IRIS DATA CLASSIFICATION BY MEANS OF PSEUDO NEURAL NETWORKS BASED ON EVOLUTIONARY SYMBOLIC REGRESSION

Zuzana Kominkova Oplatkova, Roman Senkerik

Tomas Bata University in Zlin, Faculty of Applied Informatics
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{oplatkova, senkerik}@fai.utb.cz

KEYWORDS
Iris data, Pseudo Neural Network, Analytic programming, Differential Evolution.

ABSTRACT

This research deals with a novel approach to classification. Iris data was used for the experiments. Classical artificial neural networks, where a relation between inputs and outputs is based on the mathematical transfer functions and optimized numerical weights, was an inspiration for this work. Artificial neural networks need to optimize weights, but the structure and transfer functions are usually set up before the training. The proposed method utilizes the symbolic regression for synthesis of a whole structure, i.e. the relation between inputs and output(s) and tested on iris data in this case. For experimentation, Differential Evolution (DE) for the main procedure and also for meta-evolution version of analytic programming (AP) was used.

INTRODUCTION

The interest about classification by means of some automatic process has been enlarged with the development of artificial neural networks (ANN). They can be used also for a lot of other possible applications like pattern recognition, prediction, control, signal filtering, approximation, etc. All artificial neural networks are based on some relation between inputs and output(s), which utilizes mathematical transfer functions and optimized weights from training process. The setting-up of layers, number of neurons in layers, estimating of suitable values of weights is a demanding procedure. On account of this fact, pseudo neural networks, which represent the novelty approach using symbolic regression with evolutionary computation, is proposed in this paper.

Symbolic regression in the context of evolutionary computation means to build a complex formula from basic operators defined by users. The basic case represents a process in which the measured data is fitted and a suitable mathematical formula is obtained in an analytical way. This process is widely known for mathematicians. They use this process when a need arises for mathematical model of unknown data, i.e. relation between input and output values. The symbolic regression can be used also for design of electronic circuits or optimal trajectory for robots and within other applications (Back et al., 1997), (Koza, 1998), (Koza, 1999), (O'Neill et al., 2003), (Zelinka et al., 2011), (Oplatkova, 2009), (Varacha et al., 2006). Everything depends on the user-defined set of operators. The proposed technique is similar to synthesis of analytical form of mathematical model between input and output(s) in training set used in neural networks. Therefore it is called Pseudo Neural Networks.

Initially, John Koza proposed the idea of symbolic regression done by means of a computer in Genetic Programming (GP) (Back et al., 1997), (Koza, 1998), (Koza, 1999). The other approaches are e.g. Grammatical Evolution (GE) developed by Conor Ryan (O'Neill et al., 2003) and here described Analytic Programming (Zelinka et al., 2011), (Oplatkova, 2009), (Varacha et al., 2006).

The above-described tools were recently commonly used for synthesis of artificial neural networks but in a different manner than is presented here. One possibility is the usage of evolutionary algorithms for optimization of weights to obtain the ANN training process with a small or no training error result. Some other approaches represent the special ways of encoding the structure of the ANN either into the individuals of evolutionary algorithms or into the tools like Genetic Programming. But all of these methods are still working with the classical terminology and separation of ANN to neurons and their transfer functions (Fekiac, 2011). In this paper, the proposed technique synthesizes the structure without a prior knowledge of transfer functions and inner potentials. It synthesizes the relation between inputs and output of training set items used in neural networks so that the items of each group are correctly classified according the rules for cost function value. The data set used for training is Iris data set (Machine Learning Repository, Fisher 1936). It is a very known benchmark data set for classification problem, which was introduced by Fisher for the first time.

Firstly, Analytic Programming used as a symbolic regression tool is described. Subsequently Differential Evolution used for main optimization procedure within Analytic Programming and also as a second algorithm within metaevolution purposes is mentioned. After that a brief description of artificial neural network (ANN) follows. Afterwards, the proposed experiment with differences compared to classical ANN is explained. The result section and conclusion finish the paper.

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

ANALYTIC PROGRAMMING

Basic principles of the AP were developed in 2001 (Zelinka et al., 2005), (Zelinka et al., 2008), (Oplatkova, 2009), (Zelinka et al., 2011). Until that time only genetic programming (GP) and grammatical evolution (GE) had existed. GP uses genetic algorithms while AP can be used with any evolutionary algorithm, independently on individual representation. To avoid any confusion, based on use of names according to the used algorithm, the name - Analytic Programming was chosen, since AP represents synthesis of analytical solution by means of evolutionary algorithms.

The core of AP is based on a special set of mathematical objects and operations. The set of mathematical objects is set of functions, operators and so-called terminals (as well as in GP), which are usually constants or independent variables. This set of variables is usually mixed together and consists of functions with different number of arguments. Because of a variability of the content of this set, it is called here "general functional set" – GFS. The structure of GFS is created by subsets of functions according to the number of their arguments. For example GFS_{all} is a set of all functions, operators and terminals, GFS_{3arg} is a subset containing functions with only three arguments, GFS_{0arg} represents only terminals, etc. The subset structure presence in GFS is vitally important for AP. It is used to avoid synthesis of pathological programs, i.e. programs containing functions without arguments, etc. The content of GFS is dependent only on the user. Various functions and terminals can be mixed together (Zelinka et al., 2005), (Zelinka et al., 2008), (Oplatkova, 2009).

The second part of the AP core is a sequence of mathematical operations, which are used for the program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is a mapping from an individual domain into a program domain. This mapping consists of two main parts. The first part is called discrete set handling (DSH) (See Figure 1) (Zelinka et al., 2005), (Lampinen and Zelinka, 1999) and the second one stands for security procedures which do not allow synthesizing pathological programs. The method of DSH, when used, allows handling arbitrary objects including nonnumeric objects like linguistic terms {hot, cold, dark…}, logic terms (True, False) or other user defined functions. In the AP DSH is used to map an individual into GFS and together with security procedures creates the above mentioned mapping which transforms arbitrary individual into a program.

AP needs some evolutionary algorithm (Zelinka, 2004) that consists of population of individuals for its run. Individuals in the population consist of integer parameters, i.e. an individual is an integer index pointing into GFS. The creation of the program can be schematically observed in Fig. 2. The individual contains numbers which are indices into GFS. The detailed description is represented in (Zelinka et al., 2005), (Zelinka et al., 2008), (Oplatkova et al., 2009).

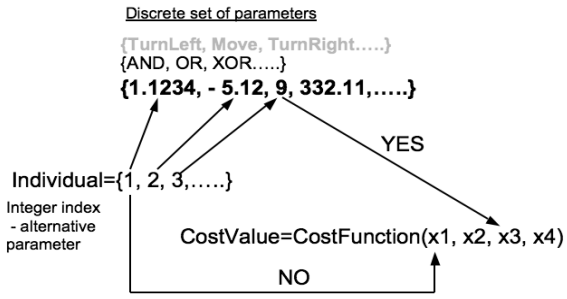

Figure 1: Discrete set handling

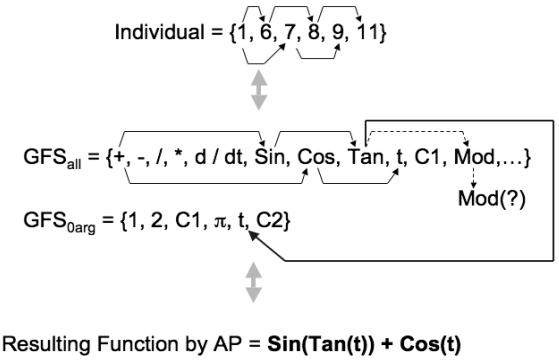

Figure 2: Main principles of AP

AP exists in 3 versions – basic without constant estimation, AP_{nf} – estimation by means of nonlinear fitting package in Mathematica environment and AP_{meta} – constant estimation by means of another evolutionary algorithms; meta means metaevolution.

ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are inspired in the biological neural nets and are used for complex and difficult tasks (Hertz et al., 1991), (Wasserman, 1980), (Gurney, 1997), (Fausset, 2003). The most often usage is classification of objects as also in this case. ANNs are capable of generalization and hence the classification is natural for them. Some other possibilities are in pattern recognition, control, filtering of signals and also data approximation and others.

There are several kinds of ANN. Simulations were based on similarity with feedforward net with supervision. ANN needs a training set of known solutions to be learned on them. Supervised ANN has to have input and also required output.

The neural network works so that suitable inputs in numbers have to be given on the input vector. These inputs are multiplied by weights which are adjusted during the training. In the neuron the sum of inputs multiplied by weights are transferred through mathematical function like sigmoid, linear, hyperbolic tangent etc. Therefore ANN can be used for data approximation (Hertz et al., 1991) – a regression model on measured data, relation between input and required (measured data) output.

These single neuron units (Fig. 3) are connected to different structures to obtain different structures of ANN (e.g. Fig. 4 and Fig. 5), where $\sum \delta = TF[\sum (w_i x_i + b w_b)]$ and $\sum = TF[\sum (w_i x_i + b w_b)]$; TF is logistic sigmoid function in this case.

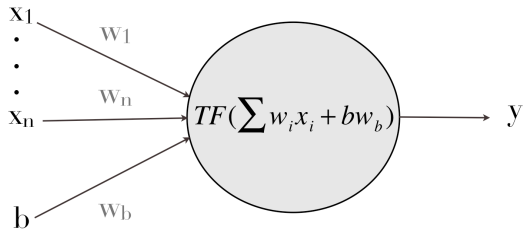

Figure 3: Neuron model, where TF (transfer function like sigmoid), x_1 - x_n (inputs to neural network), b – bias (usually equal to 1), w_1 – w_n, w_b – weights, y – output

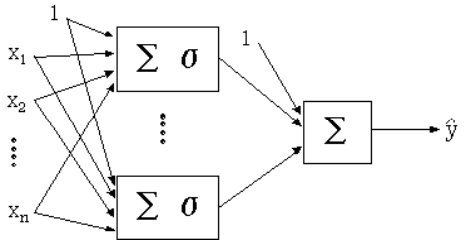

Figure 4: ANN models with one hidden layer

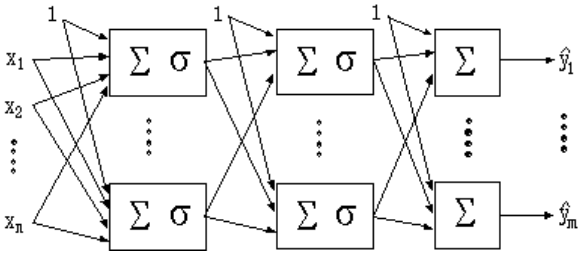

Figure 5: ANN models with two hidden layers and more outputs

The example of relation between inputs and output can be shown as a mathematical form (1). It represents the case of only one neuron and logistic sigmoid function as a transfer function.

$$y = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}}, \qquad (1)$$

where y – output
x_1, x_2 – inputs
w_1, w_2 – weights.

The aim of the proposed technique is to find similar relation to (1). This relation is completely synthesized by evolutionary symbolic regression – Analytic Programming.

USED EVOLUTIONARY ALGORITHMS

This research used one evolutionary algorithm Differential Evolution (Price, 2005) for both parts – main procedure and metaevolutionary part of the analytic programming. Future simulations expect a usage of soft computing GAHC algorithm (modification of HC12) (Matousek, 2007) and a CUDA implementation of HC12 algorithm (Matousek, 2010).

Differential evolution

DE is a population-based optimization method that works on real-number-coded individuals (Price, 2005). For each individual $\vec{x}_{i,G}$ in the current generation G, DE generates a new trial individual $\vec{x}'_{i,G}$ by adding the weighted difference between two randomly selected individuals $\vec{x}_{r1,G}$ and $\vec{x}_{r2,G}$ to a randomly selected third individual $\vec{x}_{r3,G}$. The resulting individual $\vec{x}'_{i,G}$ is crossed-over with the original individual $\vec{x}_{i,G}$. The fitness of the resulting individual, referred to as a perturbed vector $\vec{u}_{i,G+1}$, is then compared with the fitness of $\vec{x}_{i,G}$. If the fitness of $\vec{u}_{i,G+1}$ is greater than the fitness of $\vec{x}_{i,G}$, then $\vec{x}_{i,G}$ is replaced with $\vec{u}_{i,G+1}$; otherwise, $\vec{x}_{i,G}$ remains in the population as $\vec{x}_{i,G+1}$. DE is quite robust, fast, and effective, with global optimization ability. It does not require the objective function to be differentiable, and it works well even with noisy and time-dependent objective functions. Description of used DERand1Bin strategy is presented in (2). Please refer to (Price and Storn 2001, Price 2005) for the description of all other strategies.

$$u_{i,G+1} = x_{r1,G} + F \bullet \left(x_{r2,G} - x_{r3,G}\right) \qquad (2)$$

PROBLEM DESIGN – IRIS PLANT DATA SET DEFINITION

For this classification problem, iris data set was used (Machine Learning Repository, Fisher 1936). This set contains 150 instances. Half amount was used as training data and the second half was used as testing data. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. Each instance has 4 attributes (sepal length, sepal width, petal length and petal width) and type of class – iris virginica (Fig. 6), iris versicolor (Fig. 7) and iris sentosa (Fig. 8). The attributes were of real values. Usually, the class is defined by 3 output neurons in classical artificial neural net and binary code. In this paper, AP was able to create pseudo neural net structure between inputs and one output. Therefore, three classes were designed as continues value of the output, i.e. iris setosa was -1 for output less than -1, iris virginica has value 1 for output

greater than 1, and iris versicolor was 0 for outputs between -1 and 1.
The cost function value is given in eq. (3), i.e. if the cv is equal to zero, all training patterns are classified correctly.

$$cv = \sum_{i=1}^{n} |requiredOutput - currentOutput| \qquad (3)$$


Figure 6**:** Iris virginica


Figure 7: Iris versicolor


Figure 8: Iris setosa

RESULTS

As described in section about Analytic Programming, AP requires some EA for its run. In this paper AP_{meta} version was used. Meta-evolutionary approach means usage of one main evolutionary algorithm for AP process and second algorithm for coefficient estimation, thus to find optimal values of constants in the structure of pseudo neural networks.
In this paper, DE was used for main AP process and also in the second evolutionary process. Settings of EA parameters for both processes were based on performed numerous experiments with chaotic systems and simulations with AP_{meta} (Table 1 and Table 2).

Table 1: DE settings for main process

PopSize	20
F	0.8
CR	0.8
Generations	50
Max. CF Evaluations (CFE)	1000

Table 2: DE settings for meta-evolution

PopSize	40
F	0.8
CR	0.8
Generations	150
Max. CF Evaluations (CFE)	6000

The set of elementary functions for AP was inspired in the items used in classical artificial neural nets. The components of input vector x contain values of each attribute (x_1, x_2, x_3, x_4).

Basic set of elementary functions for AP:
GFS2arg= +, -, /, *, ^, exp
GFS0arg= x_1, x_2, x_3, x_4, K

Total number of cost function evaluations for AP was 1000, for the second EA it was 6000, together 6 millions per each simulation.

From carried simulations, several notations of input dependency were obtained, e.g. (4) and (5). The case (4) had a training error equal to 2 misclassified items (it means 2.66% error, 97.34% success from all 75 training patterns) and testing error equal to 3 misclassified items (4% error, 96% success). The case (5) was worse, during training phase the error equal to 3 misclassified patterns was reached and testing error was equal to 5.

Following numbers (6) are the outputs from the pseudo neural network for the training set, which has been then saturated to 3 numbers -1, 0 and 1 according to the class type interval (7). The (8) and (9) is the same way obtained numbers for the testing set.

$$y = \left(-\frac{7.57336\, e^{-x_3}\, x_2}{x_4} - x_3 + x_4\right) \exp\left(-6.91605 \times 10^{117344.416^{e^{-x_2^{x_4}}\left(x_1 + 1.0796 \times 10^{-233 x_1} - e^{x_2}\right) - e^{-x_4} - e^{x_2}}}\right) \quad (4)$$

$$y = \left(e^{-0.0066007\,(x_4 + 693.44)}\right)^{-0.00311885\,(x_3 + e^{x_3} - 144.991)} - 953.535\, e^{x_3\,(-x_4 - 637.961)} \left(-x_2 - x_4 + \frac{6.11200456962 \times 10^{625}}{(x_2\, x_4)^{223.656}}\right) \quad (5)$$

classical output = {-2.32401, -1.82401, -2.02399, -1.92404, -2.42401, -2.51714, -2.1247, -2.22404, -1.72401, -2.01559, -2.52404, -2.22406, -1.91558, -1.91555, -2.82397, -3.01703, -2.51692, -2.2247, -2.52481, -2.52474, -2.22408, -2.31703, -2.42392, -1.80036, -2.22413, -0.100247, 0.162017, 0.259989, 0.561469, 0.56151, 0.0595996, 0.344323, -0.153875, -0.0407742, 0.403247, 0.245674, 0.36354, 0.0445453, 0.199753, -0.037034, 0.00106313, 0.362017, -0.45568, 1.16202, -0.0659825, 1.08403, 0.0614693, 0.859989, -0.163289, -0.0396526, 4.23024, 1.94265, 2.4494, 1.37783, 2.90987, 2.44134, 1.45294, 1.37241, 1.77628, 3.92834, 1.8341, 1.94088, 2.45401, 2.53511, 4.15996, 3.21541, 1.27861, 2.0981, 3.79167, 1.15948, 3.20947, 2.23613, 2.21791, 1.58326, 2.15171} (6)

saturated output to class type = {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 1., 0, 0, 0, 0, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.} (7)

testing classical output = {-1.82406, -2.01709, -2.32404, -2.22401, -2.02406, -1.92406, -2.01703, -3.01559, -3.02401, -2.01559, -2.02397, -2.32399, -2.01559, -1.82399, -2.22404, -2.22466, -1.02466, -2.02399, -1.87359, -2.41725, -1.7247, -2.62406, -2.02401, -2.52404, -2.12401, 0.101063, 0.299317, 0.949556, 0.462017, -0.354326, 0.0342867, -0.154778, -0.0607432, 0.941977, 0.362017, 0.245497, 0.261003, 0.559973, -0.138905, 0.361469, 0.0376652, 0.10019, 0.0389384, -0.053875, 0.160721, -0.361698, -0.0392787, -0.0396526, -0.063559, 0.0610953, 1.07473, 1.48403, 1.28326, 2.65286, 0.637876, 1.83378, 1.22094, 3.11249, 0.558975, 0.495828, 3.40353, 3.55155, 1.17861, 1.28403, 2.35517, 3.85155, 3.31839, 1.94265, 3.2065, 4.23593, 3.4169, 2.14354, 2.03309, 3.01393, 1.28171} (8)

testing saturated output to class type = {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 1., 1., 1., 0, 1., 1., 1., 0, 0, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.} (9)

From the above values, it can be visible which items were classified in a wrong group. During training iris versicolor was misclassified twice as iris sentosa, in the testing phase it was viceversa – iris sentosa as iris versicolor.

CONCLUSION

This paper deals with a novel approach – pseudo neural networks. Within this approach, classical optimization of the structure or weights was not performed. The proposed technique is based on symbolic regression with evolutionary computation. It synthesizes a whole structure in symbolic form without a prior knowledge of the ANN structure or transfer functions. It means that the relation between inputs and output(s) is synthesized. As can be seen from the result section, such approach is promising. For further tests some observed critical points have to be taken into consideration. Future plans will be focused on further tests and comparisons with classical neural nets.

ACKNOWLEDGEMENT

This work was supported by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

REFERENCES

Back T., Fogel D. B., Michalewicz Z., *Handbook of evolutionary algorithms*, Oxford University Press, 1997, ISBN 0750303921

Fausett L. V.: Fundamentals of Neural Networks: Architectures, Algorithms and Applications, Prentice Hall, 1993, ISBN: 9780133341867

Fekiac J., Zelinka I., Burguillo J. C.: A review of methods for encoding neural network topologies in evolutionary computation, ECMS 2011, Krakow, Poland, ISBN: 978-0-9564944-3-6

Fisher R. A. (1936). "The use of multiple measurements in taxonomic problems". Annals of Eugenics 7 (2): 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x

Gurney K.: An Introduction to Neural Networks, CRC Press, 1997, ISBN: 1857285034

Hertz J., Kogh A. and Palmer R. G.: Introduction to the Theory of Neural Computation, Addison – Wesley 1991

Koza J. R. et al., *Genetic Programming III; Darwinian Invention and problem Solving*, Morgan Kaufmann Publisher, 1999, ISBN 1-55860-543-6

Koza J. R., *Genetic Programming,* MIT Press, 1998, ISBN 0-262-11189-6

Lampinen J., Zelinka I., 1999, "New Ideas in Optimization – Mechanical Engineering Design Optimization by

Differential Evolution", Volume 1, London: McGraw-hill, 1999, 20 p., ISBN 007-709506-5.

Machine learning repository with Iris data set http://archive.ics.uci.edu/ml/datasets/Iris

Matousek R., 2010, „HC12: The Principle of CUDA Implementation". In MENDEL 2010, Mendel Journal series, pp. 303-308. ISBN: 978-80-214-4120- 0. ISSN: 1803- 3814.

Matousek R., 2007, „GAHC: Improved GA with HC station", In WCECS 2007, San Francisco, pp. 915-920. ISBN: 978-988-98671-6-4.

O'Neill M., Ryan C., *Grammatical Evolution. Evolutionary Automatic Programming in an Arbitrary Language*, Kluwer Academic Publishers, 2003, ISBN 1402074441

Oplatkova Z.: Metaevolution: Synthesis of Optimization Algorithms by means of Symbolic Regression and Evolutionary Algorithms, Lambert Academic Publishing Saarbrücken, 2009, ISBN: 978-3-8383-1808-0

Price K., Storn R. M., Lampinen J. A., 2005, "Differential Evolution : A Practical Approach to Global Optimization", (Natural Computing Series), Springer; 1 edition.

Price, K. and Storn, R. (2001), *Differential evolution homepage*, [Online]: http://www.icsi.berkeley.edu/~storn/code.html, [Accessed 29/02/2012].

Wasserman P. D.: Neural Computing: Theory and Practice, Coriolis Group, 1980, ISBN: 0442207433

Zelinka et al.: Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures, in Kita E.: Evolutionary Algorithms, InTech 2011, ISBN: 978-953-307-171-8

Zelinka I., Varacha P., Oplatkova Z., *Evolutionary Synthesis of Neural Network*, Mendel 2006 – 12th International Conference on Softcomputing, Brno, Czech Republic, 31 May – 2 June 2006, pages 25 – 31, ISBN 80-214-3195-4

Zelinka I.,Oplatkova Z, Nolle L., 2005. *Boolean Symmetry Function Synthesis by Means of Arbitrary Evolutionary Algorithms-Comparative Study*, International Journal of Simulation Systems, Science and Technology, Volume 6, Number 9, August 2005, pages 44 - 56, ISSN: 1473-8031.