world time series and in particular financial time series. It is worth noting that the second binary transformation is particularly meaningful in the study of financial time series as it corresponds to the direction of change of the next value relative to the present one.

Acknowledgments

This work was partially supported by the Hellenic Ministry of Education and the European Union under Research Program PYTHAGORAS-89203.

REFERENCES

[1] Free Software Foundation Inc., *GNU Multiple Precision Arithmetic Library ver. 4.1.4.*

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[3] M. Mitchell, *Introduction to Genetic Algorithms*, MIT Press, 1996.

[4] N. G. Packard, 'A genetic learning algorithm for the analysis of complex data', *Complex Systems*, **4**(5), 543–572, (1990).

[5] T. Yoshida, H. Mori, and H. Shigematsu, 'Analytic study of chaos of the tent map: Band structures, power spectra, and critical behaviors', *Journal of Statistical Physics*, **31**(2), 279–308, (1983).

Improving the Accuracy of Neuro-Symbolic Rules with Case-Based Reasoning

Jim Prentzas[1], Ioannis Hatzilygeroudis[2] and Othon Michail[2]

Abstract. In this paper, we present an improved approach integrating rules, neural networks and cases, compared to a previous one. The main approach integrates neurules and cases. Neurules are a kind of integrated rules that combine a symbolic (production rules) and a connectionist (adaline unit) representation. Each neurule is represented as an adaline unit. The main characteristics of neurules are that they improve the performance of symbolic rules and, in contrast to other hybrid neuro-symbolic approaches, retain the modularity of production rules and their naturalness in a large degree. In the improved approach, various types of indices are assigned to cases according to different roles they play in neurule-based reasoning, instead of one. Thus, an enhanced knowledge representation scheme is derived resulting in accuracy improvement. Experimental results demonstrate its effectiveness.

1 INTRODUCTION

In contrast to rule-based systems that solve problems from scratch, case-based systems use pre-stored situations (i.e., cases) to deal with similar new situations. Case-based reasoning offers some advantages compared to symbolic rules and other knowledge representation formalisms. Cases represent specific knowledge of the domain, are natural and usually easy to obtain [11], [12]. Incremental learning comes natural to case-based reasoning. New cases can be inserted into a knowledge base without making changes to the preexisting knowledge. The more cases are available, the better the domain knowledge is represented. Therefore, the accuracy of a case-based system can be enhanced throughout its operation, as new cases become available. A negative aspect of cases compared to symbolic rules is that they do not provide concise representations of the incorporated knowledge. Also it is not possible to represent heuristic knowledge. Furthermore, the time-performance of the retrieval operations is not always the desirable.

Approaches integrating rule-based and case-based reasoning have given interesting and effective knowledge representation schemes and are becoming more and more popular in various fields [3], [13], [14], [15], [17], [18], [19]. The objective of these efforts is to derive hybrid representations that augment the positive aspects of the integrated formalisms and simultaneously minimize their negative aspects. The complementary advantages and disadvantages of rule-based and case-based reasoning are a good justification for their possible combination. The bulk of the approaches combining rule-based and case-based reasoning follow the coupling models [17]. In these models, the problem-solving (or reasoning) process is decomposed into tasks (or stages) for which different representation formalisms (i.e., rules or cases) are applied.

However, a more interesting approach is one integrating more than two reasoning methods towards the same objective. In [16] and [10], such an approach integrating three reasoning schemes, namely rules, neurocomputing and case-based reasoning in an effective way is introduced. To this end, neurules and cases are combined. Neurules are a type of hybrid rules integrating symbolic rules with neurocomputing in a seamless way. Their main characteristic is that they retain the modularity of production rules and also their naturalness in a large degree. In that approach, on the one hand, cases are used as exceptions to neurules, filling their gaps in representing domain knowledge and, on the other hand, neurules perform indexing of the cases facilitating their retrieval. Finally, it results in accuracy improvement.

In this paper, we enhance the above approach by employing different types of indices for the cases according to different roles they play in neurule-based reasoning. In this way, an improved knowledge representation scheme is derived as various types of neurules' gaps in representing domain knowledge are filled in by indexed cases. Experimental results demonstrate the effectiveness of the presented approach compared to our previous one.

The rest of the paper is organized as follows. Section 2 presents neurules, whereas Section 3 presents methods for constructing the indexing scheme of the case library. Section 4 describes the hybrid inference mechanism. Section 5 presents experimental results regarding accuracy of the inference process. Section 6 discusses related work. Finally, Section 7 concludes.

2 NEURULES

Neurules are a type of hybrid rules integrating symbolic rules with neurocomputing giving pre-eminence to the symbolic component. Neurocomputing is used within the symbolic framework to improve the performance of symbolic rules [7], [10]. In contrast to other hybrid approaches (e.g. [4], [5]), the constructed knowledge base retains the modularity of production rules, since it consists of autonomous units (neurules), and also retains their naturalness in a large degree,

[1] Technological Educational Institute of Lamia, Department of Informatics and Computer Technology, 35100 Lamia, Greece, email: dprentzas@teilam.gr.
[2] University of Patras, Dept of Computer Engineering & Informatics, 26500 Patras, Greece, email: {ihatz, michailo}@ceid.upatras.gr.

since neurules look much like symbolic rules [7], [8]. Also, the inference mechanism is a tightly integrated process, which results in more efficient inferences than those of symbolic rules [7], [10]. Explanations in the form of if-then rules can be produced [9], [10].

2.1 Syntax and Semantics

The form of a neurule is depicted in Fig.1a. Each condition C_i is assigned a number sf_i, called its *significance factor*. Moreover, each rule itself is assigned a number sf_0, called its *bias factor*. Internally, each neurule is considered as an adaline unit (Fig.1b). The *inputs* C_i (i=1,...,n) of the unit are the *conditions* of the rule. The weights of the unit are the significance factors of the neurule and its bias is the bias factor of the neurule. Each input takes a value from the following set of discrete values: [1 (true), 0 (false), 0.5 (unknown)]. This gives the opportunity to easily distinguish between the falsity and the absence of a condition in contrast to symbolic rules. The *output D*, which represents the *conclusion* (decision) of the rule, is calculated via the standard formulas:

$$D = f(\mathbf{a}), \quad \mathbf{a} = sf_0 + \sum_{i=1}^{n} sf_i \ \ C_i$$

$$f(\mathbf{a}) = \begin{cases} 1 & \text{if } \mathbf{a} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where **a** is the *activation value* and *f(x)* the *activation function*, a threshold function. Hence, the output can take one of two values ('-1', '1') representing failure and success of the rule respectively.

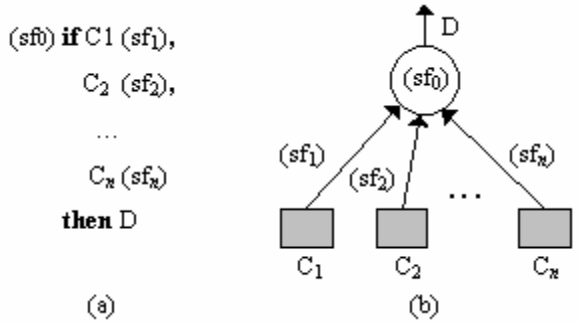


Fig. 1. (a) Form of a neurule (b) a neurule as an adaline unit

The general syntax of a condition C_i and the conclusion D is:

<condition>::= <variable> <l-predicate> <value>
<conclusion>::= <variable> <r-predicate> <value>

where <variable> denotes a *variable*, that is a symbol representing a concept in the domain, e.g. 'sex', 'pain' etc, in a medical domain. <l-predicate> denotes a symbolic or a numeric predicate. The symbolic predicates are {is, isnot} whereas the numeric predicates are {<, >, =}. <r-predicate> can only be a symbolic predicate. <value> denotes a value. It can be a *symbol* or a *number*. The significance factor of a condition represents the significance (weight) of the condition in drawing the conclusion(s). Table 1 (Section 3) presents two example neurules, from a medical diagnosis domain.

Neurules can be constructed either from symbolic rules, thus exploiting existing symbolic rule bases, or from empirical data (i.e., training examples) (see [7] and [8] respectively). An adaline unit is initially assigned to each possible conclusion. Each unit is individually trained via the Least Mean Square (LMS) algorithm. When the training set is inseparable, special techniques are used. In that case, more than one neurule having the same conclusion are produced.

Table 1. Example neurules

NR1: (-23.9) **if** patient-class is human0-20 (10.6), pain is continuous (10.5), fever is high (8.8), fever is medium (8.4), patient-class is human21-35 (6.2), fever is no-fever (2.7), ant-reaction is medium (1.1) **then** disease-type is inflammation	NR2: (-13.4) **if** patient-class is human21-35 (6.9), pain is continuous (3.2), joints-pain is yes (3.1), fever is low (1.5), fever is no-fever (1.5) **then** disease-type is chronic-inflammation

2.2 The Neurule-Based Inference Engine

The neurule-based inference engine performs a task of classification: based on the values of the condition variables and the weighted sums of the conditions, conclusions are reached. It gives pre-eminence to symbolic reasoning, based on a backward chaining strategy [7], [10]. As soon as the initial input data is given and put in the working memory, the output neurules are considered for evaluation. One of them is selected for evaluation. Selection is based on textual order. A neurule fires if the output of the corresponding adaline unit is computed to be '1' after evaluation of its conditions. A neurule is said to be 'blocked' if the output of the corresponding adaline unit is computed to be '-1' after evaluation of its conditions.

A condition evaluates to 'true' ('1'), if it matches a fact in the working memory, that is there is a fact with the same variable, predicate and value. A condition evaluates to 'unknown', if there is a fact with the same variable, predicate and 'unknown' as its value. A condition cannot be evaluated if there is no fact in the working memory with the same variable. In this case, either a question is made to the user to provide data for the variable, in case of an input variable, or an intermediate neurule with a conclusion containing the variable is examined, in case of an intermediate variable. A condition with an input variable evaluates to 'false' ('0'), if there is a fact in the working memory with the same variable, predicate and different value. A condition with an intermediate variable evaluates to 'false' if additionally to the latter there is no unevaluated intermediate neurule that has a conclusion with the same variable. Inference stops either when one or more output neurules are fired (success) or there is no further action (failure).

During inference, a conclusion is rejected (or not drawn) when none of the neurules containing it fires. This happens when: (i) all neurules containing the conclusion have been examined and are blocked or/and (ii) a neurule containing an

alternative conclusion for the specific variable fires instead. For instance, if all neurules containing the conclusion 'disease-type is inflammation' have been examined and are blocked, then this conclusion is rejected (or not drawn). If a neurule containing e.g. the alternative conclusion 'disease-type is primary-malignant' fires, then conclusion 'disease-type is inflammation' is rejected (or not drawn), no matter whether all neurules containing as conclusion 'disease-type is inflammation' have been examined (and are blocked) or not.

3 INDEXING

Indexing concerns the organization of the available cases so that combined neurule-based and case-based reasoning can be performed. Indexed cases fill in gaps in the domain knowledge representation by neurules and during inference may assist in reaching the right conclusion. To be more specific, cases may enhance neurule-based reasoning to avoid reasoning errors by handling the following situations:

(a) Examining whether a neurule misfires. If sufficient conditions of the neurule are satisfied so that it can fire, it should be examined whether the neurule misfires for the specific facts, thus producing an incorrect conclusion.

(b) Examining whether a specific conclusion was erroneously rejected (or not drawn).

In the approach in [10], the neurules contained in the neurule base were used to index cases representing their exceptions. A case constitutes an exception to a neurule if its attribute values satisfy sufficient conditions of the neurule (so that it can fire) but the neurule's conclusion contradicts the corresponding attribute value of the case. In this approach, various types of indices are assigned to cases. More specifically, indices are assigned to cases according to different roles they play in neurule-based reasoning and assist in filling in different types of gaps in the knowledge representation by neurules. Assigning different types of indices to cases can produce an effective approach combining symbolic rule-based with case-based reasoning [1].

In this new approach, a case may be indexed by neurules and by neurule base conclusions as well. In particular, a case may be indexed as:

(a) *False positive (FP)*, by a neurule whose conclusion is contradicting. Such cases, as in our previous approach, represent exceptions to neurules and may assist in avoiding neurule misfirings.

(b) *True positive (TP)*, by a neurule whose conclusion is endorsing. The attribute values of such a case satisfy sufficient conditions of the neurule (so that it can fire) and the neurule's conclusion agrees with the corresponding attribute value of the case. Such cases may assist in endorsing correct neurule firings.

(c) *False negative (FN)*, by a conclusion erroneously rejected (or not drawn) by neurules. Such cases may assist in reaching conclusions that ought to have been drawn by neurules (and were not drawn). If neurules with alternative conclusions containing this variable were fired instead, it may also assist in avoiding neurule misfirings. 'False negative' indices are associated with conclusions and not with specific neurules because there may be more than one neurule with the same conclusion in the neurule base.

The indexing process may take as input the following types of knowledge:

(a) Available neurules and non-indexed cases.

(b) Available symbolic rules and indexed cases. This type of knowledge concerns an available formalism of symbolic rules and indexed exception cases as the one presented in [6].

The availability of data determines which type of knowledge is provided as input to the indexing module. If an available formalism of symbolic rules and indexed cases is presented as input, the symbolic rules are converted to neurules using the 'rules to neurules' module. The produced neurules are associated with the exception cases of the corresponding symbolic rules [10]. Exception cases are indexed as 'false positives' by neurules. Furthermore, for each case 'true positive' and 'false negative' indices may be acquired using the same process as in type (a).

When available neurules and non-indexed cases are given as input to the indexing process, cases must be associated with neurules and neurule base conclusions. For each case, this information can be easily acquired as following:

Until all intermediate and output attribute values of the case have been considered:

1. Perform neurule-based reasoning for the neurules based on the attribute values of the case.
2. If a neurule fires, check whether the value of its conclusion variable matches the corresponding attribute value of the case. If it does (doesn't), associate the case as a 'true positive' ('false positive') with this neurule.
3. Check all intermediate and final conclusions. Associate the case as a 'false negative' with each rejected (or not drawn) conclusion that ought to have been drawn based on the attribute values of the case.

To illustrate how the indexing process works, we present the following example. Suppose that we have a neurule base containing the two neurules in Table 1 and the example cases shown in Table 2 (only the most important attributes of the cases are shown). The cases however, also possess other attributes (not shown in Table 2).

'disease-type' is the output attribute that corresponds to the neurules' conclusion variable. Table 3 shows the types of indices associated with each case in Table 2 at the end of the indexing process.

To acquire indexing information, the input values corresponding to the attribute values of the cases are presented to the example neurules. Recall that when a neurule condition evaluates to 'true' it gets the value '1', whereas when it is false gets '0'.

For example, given the input case C2, the final weighted sum of neurule NR1 is: -23.9 + 10.6 + 10.5 + 8.8 = 6>0. Note that the first three conditions of NR1 evaluate to 'true' whereas the remaining four (i.e., 'fever is medium', 'fever is no-fever', 'patient-class is human21-35' and 'ant-reaction is medium') to 'false' (not contributing to the weighted sum).

Table 2. Example cases

Case ID	*patient-class*	*pain*	*fever*	*ant-reaction*	*joints-pain*	*disease-type*
C1	human21-35	continuous	low	none	yes	chronic-inflammation
C2	human0-20	continuous	high	none	no	inflammation
C3	human0-20	night	high	none	no	inflammation
C4	human0-20	continuous	medium	none	no	inflammation
C5	human21-35	continuous	no-fever	medium	yes	chronic-inflammation
C6	human0-20	continuous	low	none	no	chronic-inflammation

The fact that the final weighted sum is positive means that sufficient conditions of NR1 are satisfied so that it can fire. Furthermore, the corresponding output attribute value of the case matches the conclusion of NR1 and therefore C2 is associated as 'true positive' with NR1.

Table 3. Indices assigned to the example cases in Table 2

Case ID	*Type of index*	*Indexed by*
C1	'True positive'	Neurule NR2
C2	'True positive'	Neurule NR1
C3	'False negative'	Conclusion 'disease-type is inflammation'
C4	'True positive'	Neurule NR1
C5	'False positive'	Neurule NR1
C5	'True positive'	Neurule NR2
C6	'False negative'	Conclusion 'disease-type is chronic-inflammation'

Similarly, when the input values corresponding to the attribute values of cases C1 and C4 are given as input to the neurule base, sufficient conditions of neurules NR2 and NR1 respectively are satisfied so that they can fire and the corresponding output attribute case values match their conclusions. Furthermore, when the input values corresponding to the attribute values of case C5 are given as input to the neurule base, sufficient conditions of both neurules NR1 and NR2 are satisfied so that they can fire. However, the corresponding output attribute case values match the conclusion of NR2 and contradict the conclusion of NR1. In addition, conclusion 'disease-type is inflammation' cannot be drawn when the input values corresponding to the attribute values of case C3 are given as input because the only neurule with the corresponding conclusion (i.e., NR1) is blocked. A similar situation happens for case C6.

4 THE HYBRID INFERENCE MECHANISM

The inference mechanism combines neurule-based with case-based reasoning. The combined inference process mainly focuses on the neurules. The indexed cases are considered when: (a) sufficient conditions of a neurule are fulfilled so that it can fire, (b) all output or intermediate neurules with a specific conclusion variable are blocked and thus no final or intermediate conclusion containing this variable is drawn.

In case (a), firing of the neurule is suspended and case-based reasoning is performed for cases indexed as 'false positives' and 'true positives' by the neurule and cases indexed as 'false negatives' by alternative conclusions containing the neurule's conclusion variable. Cases indexed as 'true positives' by the neurule endorse its firing whereas the other two sets of cases considered (i.e., 'false positives' and 'false negatives') prevent its firing. The results produced by case-based reasoning are evaluated in order to assess whether the neurule will fire or whether an alternative conclusion proposed by the retrieved case will be considered valid instead.

In case (b), the case-based module will focus on cases indexed as 'false negatives' by conclusions containing the specific (intermediate or output) variable.

The basic steps of the inference process are the following:

1. Perform neurule-based reasoning for the neurules.
2. If sufficient conditions of a neurule are fulfilled so that it can fire, then
   - 2.1. Perform case-based reasoning for the 'false positive' and 'true positive' cases indexed by the neurule and the 'false negative' cases associated with alternative conclusions containing the neurule's conclusion variable.
   - 2.2. If none case is retrieved or the best matching case is indexed as 'true positive', the neurule fires and its conclusion is inserted into the working memory.
   - 2.3. If the best matching case is indexed as 'false positive' or 'false negative', insert the conclusion supported by the case into the working memory and mark the neurule as 'blocked'.
3. If all intermediate neurules with a specific conclusion variable are blocked, then
   - 3.1. Examine all cases indexed as 'false negatives' by the corresponding intermediate conclusions, retrieve the best matching one and insert the conclusion supported by the retrieved case into the working memory.
4. If all output neurules with a specific conclusion variable are blocked, then
   - 4.1. Examine all cases indexed as 'false negatives' by the corresponding final conclusions, retrieve the best matching one and insert the conclusion supported by the retrieved case into the working memory.

The similarity measure between two cases c_k and c_l is calculated via a distance metric [1]. The best-matching case to the problem at hand is the one having the maximum similarity

with (minimum distance from) the input case. If multiple stored cases have a similarity equal to the maximum one, a simple heuristic is used.

Let present now two simple inference examples concerning the combined neurule base (Table 1) and the indexed example cases (Tables 2 and 3). Suppose that during inference sufficient conditions of neurule NR1 are satisfied so that it can fire. Firing of NR1 is suspended and the case-based reasoning process focuses on the cases contained in the union of the following sets of indexed cases:

- the set of cases indexed as 'true positives' by NR1: {C2, C4},
- the set of cases indexed as 'false positives' by NR1: {C5} and
- the set of cases indexed as 'false negatives' by alternative conclusions containing variable 'disease-type' (i.e., 'disease-type is chronic inflammation'): {C6}.

So, in this example the case-based reasoning process focuses on the following set of indexed cases: {C2, C4} ∪ {C5} ∪ {C6} = {C2, C4, C5, C6}.

Suppose now that during inference both output neurules in the example neurule base are blocked. The case-based reasoning process will focus on the cases contained in the union set of the following sets of indexed cases:

- the set of cases indexed as 'false negatives' by conclusion 'disease-type is inflammation': {C3}.
- the set of cases indexed as 'false negatives' by conclusion 'disease-type is chronic-inflammation': {C6}.

Therefore, in this example the case-based reasoning process focuses on the following set of indexed cases: {C3} ∪ {C6} = {C3, C6}.

5 EXPERIMENTAL RESULTS

In this section, we present experimental results using datasets acquired from [2]. Note that there are no intermediate conclusions in these datasets. The experimental results involve evaluation of the presented approach combining neurule-based and case-based reasoning and comparison with our previous approach [10]. 75% and 25% of each dataset were used as training and testing sets respectively. Each initial training set was used to create a combined neurule base and indexed case library. For this purpose, each initial training set was randomly split into two disjoint subsets, one used to create neurules and one used to create an indexed case library. More specifically, 2/3 of each initial training set was used to create neurules by employing the 'patterns to neurules' module [8] whereas the remaining 1/3 of each initial training set constituted non-indexed cases. Both types of knowledge (i.e., neurules and non-indexed cases) were given as input to the indexing construction module presented in this paper producing a combined neurule base and an indexed case library which will be referred to as NBRCBR. Neurules and non-indexed cases were also used to produce a combined neurule base and an indexed case library according to [10] which will be referred to as NBRCBR_PREV.

Inferences were run for both NBRCBR and NBRCBR_PREV using the testing sets. Inferences from NBRCBR_Prev were performed using the inference mechanism combining neurule-based and CBR as described in [10]. Inferences from NBRCBR were performed according to the inference mechanism described in this paper. No test case was stored in the case libraries.

Table 4 presents such experimental results regarding inferences from NBRCBR and NBRCBR_PREV. It presents results regarding classification accuracy of the integrated approaches and the percentage of test cases resulting in neurule-based reasoning errors that were successfully handled by case-based reasoning. Column '% FPs handled' refers to the percentage of test cases resulting in neurule misfirings (i.e., 'false positives') that were successfully handled by case-based reasoning. Column '% FNs handled' refers to the percentage of test cases resulting in having all output neurules blocked (i.e., 'false negatives') that were successfully handled by case-based reasoning. 'False negative' test cases are handled in NBRCBR_PREV by retrieving the best-matching case from the whole library of indexed cases.

Table 4. Experimental results

Dataset	NBRCBR			NBRCBR_PREV		
	Classification Accuracy	**% FPs Handled**	**% FNs Handled**	**Classification Accuracy**	**% FPs Handled**	**% FNs Handled**
Car (1728 patterns)	96.04%	52.81%	64.07%	92.49%	15.51%	20.36%
Nursery (12960 patterns)	98.92%	58.68%	52.94%	97.68%	6.60%	18.82%

As can be seen from the table, the presented approach results in improved classification accuracy. Furthermore, in inferences from NBRCBR the percentages of both 'false positive' and 'false negative' test cases successfully handled are greater than the corresponding percentages in inferences from NBRCBR_PREV. Results also show that there is still room for improvement.

We also tested a nearest neighbor approach working alone in these two datasets (75% of the dataset used as case library and 25% of the dataset used as testing set). We used the similarity measure presented in Section 5. The approach classified the input case to the conclusion supported by the best-matching case retrieved from the case library. Classification accuracy for car and nursery dataset is 90.45% and 96.67% respectively. So, both integrated approaches perform better. This is due to the fact that the indexing schemes assist in focusing on specific parts of the case library.

7 CONCLUSIONS

In this paper, we present an approach integrating neurule-based and case-based reasoning that improves a previous hybrid approach [10]. Neurules are a type of hybrid rules integrating symbolic rules with neurocomputing. In contrast to other neuro-symbolic approaches, neurules retain the naturalness and modularity of symbolic rules. Integration of neurules and cases is done in order to improve the accuracy of the inference mechanism. Cases are indexed according to the roles they can play during neurule-based inference. More specifically, they are associated as 'true positives' and 'false positives' with neurules and as 'false negatives' with neurule base conclusions.

The presented approach integrates three types of knowledge representation schemes: symbolic rules, neural networks and case-based reasoning. Most hybrid intelligent systems implemented in the past usually integrate two intelligent technologies e.g. neural networks and expert systems, neural and fuzzy logic, genetic algorithms and neural networks, etc. A new development that should receive interest in the future is the integration of more than two intelligent technologies, facilitating the solution of complex problems and exploiting multiple types of data sources.

References

[1] G. Agre, 'KBS Maintenance as Learning Two-Tiered Domain Representation', In M.M. Veloso, A. Aamodt, (Eds.): Case-Based Reasoning Research and Development, First International Conference, ICCBR-95, Proceedings, Lecture Notes in Computer Science, Vol. 1010, Springer-Verlag, 108-120, 1995.

[2] A. Asuncion, D.J. Newman, 'UCI Repository of Machine Learning Databases' [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA, University of California, School of Information and Computer Science (2007).

[3] N. Cercone, A. An, C. Chan, 'Rule-Induction and Case-Based Reasoning: Hybrid Architectures Appear Advantageous', *IEEE Transactions on Knowledge and Data Engineering*, **11**, 164-174, (1999).

[4] S. I. Gallant, *Neural Network Learning and Expert Systems*, MIT Press, 1993.

[5] A.Z. Ghalwash, 'A Recency Inference Engine for Connectionist Knowledge Bases', *Applied Intelligence*, **9**, 201-215, (1998).

[6] A.R. Golding, P.S. Rosenbloom, 'Improving accuracy by combining rule-based and case-based reasoning', *Artificial Intelligence*, **87**, 215-254, (1996).

[7] I. Hatzilygeroudis, J. Prentzas, 'Neurules: Improving the Performance of Symbolic Rules', *International Journal on AI Tools*, **9**, 113-130, (2000).

[8] I. Hatzilygeroudis, J. Prentzas, 'Constructing Modular Hybrid Rule Bases for Expert Systems', *International Journal on AI Tools*, **10**, 87-105, (2001).

[9] I. Hatzilygeroudis, J. Prentzas, 'An Efficient Hybrid Rule-Based Inference Engine with Explanation Capability', Proceedings of the 14th International FLAIRS Conference, AAAI Press, 227-231, (2001).

[10] I. Hatzilygeroudis, J. Prentzas, 'Integrating (Rules, Neural Networks) and Cases for Knowledge Representation and Reasoning in Expert Systems', *Expert Systems with Applications*, **27**, 63-75, (2004).

[11] J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[12] D.B. Leake (ed.), *Case-Based Reasoning: Experiences, Lessons & Future Directions*, AAAI Press/MIT Press, 1996.

[13] M.R. Lee, 'An Exception Handling of Rule-Based Reasoning Using Case-Based Reasoning', *Journal of Intelligent and Robotic Systems*, **35**, 327-338, (2002).

[14] C.R. Marling, M. Sqalli, E. Rissland, H. Munoz-Avila, D. Aha, 'Case-Based Reasoning Integrations', *AI Magazine*, **23**, 69-86, (2002).

[15] S. Montani, R. Bellazzi, 'Supporting Decisions in Medical Applications: the Knowledge Management Perspective', *International Journal of Medical Informatics*, **68**, 79-90, (2002).

[16] J. Prentzas, I. Hatzilygeroudis, 'Integrating Hybrid Rule-Based with Case-Based Reasoning', In S. Craw and A. Preece (Eds), Advances in Case-Based Reasoning, Proceedings of the European Conference on Case-Based Reasoning, ECCBR-2002, Lecture Notes in Artificial Intelligence, Vol. 2416, Springer-Verlag, 336-349, 2002.

[17] J. Prentzas, I. Hatzilygeroudis, 'Categorizing Approaches Combining Rule-Based and Case-Based Reasoning', *Expert Systems*, **24**, 97-122, (2007).

[18] E.L. Rissland, D.B. Skalak, 'CABARET: Rule Interpretation in a Hybrid Architecture', *International Journal of Man-Machine Studies*, **34**, 839-887, (1991).

[19] D. Rossille, J.-F. Laurent, A. Burgun, 'Modeling a Decision Support System for Oncology using Rule-Based and Case-Based Reasoning Methodologies', *International Journal of Medical Informatics*, **74**, 299-306, (2005).

[20] H. Vafaie, C. Cecere, 'CORMS AI: Decision Support System for Monitoring US Maritime Environment', Proceedings of the 17th Innovative Applications of Artificial Intelligence Conference (IAAI), AAAI Press, 1499-1507, (2005).

Combinations of Case-Based Reasoning with Other Intelligent Methods

Jim Prentzas[1] and Ioannis Hatzilygeroudis[2]

Abstract. Case-based reasoning is a popular approach used in intelligent systems. Whenever a new case has to be dealt with, the most similar cases are retrieved from the case base and their encompassed knowledge is exploited in the current situation. Combinations of case-based reasoning with other intelligent methods have been explored deriving effective knowledge representation schemes. Although some types of combinations have been mostly explored, other types have not been thoroughly investigated. In this paper, we briefly outline popular case-based reasoning combinations. More specifically, we focus on combinations of case-based reasoning with rule-based reasoning, soft computing and ontologies. We illustrate basic types of such combinations and discuss future directions.

1 INTRODUCTION

Case-based representations store a large set of previous cases with their solutions in the case base using them whenever a similar new case has to be dealt with [19], [22]. Whenever, a new input case comes in, a case-based system performs inference in four phases known as the case-based reasoning (CBR) cycle [1]: (i) retrieve, (ii) reuse, (iii) revise and (iv) retain. The retrieval phase retrieves from the case base the most relevant stored case(s) to the new case. Indexing schemes and similarity metrics are used for this purpose. In the reuse phase, a solution for the new case is created based on the retrieved most relevant case(s). The revise phase validates the correctness of the proposed solution, perhaps with the intervention of the user. Finally, the retain phase decides whether the knowledge learned from the solution of the new case is important enough to be incorporated into the system.

CBR can be effectively combined with other intelligent methods [25], [31]. Two main trends for CBR combinations can be discerned. The first trend involves embedded approaches in which the primary intelligent method (usually CBR) embeds one or more other intelligent methods to assist its internal online and offline tasks. The second combination trend involves approaches in which the problem solving process can be decomposed into tasks for which different representation formalisms are required or available. In such situations, a CBR system as a whole (with its possible internal modules) is integrated 'externally' with other intelligent systems to create an improved overall system.

Popular CBR combinations involve combinations with rule-based reasoning (RBR), model-based reasoning (MBR) and soft computing methods. CBR has also been combined with other intelligent methods (e.g. ontologies). In certain CBR combinations both combination trends have been followed. In other combinations one of the two trends is mostly explored.

In this paper, we briefly discuss aspects involving CBR combinations. We focus on intelligent methods with which CBR is usually combined. Our purpose is not to present an extensive survey of developed CBR combinations but to present their key aspects.

3 COMBINATIONS OF CBR

Combinations of CBR with other intelligent methods have been explored for more effective knowledge representation and problem solving. CBR can be combined with various intelligent methods. However, CBR is usually combined with RBR, MBR and soft computing methods.

To categorize CBR combinations one could use Medsker's general categorization scheme for integrated intelligent systems [26]. Medsker distinguishes five main combination models: standalone, transformational, loose coupling, tight coupling and fully integrated models. Distinction between those models is based on the degree of coupling between the integrated components. Underlying categories for some of these models are also defined. Main types of underlying categories for loose and tight coupling models involve pre-processing, post-processing and co-processing models as well as embedded processing (for tight coupling models only). Not all of these combination models and/or their underlying categories have been thoroughly explored in the case of CBR combinations. The types of combination models that have been applied to CBR combinations depend on the nature of the other intelligent methods combined with CBR. Some combination models are difficult to apply in certain CBR combinations. For instance, it is difficult to apply the fully integrated model in combinations of RBR with CBR. Obviously, the standalone model can be applied to combinations of CBR with any other method.

Generally speaking, coupling models are the most usual CBR combination models. More specifically, embedded coupling approaches constitute perhaps the most popular trend.

[1] Technological Educational Institute of Lamia, Department of Informatics and Computer Technology, 35100 Lamia, Greece, email: dprentzas@teilam.gr.
[2] University of Patras, Dept of Computer Engineering & Informatics, 26500 Patras, Greece, email: ihatz@ceid.upatras.gr.

Most of the combinations following this trend use other intelligent methods to assist various CBR tasks. CBR is a generic methodology for building knowledge-based systems and its internal reasoning tasks can be implemented using a number of techniques as long as the guiding CBR principles are followed [36]. The reverse approach that is, embedding case-based modules into intelligent systems employing other representations to assist in their internal tasks does not seem to be popular with the exception of combinations with genetic algorithms. In combinations of CBR with RBR and MBR, various coupling approaches have also been investigated besides embedded approaches [31]. In coupling combinations of CBR with soft computing methods, embedded approaches seem to be the most thoroughly investigated.

In the following, we discuss main issues involving combinations of CBR with RBR, fuzzy logic, neural networks, genetic algorithms and ontologies.

3.1 Combinations of CBR with RBR

Various types of coupling models involving combinations of CBR and RBR have been investigated i.e., sequential processing, co-processing and embedded processing [31].

In sequential processing, information (produced by reasoning) necessarily passes sequentially through some or all of the combined modules to produce the final result [33], [11].

In co-processing approaches, the combined modules closely interact in producing the final result. Such systems can be discerned into two types: cooperation-oriented, which give emphasis on cooperation, and reconciliation-oriented, which give emphasis on reconciliation. In the former type, the combined components cooperate with each other (usually by interleaving their reasoning steps) [27], [32]. In the latter, each component produces its own conclusion, possibly differing from the conclusion of the other component, and thus a reconciliation process is necessary [14].

In embedded processing, CBR systems employ one or more RBR modules to perform tasks of their CBR cycle (e.g. retrieval and adaptation). Such approaches are quite common in CBR especially for adaptation. RBR systems embedding CBR modules do not seem to exist.

3.2 Combinations of CBR with Fuzzy Logic

CBR can be combined with fuzzy logic in fruitful ways in order to handle imprecision. A usual approach is the incorporation of fuzzy logic into a CBR system in order to improve CBR aspects [4], [29], [35], [9]. Such combinations have been vastly explored as imprecision and uncertainty are inherent in various CBR tasks. Fuzzy terms may be used in case representation enabling a flexible encoding of case features that encompasses imprecise and uncertain information. Fuzzy logic may be also proved very useful in indexing and retrieval. Fuzzy indexing enables multiple indexing of a case on a single feature with different degrees of membership [35]. Fuzzy similarity assessment and matching methods can produce more accurate results. Fuzzy clustering and classification methods can also be applied in case retrieval. In addition, fuzzy adaptation rules can be employed in case adaptation.

The works concerning combination of RBR with CBR [31] could potentially be improved with use of fuzzy rules. Investigation of coupling approaches in combinations of CBR with fuzzy systems besides embedded ones could be fruitful.

3.3 Combinations of CBR with Neural Networks

Neural networks are usually employed by CBR to perform tasks such as indexing, retrieval and adaptation. In this way, appealing characteristics of neural networks such as parallelism, robustness, adaptability, generalization and ability to cope with incomplete input data are exploited [10], [35]. Due to the fact that different types of neural networks have been developed (e.g. back propagation neural networks, radial basis function networks, Self-Organizing Map networks, ART network), different types of neural capabilities for classification and clustering can be exploited. Certain CBR approaches have employed different types of neural networks for the various internal CBR tasks (e.g. [12], [34]). Knowledge extracted from neural networks could also be exploited by CBR [10], [35]. An interesting direction could involve non-embedded coupling approaches combining CBR with neural networks.

3.4 Combinations of CBR with Genetic Algorithms

Usual combinations of CBR with genetic algorithms (GAs) involve use of GAs to optimize (one or more) aspects of a CBR system. On the other hand, CBR can be exploited to enhance GAs. Other types of combinations of CBR with GAs can be also implemented.

GAs can be used within CBR to enhance indexing and retrieval. GAs have been used to assign case feature weights enhancing similarity assessment [39], [8], to perform feature selection [18] and generally to select relevant indices for evolving environments. GAs have also been used to retrieve multiple similar cases [38]. If k nearest neighbor retrieval is applied, genetic algorithms can be used to find the optimal k parameter in order to improve the retrieval accuracy [2]. Furthermore, GAs can be used to perform instance selection i.e., finding the representative cases in a case base and determining a reduced subset of a case base. In this way, time performance is improved by reducing search space and accuracy can be improved through elimination of noisy and useless cases [2].

Additionally, GAs have been used to enhance case adaptation [16], [17]. Genetic algorithms can also optimize case representation, e.g. by performing case feature discretization [18] and removing irrelevant features. Such optimizations improve accuracy, search time and storage requirements. It is also quite usual to simultaneously optimize more than one CBR aspect with GAs (e.g. [2], [18]).

On the other hand CBR can be employed to enhance GAs. CBR can be applied to GAs by creating cases to track the history of a search. This case base can contribute in the understanding of how a solution was reached, why a solution works, and what the search space looks like. It could thus be

used to design highly tailored search strategies for future use [23]. Such an approach could therefore be used to explain the results of the genetic algorithm and for knowledge extraction. Moreover, similar stored cases can be also incorporated into a genetic algorithm to reduce convergence time and improve solution accuracy. GAs randomly initialize their starting population. Instead, relevant stored cases can be used as part of the initial population (solution) of GAs. Additionally, relevant stored cases can be periodically injected into the pool of chromosomes while the genetic algorithm runs [24], [7]. In certain approaches, CBR is exploited by GAs for both knowledge extraction and case injection [30].

3.5 Combinations of CBR with Ontologies

Ontologies facilitate knowledge sharing and reuse. They can provide an explicit conceptualization describing data semantics and a shared and common understanding of the domain knowledge that can be communicated among agents and application systems [6]. Ontologies play a crucial role in enabling the processing and sharing of knowledge between programs on the Web [21]. Intelligent Decision Support Systems in the semantic Web framework should be able to handle, integrate with and reason from distributed data and information on the Web [3].

Therefore ontologies can be combined with CBR in various ways. Ontologies can be used by a CBR system to represent the input problem [20], to enhance similarity assessment [13], case representation, case abstraction and case adaptation [3]. Ontologies may perform all such CBR tasks [37].

3.6 Combinations of CBR with Multiple Intelligent Methods

The previous sections focused on combinations of CBR with one other individual intelligent method. However, intelligent systems have been developed that combine CBR with multiple other intelligent methods. Such multi-integrated paradigms usually follow a coupling model.

Obviously, a CBR system may employ multiple intelligent methods (e.g. rules and various soft computing methods) to perform its internal tasks [36]. Typical examples of approaches employing multiple soft computing methods within the CBR cycle are presented in [12] and [34]. In [12] all of the four phases of the CBR cycle employ soft computing methods. Employed soft computing methods are a self-organizing neural network for retrieval, a radial basis neural network for reuse, fuzzy systems for revise and all soft computing methods for retain. In [34] fuzzy logic, supervised and unsupervised neural networks and a genetic algorithm are employed for case representation, indexing, retrieval and adaptation.

More interesting approaches concern multi-integrated systems not following the embedded approach. Typical such multi-integrated approaches involve combinations of CBR, RBR and MBR (e.g. [28]). Such approaches seem to be quite effective, because combinations of CBR with RBR and MBR individually have been thoroughly investigated. Quite often such systems have been implemented to deal with deficiencies of earlier systems combining CBR with only one of the other two intelligent methods (e.g. RBR or MBR alone). Multi-integrated CBR approaches, besides those involving RBR/MBR, could be developed. For instance, ontologies could constitute an interesting candidate method that could be combined with CBR and another intelligent method in order to facilitate knowledge sharing and reuse among the integrated system components themselves [5] and among integrated systems. Such a combination could be useful in Web-based systems that need to share knowledge. Fruitful such approaches could involve combinations of CBR, ontologies and RBR/MBR. For instance in [6] an approach combining CBR, RBR and an ontology is presented.

Multi-integrated paradigms could also be considered systems combining CBR with certain types of neuro-symbolic or neuro-fuzzy approaches in which the neuro-symbolic (neuro-fuzzy) module fully integrates the neural and symbolic (fuzzy) approach. Such modules could be used within CBR instead of plain neural or fuzzy components. Non-embedded coupling approaches can be applied as well. For instance, in [15] a neuro-symbolic method is combined with CBR according to the reconciliation coupling approach.

4 CONCLUSIONS

In this paper, we discuss key aspects involving combinations of CBR with other intelligent methods. Such combinations are becoming increasingly popular due to the fact that in many application domains a vast amount of case data is available. Such combined approaches have managed to solve problems in application domains where a case-based module needs the assistance and/or completion of other intelligent modules in order to produce effective results. This trend is very likely to carry on in the following years.

Future directions in combinations of CBR with other intelligent methods could involve a number of aspects. Main such aspects involve: (a) combinations of CBR with soft computing methods, (b) combinations of CBR with fuzzy rules, (c) combinations of CBR with ontologies and (d) combinations of CBR with neuro-symbolic and neuro-fuzzy approaches.

Combinations of CBR with soft computing methods not following an embedded coupling approach could be an interesting future research direction. At present there seems to be a lack of great interest in pursuing this direction since the main interest has been focused on employing soft computing methods within CBR. A non-embedded direction in the combinations of CBR with soft computing could be pursued as thoroughly as in the case of combinations of CBR with RBR/MBR. A further step towards this direction could involve non-embedded approaches combining CBR with multiple soft computing methods or combinations of CBR, soft computing and other intelligent methods (e.g. RBR, MBR or ontologies).

Combinations of CBR with fuzzy rule-based systems could be based on work combining CBR with RBR that is, investigation of various coupling approaches.

The increasing interest in Web-based intelligent systems and future advances in the Semantic Web is likely to provide an impetus to approaches combining CBR with ontologies. This

trend is likely to involve multi-integrated approaches combining CBR, ontologies and other intelligent methods.

Finally, a direction that may be useful to pursue involves non-embedded coupling approaches combining CBR with neuro-symbolic and neuro-fuzzy modules. Few such approaches have been developed.

REFERENCES

[1] A. Aamodt, E. Plaza, 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches', *AI Communications*, **7**, 39-59, (2004).

[2] H. Ahn, K. Kim, 'Global optimization of case-based reasoning for breast cytology diagnosis', *Expert Systems with Applications* (to appear).

[3] I. Bischindaritz, 'Memoire: A Framework for Semantic Interoperability of Case-Based Reasoning Systems in Biology and Medicine', *Artificial Intelligence in Medicine*, **36**, 177-192, (2006).

[4] P.P. Bonissone, R. Lopez de Mantaras, 'Fuzzy Case-Based Reasoning Systems', *Handbook on Fuzzy Computing*, vol. F 4.3, Oxford University Press, 1998.

[5] L. Castillo, E. Armengol, E. Onaindia, L. Sebastia, J. Gonzalez-Boticario, A. Rodriguez, S. Fernandez, J.D. Arias, D. Borrajo, 'SAMAP: An User Oriented Adaptive System for Planning Tourist Visits', *Expert Systems with Applications*, **34**, 1318-1332, (2008).

[6] L. Ceccaroni, U. Cortes, M. Sanchez-Marre, 'OntoWEDSS : Augmenting Environmental Decision-Support Systems with Ontologies', *Environmental Modelling & Software*, **19**, 785-797, (2004).

[7] P.-C. Chang, J.-C. Hsieh, C.-H. Liu, 'A Case-Injected Genetic Algorithm for Single Machine Scheduling Problems with Release Time', *International Journal on Production Economics*, **103**, 551-564, (2006).

[8] P.-C. Chang, C.-Y. Lai, K. R. Lai, 'A Hybrid System by Evolving Case-Based Reasoning with Genetic Algorithm in Wholesaler's Returning Book Forecasting', *Decision Support Systems*, **42**, 1715-1729, (2006).

[9] W. Cheetam, S.C.K. Shiu, R.O. Weber, 'Soft Case-Based Reasoning', *Knowledge Engineering Review*, **20**, 267-269, (2006).

[10] D. Chen, P. Burrell, 'Case-Based Reasoning System and Artificial Neural Networks: A Review', *Neural Computing & Applications*, **10**, 264-276, (2001).

[11] R.-J. Dzeng, H.-Y. Lee, 'Critiquing Contractors' Scheduling by Integrating Rule-Based and Case-Based Reasoning', *Automation in Construction*, **13**, 665-678, (2004).

[12] F. Fdez-Riverola, J.M. Corchado, 'FSfRT: Forecasting System for Red Tides', *Applied* Intelligence, **21**, 251-264, (2004).

[13] P. Gervas, B. Diaz-Agudo, F. Peinado, R. Hervas, 'Story Plot Generation Based on CBR', *Knowledge-Based Systems*, **18**, 235-242, (2005).

[14] A.R. Golding, P.S. Rosenbloom, 'Improving Accuracy by Combining Rule-Based and Case-Based Reasoning', *Artificial Intelligence*, **87**, 215-254, (1996).

[15] I. Hatzilygeroudis, J. Prentzas, 'Integrating (Rules, Neural Networks) and Cases for Knowledge Representation and Reasoning in Expert Systems', *Expert Systems with Applications*, **27**, 63-75, (2004).

[16] B. W. Huang, M. L. Shih, N.-H. Chiu, W. Y. Hu, C. Chiu, 'Price information evaluation and prediction for broiler using adapted case-based reasoning approach', *Expert Systems with Applications* (to appear).

[17] Y.-K. Juan, S.-G. Shih, Y.-H. Perng, 'Decision Support for Housing Customization: A hybrid approach using case-based reasoning and genetic algorithm', *Expert Systems with Applications*, **31**, 83-93, (2006).

[18] K.-J. Kim, 'Toward Global Optimization of Case-Based Reasoning Systems for Financial Forecasting', *Applied Intelligence*, **21**, 239-249, (2004).

[19] J.L. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann, 1993.

[20] O. Byung Kwon, 'Meta Web Service: Building Web-based Open Decision Support System based on Web Services', *Expert Systems with Applications*, **24**, 375-389, (2003).

[21] O. Kwon, M. Kim, 'MyMessage: Case-Based Reasoning and Multicriteria Decision Making Techniques for Intelligent Context-Aware Message Filtering', *Expert Systems with Applications*, **27**, 467-480, (2004).

[22] D.B. Leake (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press, 1996.

[23] S.J. Louis, G. McGraw, R.O. Wyckoff, 'Case-based reasoning assisted explanation of genetic algorithm results', *Journal of Experimental & Theoretical Artificial Intelligence*, **5**, 21-37, (1993).

[24] S. J. Louis, C. Miles, 'Playing to Learn: Case-Injected Genetic Algorithms for Learning to Play Computer Games', *IEEE Transactions on Evolutionary Computation*, **9**, 669-681, (2005).

[25] C. Marling, M. Sqalli, E. Rissland, H. Munoz-Avila, D. Aha, 'Case-Based Reasoning Integrations', *AI Magazine*, **23**, 69-86, (2002).

[26] L.R. Medsker, *Hybrid Intelligent Systems*, Kluwer Academic Publishers, Second Printing, 1998.

[27] S. Montani, R. Bellazzi, 'Supporting Decisions in Medical Applications: the Knowledge Management Perspective', *International Journal of Medical Informatics*, **68**, 79-90, (2002).

[28] S. Montani, P. Magni, R. Bellazzi, C. Larizza, A.V. Roudsari, E.R. Carson, 'Integrating model-based decision support in a multi-modal reasoning system for managing type 1 diabetic patients', *Artificial Intelligence in Medicine*, **29**, 131-151, (2003).

[29] S. K. Pal, S.C.K. Shiu, *Foundations of Soft Case-Based Reasoning*, John Wiley, 2004.

[30] E.I. Perez, C.A. Coello Coello, A. Hernandez-Aguirre, 'Extraction and Reuse of Design Patterns from Genetic Algorithms using Case-Based Reasoning', *Soft Computing*, **9**, 44-53, (2005).

[31] J. Prentzas, I. Hatzilygeroudis, 'Categorizing Approaches Combining Rule-Based and Case-Based Reasoning', *Expert Systems*, **24**, 97-122, (2007).

[32] E.L. Rissland, D.B. Skalak, 'CABARET: Rule Interpretation in a Hybrid Architecture', *International Journal of Man-Machine Studies*, **34**, 839-887, (1991).

[33] D. Rossille, J.-F. Laurent, A. Burgun, 'Modeling a Decision Support System for Oncology using Rule-Based and Case-Based Reasoning Methodologies', *International Journal of Medical Informatics*, **74**, 299-306, (2005).

[34] K.M. Saridakis, A.J. Dentsoras, 'Case-DeSC: A System for Case-Based Design with Soft Computing Techniques', *Expert Systems with Applications*, **32**, 641-657, (2007).

[35] S.C.K. Shiu, S.K. Pal, 'Case-Based Reasoning: Concepts, Features and Soft Computing', *Applied Intelligence*, **21**, 233-238, (2004).

[36] I. Watson, 'Case-Based Reasoning is a Methodology not a Technology', *Knowledge-Based Systems*, **12**, 303-308, (1997)

[37] P. Wriggers, M. Siplivaya, I. Joukova, R. Slivin, 'Intelligent Support of Engineering Analysis using Ontology and Case-Based Reasoning', *Engineering Applications of Artificial Intelligence*, **20**, 709-720, (2007).

[38] H.-L. Yang, C.-S. Wang, 'Two stages of case-based reasoning – Integrating genetic algorithm with data mining mechanism', *Expert Systems with Applications* (to appear).

[39] F.-C. Yuan, C. Chiu, 'A Hierarchical Design of Case-Based Reasoning in the Balanced Scorecard Application', *Expert Systems with Applications* (to appear).

Combining Argumentation and Hybrid Evolutionary Systems in a Portfolio Construction Application

Nikolaos Spanoudakis[1] and **Konstantina Pendaraki**[2] and **Grigorios Beligiannis**[2]

Abstract. In this paper we present an application for the construction of mutual fund portfolios. It is based on a combination of Intelligent Methods, namely an argumentation based decision making framework and a forecasting algorithm combining Genetic Algorithms (GA), MultiModel Partitioning (MMP) theory and Extended Kalman Filters (EKF). The argumentation framework is employed in order to develop mutual funds performance models and to select a small set of mutual funds, which will compose the final portfolio. The forecasting algorithm is employed in order to forecast the market status (inflating or deflating) for the next investment period. The knowledge engineering approach and application development steps are also discussed.

1 INTRODUCTION

Portfolio management [8] is concerned with constructing a portfolio of securities (e.g., stock, bonds, mutual funds [13], etc.) that maximizes the investor's utility. In a previous study [14], we constructed mutual fund (MF) portfolios using an argumentation based decision making framework. We developed rules that characterize the market and different investor types policies using evaluation criteria of fund performance and risk. We also defined strategies for resolving conflicts over these rules. Furthermore, the developed application can be used for a set of different investment policy scenarios and supports the investor/portfolio manager in composing efficient MF portfolios that meet his investment preferences. The traditional portfolio theories ([8], [11], [12]) were based on unidimensional approaches that did not fit to the multidimensional nature of risk ([3]), and they did not capture the complexity presented in the data set. In [14], this troublesome situation was resolved by the high level of adaptability in the decisions of the portfolio manager or investor when his environment is changing and the characteristics of the funds are multidimensional that was demonstrated by the use of argumentation.

Our study showed that when taking into account the market context, the results were better if we could forecast the status of the market of the following investment period. In order to achieve this goal we employed a hybrid system that combines Genetic Algorithms (GA), MultiModel Partitioning (MMP) theory and the Extended Kalman Filter (EKF). A general description of this algorithm and its application in linear and non-linear data is discussed in [2], while the specific version used in this contribution is presented in [1], where its successful application to non-linear data is also presented. This algorithm captured our attention because it had been successfully used in the past for accurately predicting the evolution of stock values in the Greek market (its application on economic data is presented in [2]). Moreover, there is a lot of work on hybrid evolutionary algorithms and their application on many difficult problems has shown very promising results [4]. The problem of predicting the behavior of the financial market is an open problem and many solutions have been proposed. However, there isn't any known algorithm able to identify effectively all kinds of behaviors. Also, many traditional methods have been applied to the same problem and the results obtained were not very satisfactory. There are two main difficulties in this problem, firstly the search space is huge and, secondly, it comprises of many local optima.

In this contribution, we present the whole application resulting from the combination of argumentation with hybrid evolutionary systems along with the respective results.

The rest of the paper is organized as follows: Section two presents an overview of the concepts and application domain knowledge. Section three outlines the main features of the proposed argumentation based decision-making framework and the developed argumentation theory. The forecasting hybrid evolutionary system is presented in section four, followed by section five, which presents the developed application and discusses the obtained empirical results. Finally, section six summarizes the main findings of this research.

2 DOMAIN KNOWLEDGE

This section describes the criteria (or variables) used for creating portfolios and the knowledge on how to use these criteria in order to construct a portfolio.

The data used in this study is provided from the Association of Greek Institutional Investors and consists of daily data of domestic equity mutual funds (MFs) over the period January 2000 to December 2005.

The proposed framework is based on five fundamental variables. The *return of the funds* is the actual value of return of an investment defined by the difference between the nominal return and the rate of inflation. This variable is based on the net price of a fund. At this point, it is very important to mention that transaction costs such as management commission are included in the net price. Frond-end commission and redemption commission fluctuate depending on the MF class and in most cases are very low. The *standard deviation* is used to measure the variability of the fund's daily returns, thus representing the total risk of the fund. The *beta coefficient (β)* is a measure of fund's risk in relation to the capital risk. The *Sharpe index* [13] is a useful measure of performance and is used to measure the expected return of a fund per unit of risk, defined by the standard deviation. The *Treynor index* [15] is similar to the Sharpe index except that

[1] Technical University of Crete, Greece, email: nikos@science.tuc.gr
[2] University of Ioannina, Greece, email: {dpendara, gbeligia}@cc.uoi.gr

performance is measured as the risk premium per unit of systematic (beta coefficient) and not of total risk.

On the basis of the argumentation framework for the selection of a small set of MF, which will compose the final multi-portfolios, the examined funds are clustered in three groups for each criterion for each year. For example, we have funds with high, medium and low performance (return), the same for the other criteria.

The aforementioned performance and risk variables visualize the characteristics of the capital market (bull or bear) and the type of the investor according to his investment policy (aggressive or moderate). Further information is represented through variables that describe the general conditions of the market and the investor policy (selection of portfolios with high performance per unit of risk).

The general conditions of the market are characterized through the development of funds which have high performance levels (high return). Regarding the market context, in a bull market, funds are selected if they have high systematic or total risk. On the other hand, in a bear market, we select funds with low systematic and total risk. An aggressive investor is placing his capital upon funds with high performance and high systematic risk. Accordingly, a moderate investor selects funds with high performance and low or medium systematic risk. Some types of investors select portfolios with high performance per unit of risk. Such portfolios are characterized by high Sharpe ratio and high Treynor ratio.

3 ARGUMENTATION-BASED DECISION MAKING

In this section we firstly present the argumentation framework that we used and then we describe the domain knowledge modeling based on the argumentation framework.

3.1 The Argumentation Framework

Autonomous agents, be they artificial or human, need to make decisions under complex preference policies that take into account different factors. In general, these policies have a dynamic nature and are influenced by the particular state of the environment in which the agent finds himself. The agent's decision process needs to be able to synthesize together different aspects of his preference policy and to adapt to new input from the current environment. Such agents are the mutual fund managers.

In order to address requirements like the above, Kakas and Moraitis ([6]) proposed an argumentation based framework to support an agent's self deliberation process for drawing conclusions under a given policy.

Argumentation can be abstractly defined as the principled interaction of different, potentially conflicting arguments, for the sake of arriving at a consistent conclusion (see e.g. [10]). The nature of the "conclusion" can be anything, ranging from a proposition to believe, to a goal to try to achieve, to a value to try to promote. Perhaps the most crucial aspect of argumentation is the interaction between arguments. This means that argumentation can give us means for allowing an agent to reconcile conflicting information within itself, for reconciling its informational state with new perceptions from the environment, and for reconciling conflicting information between multiple agents through communication. A single agent may use argumentation techniques to perform its individual reasoning because it needs to make decisions under complex preferences policies, in a highly dynamic environment (see e.g. [6]). This is the case used in this research. In the following paragraphs we describe the theoretical framework that we adopted:

Definition 1. A **theory** is a pair $(\mathcal{T}, \mathcal{P})$ whose sentences are formulae in the **background monotonic logic** $(\mathcal{L}, \vdash)$ of the form $L \leftarrow L_1,\ldots,L_n$, where L, L_1, …, L_n are positive or negative ground literals. For rules in P the head L refers to an (irreflexive) higher priority relation, i.e. L has the general form $L = h\_p(\text{rule1}, \text{rule2})$. The derivability relation, $\vdash$, of the background logic is given by the simple inference rule of modus ponens.

An **argument** for a literal L in a theory $(\mathcal{T}, \mathcal{P})$ is any subset, T, of this theory that derives L, $T \vdash L$, under the background logic. A part of the theory $\mathcal{T}_0 \subset \mathcal{T}$, is the **background theory** that is considered as a non defeasible part (the indisputable facts).

An argument attacks (or is a counter argument to) another when they derive a contrary conclusion. These are conflicting arguments. A conflicting argument (from $\mathcal{T}$) is admissible if it counter-attacks all the arguments that attack it. It counter-attacks an argument if it takes along priority arguments (from $\mathcal{P}$) and makes itself at least as strong as the counter-argument (we omit the relevant definitions from [6] due to limited space).

Definition 2. An agent's **argumentative policy theory** is a theory $T = ((\mathcal{T}, \mathcal{T}_0), \mathcal{P}_R, \mathcal{P}_C)$ where $\mathcal{T}$ contains the **argument** rules in the form of definite Horn logic rules, $\mathcal{P}_R$ contains **priority** rules which are also definite Horn rules with head $h\_p(r_1, r_2)$ s.t. $r_1, r_2 \in \mathcal{T}$ and all rules in $\mathcal{P}_C$ are also priority rules with head $h\_p(R_1, R_2)$ s.t. $R_1, R_2 \in \mathcal{P}_R \cup \mathcal{P}_C$. $\mathcal{T}_0$ contains auxiliary rules of the agent's background knowledge.

Thus, in defining the decision maker's theory we specify three levels. The first level ($\mathcal{T}$) defines the (background theory) rules that refer directly to the subject domain, called the *Object-level Decision Rules*. In the second level we have the rules that define priorities over the first level rules for each *role* that the agent can assume or *context* that he can be in (including a *default context*). Finally, the third level rules define priorities over the rules of the previous level (which context is more important) but also over the rules of this level in order to define *specific contexts*, where priorities change again.

3.2 The Decision Maker's Argumentation Theory

Using the presented argumentation framework, we transformed the criteria for all MFs and experts knowledge (§2) to background theory (facts) and rules of the first and second level. Then, we defined the strategies (or specific contexts) in the third level rules.

The goal of the knowledge base is to select some MFs in order to construct our portfolio. Therefore our rules have as their head the predicate *selectFund/1* and its negation. We write rules supporting it or its negation and use argumentation for resolving conflicts. We introduce the *hasInvestPolicy/2*, *preference/1* and *market/1* predicates for defining the different contexts and roles. For example, John, an aggressive investor is expressed with the predicate *hasInvestPolicy(john, aggressive)*.

The knowledge base facts are the performance and risk variables values for each MF, the thresholds for each group of

values for each year and the above mentioned predicates characterizing the investor and the market. The following rules are an example of the object-level rules (level 1 rules of the framework - $\mathcal{T}$):

$r_1(Fund)$: $selectFund(Fund) \leftarrow highR(Fund)$

$r_2(Fund)$: $\neg selectFund(Fund) \leftarrow highB(Fund)$

The *highR* predicate denotes the classification of the MF as a high return fund and the *highB* predicate denotes the classification of the MF as a high risk fund. Thus, the r_1 rule states that a high performance fund should be selected, while the r_2 rule states that a high risk fund should not be selected. Such rules are created for the three groups of our performance and risk criteria.

Then, in the second level we assign priorities over the object level rules. The $\mathcal{P}_R$ are the *default context rules* or level 2 rules. These rules are added by experts and express their preferences in the form of priorities between the object level rules that should take place within defined contexts and roles. For example, the level 1 rules with signatures r_1 and r_2 are conflicting. In the default context the first one has priority, while the bear market context reverses this priority:

R_1: $h\_p(r_1(Fund), r_2(Fund)) \leftarrow true$

R_2: $h\_p(r_2(Fund), r_1(Fund)) \leftarrow market(bear)$

Rule R_1 defines the priorities set for the default context, i.e. an investor selects a fund that has high return on investment (RoI) even if it has high risk. Rule R_2 defines the default context for the bear market context (within which, the fund selection process is cautious and does not select a high RoI fund if it has high risk).

Finally, in $\mathcal{P}_C$ (level 3 rules) the decision maker defines his strategy and policy for integrating the different roles and contexts rules. When combining the *Aggressive investor* role and *bear market* context, for example, the final portfolio is their union except that the aggressive investor now would accept to select high and medium risk MFs (instead of only high). The decision maker's strategy sets preference rules between the rules of the previous level but also between rules at this level. Relating to the level 2 priorities, the bear market context's priority of not buying a high risk MF, even if it has a high return, is set at higher priority than that of the general context. Then, the specific context of an aggressive investor in a bear market defines that the bear market context preference is inverted. See the relevant priority rules:

C_1: $h\_p(R_2, R_1) \leftarrow true$

C_2: $h\_p(R_1, R_2) \leftarrow hasInvestPolicy(Investor, aggressive).$

C_3: $h\_p(C_2, C_1) \leftarrow true$

Thus, an aggressive investor in a bear market context would continue selecting high risk funds. In the latter case, the argument r_1 takes along the priority arguments R_1, C_2 and C_3 and becomes stronger (is the only admissible one) than the conflicting r_2 argument that can only take along the R_2 and C_1 priority arguments. Thus, the *selectFund(Fund)* predicate is true and the fund is inserted in the portfolio.

The problem with the above rules is that the facts *market(bear)* or (exclusive) *market(bull)* could not be safely determined for the next investment period. In the application version presented in [14] it was just assumed to remain the same as at the time of the investment. This strategy, however produced quite poor results for this context if it should change in the next period.

4 FORECASTING THE STATUS OF THE FINANCIAL MARKET

One of the most prominent issues in the field of signal processing is the adaptive filtering problem, with unknown time-invariant or time-varying parameters. Selecting the correct order and estimating the parameters of a system model is a fundamental issue in linear and nonlinear prediction and system identification. The problem of fitting an AutoRegressive Moving Aaverage model with eXogenous input (ARMAX) or a Nonlinear AutoRegressive Moving Aaverage model with eXogenous input (NARMAX) to a given time series has attracted much attention because it arises in a large variety of applications, such as time series prediction in economic and biomedical data, adaptive control, speech analysis and synthesis, neural networks, radar and sonar, fuzzy systems, and wavelets [5].

The forecasting algorithm used in this contribution is a generic applied evolutionary hybrid technique, which combines the effectiveness of adaptive multimodel partitioning filters and GAs' robustness [1]. This method has been first presented in [7]. Specifically, the a posteriori probability that a specific model, of a bank of the conditional models, is the true model, can be used as fitness function for the GA. In this way, the algorithm identifies the true model even in the case where it is not included in the filters' bank. It is clear that the filter's performance is considerably improved through the evolution of the population of the filters' bank, since the algorithm can search the whole parameter space. The proposed hybrid evolutionary algorithm can be applied to linear and nonlinear data; is not restricted to the Gaussian case; does not require any knowledge of the model switching law; is practically implementable, computationally efficient and applicable to online/adaptive operation; and exhibits very satisfactory performance as indicated by simulation experiments [2]. The structure of the hybrid evolutionary system used is depicted in Figure 1.

The representation used for the genomes of the population of the GA is the following. We use a mapping that transforms a fixed dimensional internal representation to variable dimensional problem instances. Each genome consists of a vector $\mathbf{x}$ of real values $x_i \in \Re$, $i = 1, ..., k$, and a bit string $\mathbf{b}$ of binary digits $b_i \in \{0,1\}$, $i = 1, ..., k$. Real values are summed up as long as the corresponding bits are equal. Obviously, k is an upper bound for the dimension of the resulting parameter vector. We use the first $k/3$ real values for the autoreggressive part, the second $k/3$ real values for the moving average part, and the last $k/3$ real values for the exogenous input part. An example of this mapping is presented in Figure 2. For a more detailed description of this mapping refer to [2].

At first, an initial population of m genomes is created at random (each genome consists of a vector of real values and a bit string). As stated before, each vector of real values represents a possible value of the NARMAX model order and its parameters. For each such population we apply an MMAF with EKFs and

have as result the model-conditional probability density function (pdf) of each candidate model. This pdf is the fitness of each candidate model, namely the fitness of each genome of the population (Figure 3).

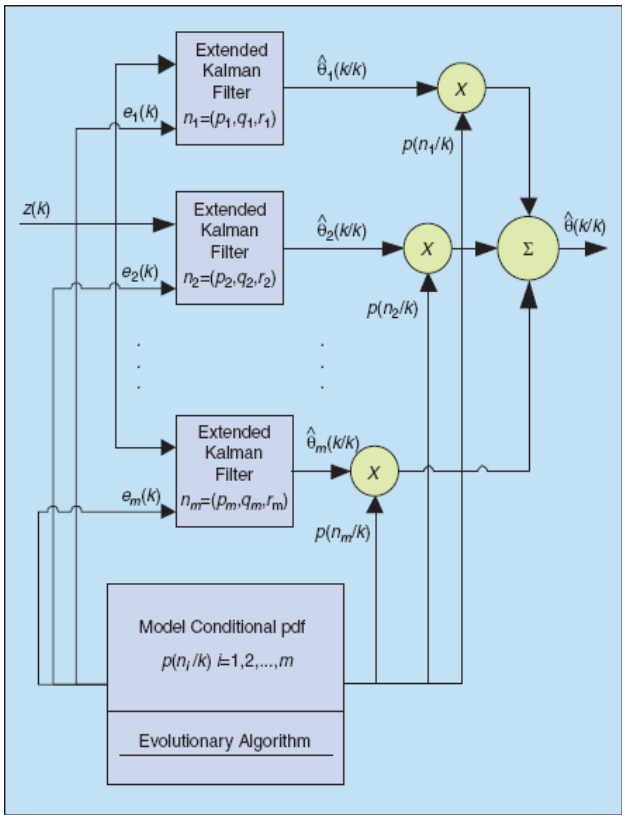


Figure 1: The structure of the hybrid evolutionary system used for forecasting

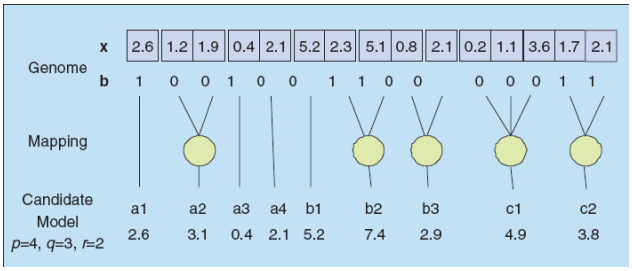


Figure 2: Mapping from a fixed dimensional internal representation to a variable length NARMAX parameter vector. The resulting order is n(p, q, r) = (4, 3, 2).

The reproduction operator we decided to use is the classic biased roulette wheel selection according to the fitness function value of each possible model order [9]. As far as crossover is concerned, we use the one-point crossover operator for the binary strings and the uniform crossover operator for the real values [9]. Finally, we use the flip mutation operator for the binary strings and the Gaussian mutation operator for the real values [9]. Every new generation of possible solutions iterates the same process as the old ones and all this process may be repeated as many generations as we desire or till the fitness function has value 1 (one) which is the maximum value it is able to have as a probability For a more detailed description of this hybrid evolutionary system refer to [2].

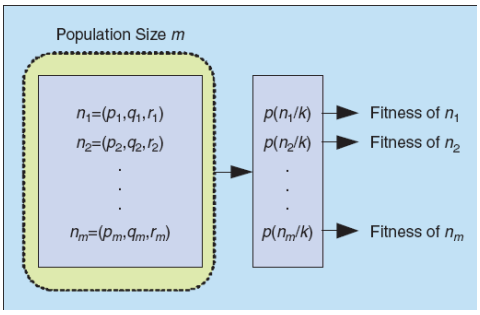


Figure 3: The fitness of each candidate model is the model conditional pdf (m is the number of the extended Kalman filters in the multimodel adaptive filter)

In this contribution we apply a slightly different approach compared to the one presented in [2]. In [2], at the algorithm's step where the value of the estimation (output) x of each filter is calculated, the past values of x that are used in order to estimate the next value of x are always taken from the estimation file (the file of all past values of x that have been estimated by the algorithm till this point). All these values are used in each generation in order to estimate the next value of the estimation (output) vector $\boldsymbol{x}$. The method presented in this contribution uses a different approach in order to estimate $\boldsymbol{x}$. At the algorithm's step where the value of x for each filter is calculated, the past values of x that are used in order to estimate the next value of x are smaller than the total length of the time series that has been estimated till this point. The length of past values used in each generation in order to estimate the next value of $\boldsymbol{x}$ equals to $n/2$, where n is the total length of the time series to be estimated. Every new value of x, estimated by the algorithm, is added to this time series of length $n/2$ and the oldest one is removed in order this time series to sustain a length of $n/2$. The value of $n/2$ was not selected arbitrarily. We have conducted exhaustive experiments using many different values. The value of $n/2$, that has been finally selected, was the most effective one, that is, the one that resulted in the best prediction results.

Thus, the hybrid evolutionary system presented in Figure 1 is used in order to forecast the behavior of the financial market in relation to its current status. The market is characterized as bull market if it is forecasted to rise in the next semester, or as bear market if it is forecasted to fall. We used the return values of the Greek market index for each semester starting from year 1985 to the years of our sample data (2000 to 2005). The algorithm performed very well considering that it could forecast the next semester market behavior with a success rate of 85.17% (12 out of 14 right predictions).

5 THE PORTFOLIO CONSTRUCTION APPLICATION

In this section we firstly present the system architecture, i.e. the combination method for the argumentation decision making sub-system and the hybrid forecasting sub-system that resulted in a

coherent application. Then we present the results of this combination.

5.1 System Architecture

The portfolio generation application is a Java program creating a human-machine interface and managing its modules, namely the decision making module, which is a prolog rule base (executed in SWI-prolog[1]) using the Gorgias[2] framework, and the forecasting module, which is a Matlab[3] implementation of the forecasting hybrid system (see Figure 4).

The application connects to the SWI-Prolog module using the provided Java interface (JPL) that allows for inserting facts to an existing rule-base and running it for reaching goals. The goals can be captured and returned to the Java program. The application connects to Matlab by executing it in a system shell. The matlab program writes the results of the algorithm to a MySQL[4] database using SQL (Structured Query Language). The application first executes the forecasting module, then updates the database, using JDBC (Java DataBase Connectivity interface) technology, with the investor profile (selected roles) and, finally, queries the decision making module setting as goal the funds to select for participation in the final portfolio. Thus, after the execution of the forecasting module the predicate *market/1* is determined as bull or bear and inserted as a fact in the rule base before the decision making process is launched. The reader can see in Figure 5 a screenshot of the integrated system.

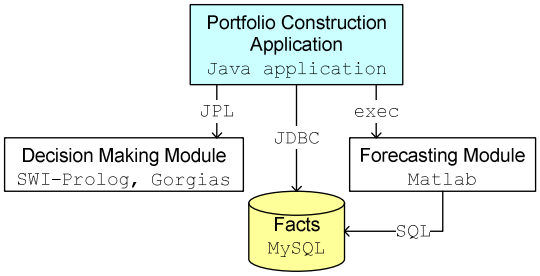


Figure 4: System Architecture

5.2 System Evaluation

For evaluating our results we defined scenarios for all years for which we had available data (2000-2005) and for all combinations of contexts. That resulted to the two investor types combined with the market status, plus the two investor types combined with the high performance option, plus the market status combined with the high performance option, all together five different scenarios run for six years each. Each one of the examined scenarios refers to different investment choices and leads to the selection of different number and combinations of MFs.

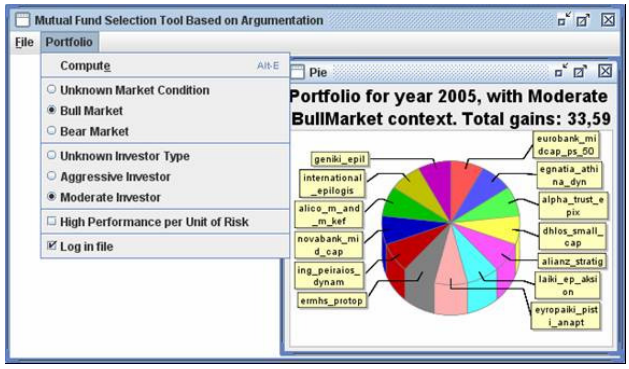


Figure 5: A screenshot for portfolio generation for a scenario of a moderate investor in a bull market context

In Table 1 the reader can inspect the average *return on investment* (RoI) for the six years for all different contexts. The reader should notice that the table contains two RoI columns, the first ("Previous RoI") depicts the results before changing the system as they appeared in [14]. The second presents the results of upgrading the application by combining it with the hybrid evolutionary forecasting sub-system and by fixing the selected funds participation to the final portfolio. The latter modification is out of the scope of this paper but the reader can clearly see that it has greatly influenced the performance of all scenarios.

Table 1, however, shows the added value of this contribution as the market context has become the most profitable in the "New RoI" column (8.17% RoI), while in the "Previous RoI" column it was one of the worst cases (3.72% RoI). Consequently the specific contexts containing the market context have better results.

Table 1: Average RoI for six years. The New RoI column shows the gains after the evolutionary hybrid forecasting system's integration

Context type	*Context*	*Previous RoI (%)*	*New RoI (%)*
simple	general	3.53	6.86
role	aggressive	2.65	7.38
role	moderate	4.02	6.09
context	**market**	**3.72**	**8.17**
role	high performance	4.98	7.16
specific context	aggressive – market	3.56	7.92
specific context	moderate – market	4.72	6.08
specific context	aggressive - high p.	4.88	7.46
specific context	moderate - high p.	4.98	7.16
specific context	Market - high perf.	4.59	7.23
ASE-GI	RASE	6.75	

Moreover, Table 1 also shows the added value of our approach as the reader can compare our results with the return on investment (RASE) of the General Index of the Athens Stock Exchange (ASE-GI). According to the results of this table, the average return of the constructed portfolios for all contexts, except two, achieves higher return than the market index. The two cases where the constructed portfolios did not beat the market index are the moderate simple context and moderate-market specific

[1] SWI-Prolog offers a comprehensive Free Software Prolog environment, http://www.swi-prolog.org

[2] Gorgias is an open source general argumentation framework that combines the ideas of preference reasoning and abduction, http://www.cs.ucy.ac.cy/~nkd/gorgias/

[3] MATLAB® is a high-level language and interactive environment for performing computationally intensive tasks, http://www.mathworks.com/products/matlab

[4] MySQL is an open source database, http://www.mysql.com

context. This is, maybe, due to the fact that in these two contexts we have an investor who wishes to earn more without taking into account any amount of risk in relation to the variability which characterizes the conditions of the market during the examined period. This fact makes it very difficult to implement investment strategies that can help a fund manager outperform a passive investment policy.

Furthermore, we notice that in some specific contexts the results are more satisfying than the results obtained by simple contexts, while in others there is little or no difference. This means that by using effective strategies in the third preference rules layer the decision maker can optimize the combined contexts. Specifically, the *aggressive-high performance* specific context provides better results than both the simple contexts *aggressive* and *high performance* (the ones that it combines) and the *general* context. The *moderate-high performance* specific context's returns on investment are equal to the higher simple context's returns (*high performance*) while the *aggressive-market* specific context returns are closer to the higher simple context's returns (*market*).

Finally, in Figure 6, we present the RoI of all contexts separately for each year. This view is also useful, as it shows that for two years, 2003 and 2004, RASE was greater than all our contexts RoI performance. This shows that our application, for the time being, performs better for medium term to long term investments, i.e. those that range over five years.

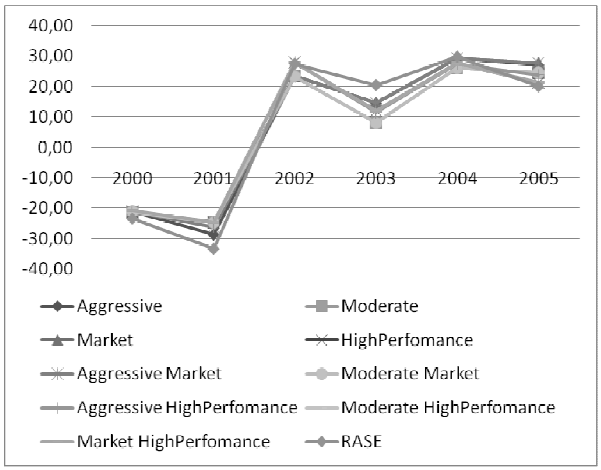


Figure 6: Comparative RoIs of all contexts for each year.

6 CONCLUSION

The objective of this paper was to present an artificial intelligence based application for the MF portfolio generation problem that combines two different intelligent methods, argumentation based decision making and a hybrid system that combines Genetic Algorithms (GA), MultiModel Partitioning (MMP) theory and the Extended Kalman Filter (EKF).

We described in detail how we developed our argumentation theory and how we combined it with the hybrid system to determine an important fact for the decision making process, i.e. the status of the financial market in the next investment period.

The developed application allows a decision maker (fund manager) to construct multi-portfolios of MFs under different, possibly conflicting contexts. Moreover, for medium to long term investments, the returns on investment of the constructed portfolios are better than those of the General Index of the Athens Stock Exchange, while the best results are those that involve the forecasting of the financial market.

Our future work will be to develop a new rule base for the problem of determining when to construct a new portfolio for a specific investor. We will also make the application web-based so that it can get on-line financial data available from the internet for computing the decision variables and for allowing the investors to insert their profiles by filling on-line forms. Finally, we will continue evaluating our application as new data become available for years after 2005. Our aim is to be able to guarantee a better RoI than that of the ASE.

7 REFERENCES

[1] A. V. Adamopoulos, Anninos, P. A., Likothanassis, S .D., Beligiannis, G. N., Skarlas, L. V., Demiris E. N. and Papadopoulos, P., 2002. Evolutionary Self-adaptive Multimodel Prediction Algorithms of the Fetal Magnetocardiogram, 14th Int. Conf. on Digital Signal Processing (DSP 2002), Vol. II, 1-3 July, Santorini, Greece, pp. 1149-1152.

[2] G. N. Beligiannis, L. V. Skarlas and S. D. Likothanassis. "A Generic Applied Evolutionary Hybrid Technique for Adaptive System Modeling and Information Mining", IEEE Signal Processing Magazine, 21(3), pp. 28-38, 2004

[3] G. Colson, and M. Zeleny, "Uncertain prospects ranking and portfolio analysis under the condition of partial information" in Mathematical Systems in Economics 44, 1979.

[4] G. Crina, A. Ajith, I. Hisao (Eds.), "Hybrid Evolutionary Algorithms", Studies in Computational Intelligence, Vol. 75, 2007

[5] S. Haykin, Adaptive Filter Theory. Englewood Cliffs, NJ: Prentice-Hall Int., 1991

[6] A. Kakas, and P. Moraitis, "Argumentation based decision making for autonomous agents", Proc. of the second Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS03), July 14-18, Australia, 2003.

[7] S.K. Katsikas, S.D. Likothanassis, G.N. Beligiannis, K.G. Berketis and D.A. Fotakis, "Evolutionary multimodel partitioning filters: A unified framework," IEEE Trans. Signal Processing, vol. 49, no. 10, pp. 2253–2261, 2001

[8] H. Markowitz, Portfolio Selection: Efficient Diversification of Investments, Wiley, New York, 1959.

[9] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed. New York: Springer-Verlag, 1996.

[10] I. Rahwan, P. Moraitis and C. Reed (Eds.) "Argumentation in Multi-Agent Systems", Lecture Notes in Artificial Intelligence, 3366, Springer-Verlag, Berlin, Germany, 2005

[11] S. Ross, "The Arbitrage Theory of Capital Asset Pricing", Journal of Economic Theory, 6, 1976, pp. 341-360.

[12] W.F. Sharpe, "Capital asset prices: A theory of market equilibrium under conditions of risk", Journal of Finance, 19, 1964, pp. 425-442.

[13] W.F., Sharpe, "Mutual fund performance", Journal of Business, 39, 1966, pp. 119-138.

[14] N. Spanoudakis and K. Pendaraki, "A Tool for Portfolio Generation Using an Argumentation Based Decision Making Framework" , in: Proceedings of the annual IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Patras, Greece, October 29-31, 2007

[15] J.L. Treynor, "How to rate management of investment funds", Harvard Business Review, 43, 1965, pp. 63-75.

An Architecture for Multiple Heterogeneous Case-Based Reasoning Employing Agent Technologies.

Elena I Teodorescu and Miltos Petridis[1]

Abstract. This paper presents an investigation into applying Case-Based Reasoning to Multiple Heterogeneous Case Bases using agents. The adaptive CBR process and the architecture of the system are presented. A case study is presented to illustrate and evaluate the approach. The process of creating and maintaining the dynamic data structures is discussed. The similarity metrics employed by the system are used to support the process of optimisation of the collaboration between the agents which is based on the use of a blackboard architecture. The blackboard architecture is shown to support the efficient collaboration between the agents to achieve an efficient overall CBR solution, while using case-based reasoning methods to allow the overall system to adapt and "learn" new collaborative strategies for achieving the aims of the overall CBR problem solving process.

1 Introduction

Case-based reasoning (CBR) is now an established artificial intelligence paradigm. Given a case-base of prior experiences, a CBR system solves new problems by retrieving cases from the case-base, and adapting their solutions to comply the new requirements[1].

Multiple Case Based Reasoning (MCBR) is used to retrieve solutions for a new problem from more than one case-base. Methods for managing sharing of standardized case bases have been studied in research on distributed CBR (e.g. [13]), as have methods for facilitating large-scale case distribution [10]. Leake and Sooriamuthhi propose a new strategy for MCBR - an agent selectively supplements its own case-base as needed, by dispatching problems to external case-bases and using cross-case-base adaptation to adjust their solutions for inter-case-base differences [4, 5, 6,13].

In many problems in modern organisations, the knowledge encapsulated by cases is contained in multiple case bases reflecting the fragmented way with which organisations capture and organise knowledge. The traditional approach is to merge all case bases into a central case base that can be used for the CBR process. However, this approach brings with it three challenges:

- Moving cases into a central case base potentially separates from its context and makes maintenance more difficult.
- Various case bases can use different semantics. There is therefore a need to maintain various ontologies and mappings across the case bases.
- The knowledge content "value" of individual cases can be related to its origination. This can be lost when merging into a central case base.

Keeping the cases distributed in the form of a Heterogeneous Multiple Case Based Reasoning system (HMCBR) may have a number of advantages such as increased maintainability and competence and the contextualisation of the cases. Past research at Greenwich [2][3] has shown the need to combine knowledge encoded in cases from various heterogeneous sources to achieve a competent, seamless CBR system.

Ontanon and Plaza [7] looked at a way to "improve the overall performance of the multiple case systems and of the individual CBR agents without compromising the agent's autonomy". They present [8] a framework for collaboration among agents that use CBR and strategies for case bartering (case trading by CBR agents). Nevertheless, they do not focus at the possibility of cases having different structures and what impact this will have on applying CBR to heterogeneous case bases. Leake [5] states that "An important issue beyond the scope of this paper is how to establish correspondences between case representations, if the representations used by different case-bases differ."

Given several case bases as the search domain, it is very likely that they have different structures. Ideally, accessing Multiple Case Bases should not require a change to their data structures. In order for an MCBR system to effectively use case-bases that may have been developed in different ways, for different tasks or task environments, methods are needed to adjust retrieved cases for local needs.

Leake and Sooriamurthi [4] proposed a theoretical "cross-case-base adaptation" which would adapt suggested solutions from one case base to apply to the needs of another. They are currently exploring sampling methods for comparing case-base characteristics in order to select appropriate cross-case-base adaptation strategies.

2 Adaptive CBR

In order to enable effective solution retrieval across autonomous case bases with differing structures, it is essential to have access and a good understanding of each of the different case base structures involved. This would make it possible to identify the commonalities, equivalences and specific characteristics of every case base associated with the system.

2.1 The process of adaptive CBR

Instead of trying to adapt the suggested solutions from one case base to the needs of another, the approach investigated in this study will be to create a "dynamic structure" of a general case. This dynamic structure would be modified every time a new case base with a new structure is added.

The process of adaptive CBR, within the architecture of the HMCBR System (Figure 1), will incorporate a number of steps.

Firstly, in order for the system to work with a particular case base, it will need to know the structure of that case base. Every newly added case base will therefore have to publish its structure to a Registry System. The published structures are required to have their own data dictionaries attached to enable the creation of a dynamic Data Dictionary.

[1] Department of Computing Science, University of Greenwich, Park Row, London SE10 9LS email:{ E.I.Teodorescu , M.Petridis}@gre.ac.uk

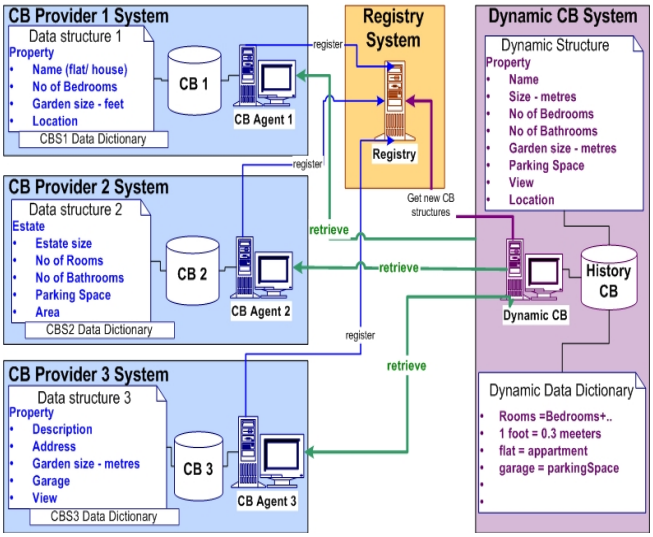


Fig. 1. The Architecture of the HMCBR System

The published structure will be retrieved by the Dynamic CB System and used to adapt the local dynamic structure to accommodate any new elements and map existing ones.

When the dynamic structure reflects all participating case bases, a case query can be submitted. The system would then reformulate the target case structure into each provider's case base structure. The target case structure will be a subset of the dynamic structure.

The reformulated cases are submitted to each provider and solution cases are retrieved using KNN techniques [1]. The structures of these solutions will be translated into the dynamic structure, thus creating a dynamic case base. Finally, the system will apply the classical CBR process to the dynamic case base.

The whole process is intended to provide a transparent view of the CBR process across the heterogeneous system.

2.2 Case Study

This case study requires searching for a property from three estate agencies without amalgamating their case bases structures.

Let us suppose that the estate agencies have different case base structures (figure 2).

A possible buyer should be able to search for a property and get all the suitable solutions from all three agencies. A search should retrieve the best matches from all case bases as if it was dealing with a single case base in a way transparent to the buyer.

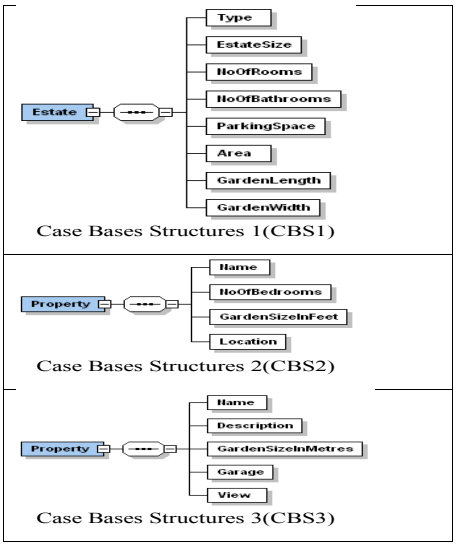


Fig. 2. Three different Case Base Structures

3 Creating the Dynamic Structure

Creating and maintaining a dynamic structure makes the self-adaptive multi case base reasoning system possible. By adding a new case base to the existing ones, new attributes are added to a global dynamic structure and new relations linked to these attributes are established.

CBS1. type / DCBS name	Apartment	Studio	Detached house
House	0	0	1
Flat	1	0.8	0

Fig. 3. Data Dictionary includes relations between some of the attributes.

A data dictionary is required to keep all the metadata for the dynamic structure. This data dictionary would have multiple functions: It records the location and the name of every attribute from the Case Base Structures (CBS) and how these are translated into the Dynamic Case Base Structure (DCBS). It also stores the type and any default value for every single attribute.

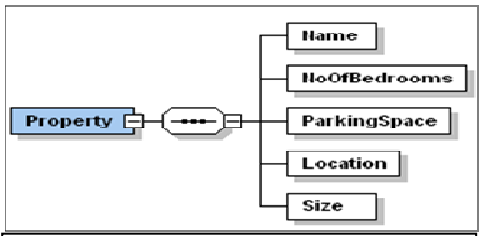
The Data Dictionary will reflect any relationships between the Dynamic Case Base Structure attributes. These relationships can be mathematical relationships or look-up tables (figure 3).

We will use the presented case study to show how a dynamic structure is created and how it is continuously changed by adding new case bases to the search domain.

Let us suppose that our general structure (the initial state of the Dynamic Structure containing few main attributes of a property) is already built (see figure 4). The structure has attached a basic Data Dictionary mainly containing the data types of the existing attributes.

We will show how this initial structure will be dynamically changed by consecutively adding the three agents to the search domain.

Adding the Case Base Structure 1 to the system implies mapping of the attributes ParkingSpace, Area and Type into the Dynamic Structure (these attributes are already existing in the initial structure) and also adding more attributes to it (i.e. NoOfRooms, NoOfBathrooms, GardenLength, GardenWidth)



Data Dictionary
Size: Double
NoOfBedrooms: Integer
Location: String
ParkingSpace: double

Name:	house	flat
house	1	0
flat	0	1

Fig. 4. Initial state of the Dynamic Structure and Data Dictionary

The Data Dictionary will reflect the mapping of attributes:

CBS1.ParkingSpace = DCBS.ParkingSpace;
CBS1.Area = DCBS.Location
CBS1.type= DCBS.name

The following attributes will be added to the dynamic data dictionary:

NoOfRooms: integer;
GardenLength: double; GardenWidth: double

Any other relevant relationships such as look-up tables for defining mappings between the values of attribute Type of CBS1 and

the values of the attribute Name of the dynamic structure will be captured.

Case Base Structure 2 will add another attribute, GardenSize, to the Dynamic Structure and the data dictionary will record mapping of attributes:

CBS2.Name = DCBS.Name,
CBS2.Location = DCBS.Location ,
CBS2.NoOfBedrooms = DCBS. NoOfBedrooms;

The mathematical relationships are recorded:
DCBS.GardenSize = DCBS.GardenLength * DCBS.GardenWidth,

Functions can be applied, for example to keep the same metric system:

DCBS.GardenSize= CBS2.GardenSizeInFeet/$(3.281)^2$

The Data Dictionary would also include a look-up table showing the conversion of values of CBS2.Name to values of DCBS.Name.

Attention has to be paid to the meanings of the names of the attributes. For example, if the attribute "Type" in CBS1 and the attribute "Name" in CBS2 have the same meaning (they would be translated as "Name" in DCBS, with values found in a look-up table), the attribute "Name" from CBS3 has not the same meaning as the one from CBS2. It is actually translated into DCBS.Location (similar to CBS2.Location)

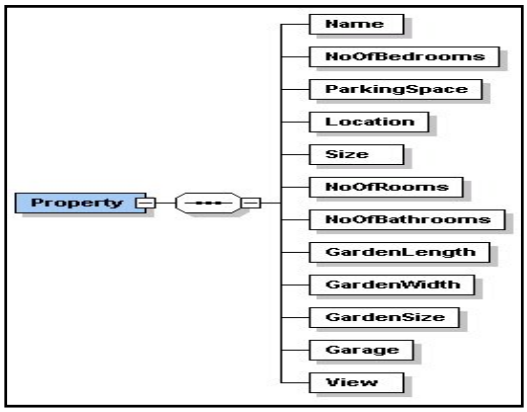


Fig. 5. Adapted Dynamic Structure after CBS3 was added

By adding the third estate agent case base to the search domain, the dynamic structure will grow even more (see figure 5) and the Data dictionary will reflect it by adding the attributes DSBS.Garage and DSBS.View.

The following attributes are mapped:

CBS3.Name = DCBS.Location
CBS3.Description = DCBS.Name
CBS3.GardenSizeInMeters = DCBS.GardenSize

Another look-up table can be created and added to the Data Dictionary to record the relationship between the Garage and ParkingSpace. Figure 6 shows the state of the Dynamic data Dictionary after CBS1, CBS2 and CBS3 are added.

Dynamic Data Dictionary
CBS1.Area = DCBS.Location
NoOfRooms: integer
CBS1.type= DCBS.name
DCBS.GardenSize: double
DCBS.GardenSize = CBS2.GardenSizeInFeet
DCBS.GardenSize = DCBS.GardenLenght *
DCBS.GardenWidth ...
CBS3.Name = DCBS.Location
CBS3.GardenSizeInMetres = DCBS.GardenSize

	Garage	ParkingSpace
Garage	1	0.7
ParkingSpace	0.7	1

Fig. 6. Adapted Dynamic Data Dictionary after CBS1, CBS2 and CBS3 are added

4 Optimising the agent collaboration process

In order to optimise the process of collaboration between the agents to achieve an efficient solution from the overall CBR process when applied across the heterogeneous case bases, an overall similarity metric is required. Additionally, an overall process to enable collaboration between the agents is necessary based on a flexible architecture to enable this collaboration.

4.1 Defining an overall similarity metric

The overall similarity metric between a target and a source Case can be defined as:

$$\sigma(C_T, C_s) = \sigma_{CBy}(C_T, C_s) * \omega_{CBy}(C_T) \qquad (1)$$

where:
σ: overall similarity
σ_{CBy}: similarity from case base provider CBy
C_T: target case
C_S: source case
$\omega_{CBy}(C_T)$: weighting for a case base provider y for case C_T

To allow for defining locally optimised similarity metrics for different providers, the following metric can be defined:

$$\sigma_{CBy}(C_T, C_s) = \sum_x \omega_{CBy}(x) * \sigma_{CBy}(C_T, C_s, x) \qquad (2)$$

where:
$\omega_{CBy}(x)$: the weighting from case base provider CBy for attribute x
$\sigma_{CBy}(C_T, C_s, x)$: the local similarity metric for provider CBy for attribute x.

This extended similarity metric takes into account the level of trust that the HMCBR system attributes to the competence of each case base provider. The level of trust is determined by applying CBR to the case-base of the history of queries. Additionally it allows to adjust the trust to particular providers to different "regions" in the case base allowing for case base providers to be "specialised" on particular types of domain knowledge. Finally, the extended metric allows for different ways of defining similarity based on possible particularities pertaining to individual case base providers.

Let us assume that in our case study the third estate agent is specialised in city apartments. After a few searches for country side houses with gardens, reasoning can be applied to the History case-base. Results will show that, for this particular query, the estate agent's level of trust is not high, i.e. there will be less solutions for this particular case base added to the Dynamic case-base.

A global level of trust of a provider's case-base can be calculating taking in consideration the results of all the previous enquiries for that provider.

4.2 An architecture and process to support effective collaboration between case base agents

The architecture of the HMCBR system shown in figure 1 contains the dynamic CB system, which incorporates a blackboard architecture. Blackboards have been used very effectively in the past for the construction of hybrid and agent based AI systems [11], [12].

The dynamic CB system is where the process for agent collaboration is controlled. It is based on a blackboard architecture incorporating the blackboard containing the target and retrieved cases from various providers together with similarity calculations and rankings. The blackboard also contains a log of the solution process and the reconciliation strategy followed, thus representing the state of the overall CBR solution process at any point in this process. Figure 7 shows the structure of the dynamic CB module incorporating the blackboard architecture.

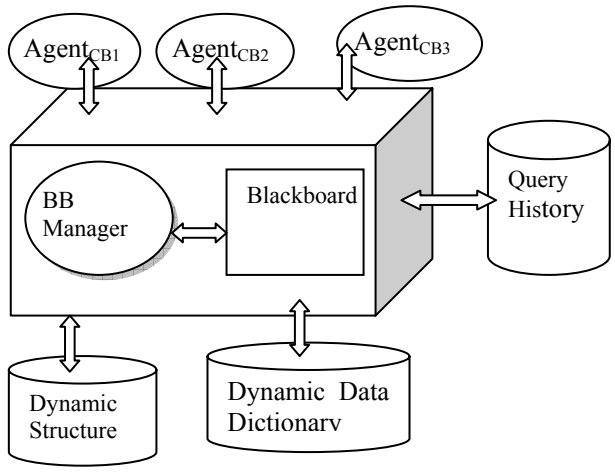


Fig. 7. The Dynamic CB system incorporating the blackboard architecture

The blackboard manager manages the overall solution process, communicates with and keeps track of the CB agents, selects and implements a solution strategy and monitors and evaluates the solutions achieved. Given a new target case, the blackboard manager decides on strategy for finding similar cases from the CB providers. The blackboard system decides which CB providers to use and the number of cases to retrieve from each one and other requirements, such as the requirement for diversity, similarity thresholds etc. The system then initialises the agents and assigns to them a mission. On return, the results (cases) are mapped using the dynamic data dictionary and written to the blackboard. A "global" CBR process is used to decide on the retrieved cases. The system then selects and presents the shortlisted cases after the reconciliation process and provides these to the user, together with links to their original forms for the user to explore. Finally, the system "reflects" on the process by updating the query history and confidence weights for each provider.

The system described here has been implemented and tested on a set of case bases from three different estate agent case bases, all using different structures. Experiments with the system have shown that the system can retrieve useful cases combining cases from all case bases to provide a more efficient overall solution when compared to using the case bases separately or mapping them to one central case base. Additionally, the system has shown that it can provide a more diverse retrieved case population in both cases. A full scale evaluation of the system, including using a different application domain is under way.

5 Conclusion

At a time of increasing web-based communication and sharing of knowledge between organisations and organisational units within enterprises, heterogeneous CBR applied to Multiple Case Bases seems to be the natural progression in this area of research.

The paper investigates an approach based on agents operating on different structures/views of the problem domain in a transparent and autonomous way. In this approach all data is kept locally by each case base provider in its native form. Agents can be dynamically added to the system, thus increasing the search domain and potentially the competence and vocabulary of the system.

This research proposes a new architecture for a self-adaptive MCBR system which involves the use of a dynamic structure based on the blackboard architecture. The Dynamic Structure reflects all participating case base provider structures. As new agents are added to the system, their case base structure is published and is used to adapt the Dynamic Structure accordingly.

The Dynamic Structure is used at runtime to translate search queries into the local structures of each agent. Each agent can then use the translated query to match it to its local cases and retrieve the best matches.

A Data Dictionary is created in order to manage the Dynamic Structure. This contains the metadata for the Dynamic Structure, such as mapping details of the case base provider's structures to the Dynamic Structure, type information and relationships between attributes of the dynamic structure.

The dynamic case base system manages the overall process, including controlling the agents, reconciling and optimising the retrieved cases and feeding back into its strategy by continuously adjusting weights representing confidence levels on individual case base providers. A prototype system to evaluate the efficiency of using a heterogeneous Multiple Case Based Reasoning system is currently being evaluated. Preliminary findings are encouraging.

Further work will concentrate into optimising the process of collaboration between the agents and methods and strategies for the reconciliation of retrieved cases.

References

[1] Kolodner, J.: Case-based Reasoning, Morgan Kaufmann, 1993

[2] Knight B, Petridis M, Mileman T: Maintenance of a Case-Base for the Retrieval of Rotationally Symmetric Shapes for the Design of Metal Castings LNAI 1998: Springer Verlag. Advances in Case-Based Reasoning, 5th European Workshop, Trento, Italy, Sept. 6-9.2000 pp 418-430

[3] Knight B, Petridis M, Mejasson P, Norman P: A Intelligent design assistant (IDA): a CBR System for Materials Design Journal of Materials and Design 22 pp 163-170, 2001

[4] David Leake & Raja Sooriamurthi: "Automatically Selecting Strategies for Multi-Case-Base Reasoning" - Proceedings of the Sixth European Conference on Case-Based (ECCBR) , Aberdeen, Scotland, September, 2002

[5] David Leake & Raja Sooriamurthi : "Managing Multiple Case Bases: Dimensions and Issues" (Proceedings of the Fifteenth Florida Artificial Intelligence Research Society (FLAIRS) Conference , Pensacola, Florida, May 2002),

[6] David Leake & Raja Sooriamurthi: "When Two Case Bases are Better Than One: Exploiting Multiple Case Bases" (Proceedings of the Fourth International Conference on Case-Based Reasoning (ICCBR) , Vancouver, Canada, July, 2001).

[7] Enric Plaza, Santiago Ontañón: Cooperative Multiagent Learning. Adaptive Agents and Multi-Agents Systems 2002

[8] Santiago Ontañón, Enric Plaza: A bartering approach to improve multiagent learning. AAMAS 2002

[9] Francisco J. Martín, Enric Plaza, Josep Lluís Arcos: Knowledge and Experience Reuse Through Communication Among Competent (Peer) Agents. International Journal of Software Engineering and Knowledge Engineering 1999

[10] Hayes, C., Doyle, M., Cunningham, P.: Distributed CBR Using XML, Proceedings of the Fourth European Conference on Case-Based (ECCBR) , Dublin, Ireland, June 1998

[11]R Engelmore, T Morgan : Blackboard Systems, Addison-Wesley, 1988

[12] Petridis M, Knight B, (2001): "A blackboard architecture for a hybrid CBR system for scientific software" in Proceedings of the Workshop Program at the 4th International Conference in Case-based Reasoning ICCBR01, Vancouver 2001, Eds R. Weber, G. von Wagenheim, pp 190-195, NCARAI, Naval Research Laboratory, Code 5510 Washington DC.

[13] Case Dispatching versus Case-Base Merging: when MCBR matters: David Leake and Raja Sooriamurthi. International Journal on Artificial Intelligence Tools: Architectures, Languages and Algorithms (IJAIT). Special issue on recent advances in techniques for intelligent systems. Vol 13, No 1, 2004