Active Sonar Target Identification Using Evolutionary Neural Logic Networks

Athanasios Tsakonas[1] and **Georgios Dounias**[2] and **Nikitas Nikitakos**[3]

Abstract. A real-world problem is addressed in this work using a novel approach belonging to the area of neural-symbolic systems. Specifically, we apply evolutionary techniques for the development of neural logic networks of arbitrary length and topology. The evolutionary algorithm consists of grammar guided genetic programming using cellular encoding for the representation of neural logic networks into population individuals. The application area is related to the classification of active sonar signals. Our aim is to demonstrate the capability of the system to produce competitive to feedforward neural networks results, yet potentially interpretable. Our experiments show that the overall system is capable to generate arbitrarily connected and competitive evolved solutions for the active sonar target identification, leading potentially to knowledge extraction.

1 INTRODUCTION

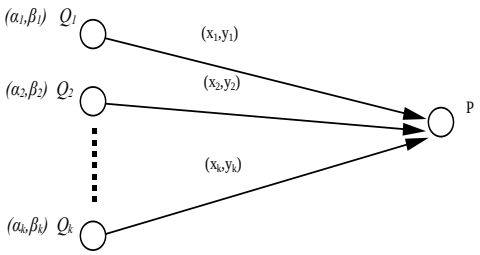
Sonar has been used for submarine and mine detection, depth detection, commercial fishing, diving safety and communication at sea. It is a system that uses transmitted and reflected underwater sound waves to detect and locate submerged objects or measure the distances underwater, qualities that fully explicate its name (i.e. an acronym for SOund, NAvigation and Ranging). There are two major kinds of sonar, active and passive. Active sonar creates a pulse of sound, often called a "ping", and then listens for reflections of the pulse. The pulse may be at constant frequency or a chirp of changing frequency. If a chirp, the receiver correlates the frequency of the reflections to the known chirp. The resultant processing gain allows the receiver to derive the same information as if a much shorter pulse of the same total power were emitted. Various intelligent techniques have been developed during the past in order to exploit the sonar data [7,24,25], and computational intelligence (CI) is among them. Although CI has nowadays substituted traditional artificial intelligence in major applications, for a number of high-level decision tasks common expert systems remain still applicable. The reason can be found into the need for symbolic representation of the knowledge into these systems, which is a feature in which many CI systems fail to succeed. In other words, it is considered that symbolic representation can be of significant value in these systems for humans, by making clear the inference process to users. Among CI methodologies, neural networks are powerful connectionist systems that still lack the element of complete and accurate interpretation into human-understandable form of knowledge and remain a black box for experts. To deal with this situation, a number of alternative approaches have been proposed. Neural logic networks [19] belong to this category, and by their definition can be interpreted into a number of logical or Prolog rules that consist an expert system. Virtually every logic rule can be represented into these networks and then transformed into Prolog commands. Although this model offers excellent results when used within the AI framework (i.e. building a system in a top-down process), the application of neural logic networks in CI's data mining tasks – considered a bottom-up procedure- has undergone limited success. The reason lies in that proposed systems suffered at least one of the following limitations: (a) The extracted neural logic network cannot be interpreted into expert rules [19]-[18]. (b)The proposed methodology cannot express neural logic networks in their generic graph form [5]. (c) The user has to select topology and network connection model [19]-[18]. The application of neural logic networks into adaptive tasks seems promising: the extracted model will preserve its interpretability into a number of expert rules and there is not needed any knowledge-acquiring step. Moreover, a solution obtained this way, leads to potential knowledge extraction. Recently, a new system, namely the evolutionary neural logic networks (ENLN), has been proposed [20] that fulfils those requirements. The new approach uses grammar-guided genetic programming to produce neural logic networks. The evolved solutions can be arbitrarily large and connected networks, since an indirect encoding is adopted. Also, neural logic networks produced by this methodology can always be interpreted into human-understandable expert rules, thus leading to potential knowledge extraction. Our aim in this paper is to demonstrate the effectiveness of the methodology of evolutionary neural networks into real-world problems. The paper is organized as follows. Next section describes the theoretical background, presenting the neural logic networks concept and the grammar guided genetic programming. Following this section, we deal with the design and the implementation of the ENLN system. Next, the results and a following discussion are presented. The paper ends with our conclusion and a description of future work.

2 BACKGROUND

2.1 Active Sonar

In Sea-Water environments, propagation to the target, reflection off

[1] University of the Aegean, Dept. of Finance and Management Eng., Greece, email: tsakonas@stt.aegean.gr
[2] University of the Aegean, Dept. of Finance and Management Eng., Greece, email: g.dounias@aegean.gr
[3] University of the Aegean, Dept. of Shipping, Trade and Transport, Greece, email: nnik@aegean.gr



$$(a,b)=\begin{cases}(1,0) & if \sum_{j=1}^{k} a_j x_j - \sum_{j=1}^{k} b_j y_j \geq 1 \\ (0,1) & if \sum_{j=1}^{k} a_j x_j - \sum_{j=1}^{k} b_j y_j \leq -1 \\ (0,0) & otherwise \end{cases}$$

Figure 1. Example of a neural logic network and its output.

the target, and propagation to the receiver spread active sonar transmit signal in time and frequency. Traditionally, detection and subsequent range estimation has been performed by thresholding a normalized matched filter output for each of several beams pointing in directions of interest. This is only justifiable as a generalized likelihood ratio test when the received echo is simply a time-shifted scaled version of the transmitted waveform plus white noise, obviously not a realistic scenario in the shallow-water active problem. The primary objective of the detector is to determine if there is a target echo present in the received time series. Subsidiary to detection is the estimation of the starting and stopping times of the echo to be used for subsequent signal processing such as more accurate range and bearing estimation, depth estimation, or classification. Traditionally, signal segmentation is performed by clustering matched filter threshold crossings. Without exact knowledge of the environment and a priori information on the target location and reflection properties, the starting time, duration, and shape of the received echo are unknown, thus hindering design of an optimal receiver. It is, however, desirable to exploit available environmental information to the extent that it can feasibly improve detection performance. Were the optimal detector implementable, it would coherently combine the standard matched filter output according to the multipath structure and the target reflection properties. Few literature papers exist in applications with active sonar. In [25], a procedure of mapping unknown obstacles using active sonar is presented. An active sonar imaging and classification system is described in [2] were three neural network architectures were used as classifiers. In [6], the Probabilistic Multi-Hypothesis Tracking (PMHT) algorithm proposed by Streit and Luginbuhl in 1995 is adapted for use in active sonar applications. In [3], a remote, aerial, laser-based sonar method for detecting and locating underwater targets from the air is discussed.

2.2 Neural logic networks

The neural logic network is a finite directed graph. It is usually consisted by a set of input nodes and an output node. In its 3-valued form, the possible value for a node can be one of three ordered pair activation values (1,0) for “true”, (0,1) for “false” and (0,0) for “don't know”. Every synapse (edge) is assigned also an ordered pair weight (x,y) where x and y are real numbers. An example neural logic and its output value (a,b) of node P is shown in Fig. 1. Different sets of weights enable the representation of different logical operations. It is actually possible to map any rule of conventional knowledge into a neural logic network. Neural logic networks can be expanded into fuzzy neural logic networks, enabling this way the handling of real valued attributes [19]. The interpretation of the network into Prolog rules is straightforward. Even though powerful in their definition, neural logic networks are not widely applied. The main reason can be located in the fact that for the known training methodologies [19]-[18], the refinement of the edge weights reduces significantly the interpretability of these networks to expert rules, thus depriving these networks from their valuable feature. Some steps for the preservation of the interpretability have been performed by [1], without however the ability to express arbitrarily large and connected neural logic networks. In this direction, the definition and use of the *neulonet* is demonstrated in [4], still however producing tree-like rule programs.

2.3 Grammar Guided Genetic Programming

The ability to construct functional trees of variable length is a major advantage of genetic programming over genetic algorithms. This property enables the search for very complex solutions that are usually in the form of a mathematical formula - an approach that is commonly known as symbolic regression. Later paradigms extended this concept to calculate any boolean or programming expression. Consequently, complex intelligent structures, such as fuzzy rule-based systems or decision trees have already been used as the desirable target solution in genetic programming approaches [1], [22], [23], [24]. The main qualification of this solving procedure is that the feature selection, and the system configuration, derive in the searching process and do not require any human involvement. Moreover, genetic programming, by inheriting the genetic algorithms' stochastic search properties, does not use local search -rather uses the hyper plane search-, and so avoids driving the solution to any local minimum. The potential gain of an automated feature selection and system configuration is obvious; no prior knowledge is required and, furthermore, not any human expertise is needed to construct an intelligent system. Nevertheless, the task of implementing complex intelligent structures into genetic programming functional sets in not rather straightforward. The function set that composes an intelligent system retains a specific hierarchy that must be traced in the GP tree permissible structures. This writing offers two advantages. First, the search process avoids candidate solutions that are meaningless or, at least, obscure. Second, the search space is reduced significantly among only valid solutions. Thus, a genotype - a point in the search space- corresponds always to a phenotype - a point in the solution space. This approach -known as legal search space handling method [28]- is applied in this work using context-free grammars. As we will discuss in the next paragraph, the implementation of constraints using a grammar can be the most natural way to express a family of allowable architectures. While each intelligent system -such as a neural logic network- has a functional equivalent -by means of being composed by smaller, elementary functions-, what defines and distinguishes this system is its grammar.

2.4 Context-free grammars

The genetic programming procedure may be proved greedy in computational and time resources. Consequently, when the syntax

```
<PROG>   : =  PROG <PLACE1><SYNAPSE>
<PLACE1> : =  S1 <PLACE1><SYNAPSE><PLACE2>
              | P1 <PLACE1><PLACE1>
              | IN
IN       : =  Data attribute (system input)
<PLACE2> : =  S2 <PLACE2><SYNAPSE><PLACE2>
              | P2 <PLACE2><SYNAPSE><PLACE2>
              | E
E        : =  ∅
<SYNAPSE>     : =  LNK <NUM><CUT><SYNAPSE>
              | CNR <CNRSEL><K>
<NUM>    : =  NUM
<CUT>    : =  CUT
<CNRSEL> : =  CNRSEL
<K>      : =  K
NUM      : =  Integer in [1,256]
CUT      : =  Integer in [0,1]
CNRSEL   : =  Integer in [0,10]
K        : =  Integer in [0,9]
```

Figure 2. Context free grammar for the production of neural logic networks within the genetic programming framework.

form of the desired solution is already known, it is useful to restrain the genetic programming from searching solutions with different syntax forms [8]-[14]. The most advantageous method to implement such restrictions among other approaches [15], is to apply syntax constraints to genetic programming trees, usually with the help of a context-free grammar declared in the Backus-Naur-Form (BNF) [14]. The BNF-grammar consists of terminal nodes and non-terminal nodes and is represented by the set {N,T,P,S} where N is the set of non-terminals, T is the set of terminals, P is the set of production rules and S is a member of N corresponding to the starting symbol. The use of the terms terminal and non-terminal in a BNF-grammar, does not correspond to what is usually referred in genetic programming as terminal and function [14]. Rather, a function -a non-terminal node in terms of the GP tree architecture- is expressed as terminal in a BNF grammar.

2.5 Cellular Encoding

Although mapping decision trees or fuzzy rule-based systems to specific grammars can be relatively easy to implement, the execution of massively parallel processing intelligent systems -such as the neural logic networks- is not forthright. In order to explore variable sized solutions, we applied indirect encoding. The most common one is the cellular encoding [9], in which a genotype can be realized as a descriptive phenotype for the desired solution. More specifically, within such a function set, there are elementary functions that modify the system architecture together with functions that calculate tuning variables. Current implementations include encoding for feed forward and Kohonen neural networks [16], [21] and fuzzy Petri-nets [27], [21]. In his original work, Gruau also used a context-free grammar - a BNF grammar- to encode indirectly the neural networks. On the other hand, in [27] a logic grammar - a context-sensitive one- is adapted to encode fuzzy Petri-nets. In our work, we show that as long as the depth-first execution of the program nodes of a GP tree is ensured -which is the default-, a context-free grammar such as a BNF grammar is adequate for expressing neural networks. Gruau's original work has been facing some scepticism [11] on the ability to express arbitrarily connected networks. Later developments [8] seem to offer less restrictive grammar, though the cut function in those implementations still maintained bounded effect. A similar technology, called edge encoding, developed by [12] is also today used with human competitive results in a wide area of applications.

Table I. Operations for Function CNR

Parameter	Calculation
0	Conjunction
1	Disjunction
2	Priority
3	At least k-true
4	At least k-false
5	Majority influence
6	Majority influence of k
7	2/3 Majority
8	Unanimity
9	If-Then operation, Kleene's model
10	Difference

3 DESIGN AND IMPLEMENTATION

The data is normalized to the system's acceptable data range and the procedure creates the evolutionary neural logic network, which is then tested on unknown data. The resulted network is stored and the rules extracted can be used without the need of a computer.

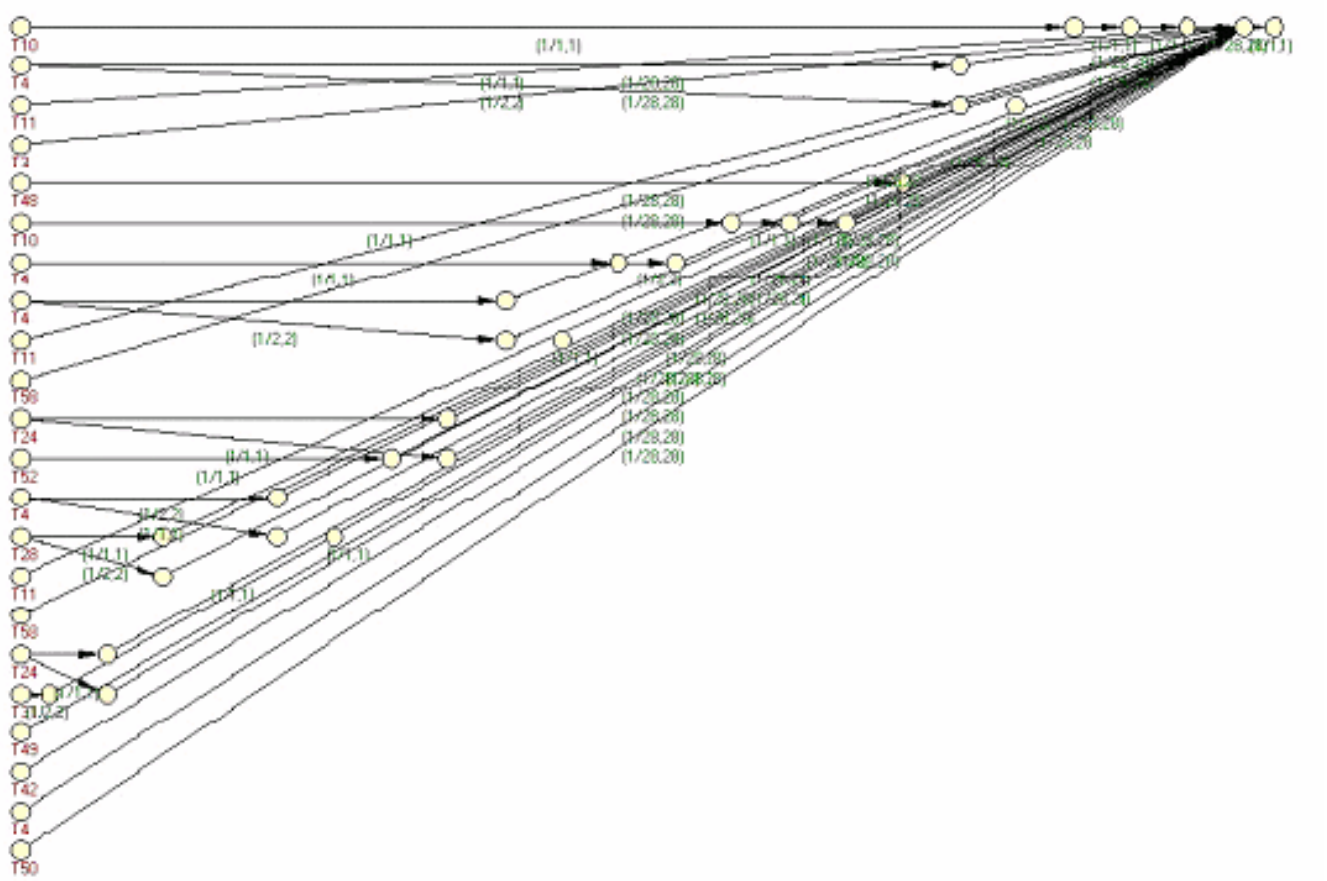
3.1 Data Pre-processing and Genetic Programming Setup

The Sonar data set is consisted of sixty real-valued attributes between 0.0 and 1.0 used to define 208 mines and rocks. Attributes are obtained by bouncing sonar signals of a metal cylinder (or rock) at various angles and rises in frequency. The value of the attribute represents the amount of energy within a particular frequency band, integrated over a certain period of time. It is accepted that the genetic programming procedure may suffer size problems during initialisation [16]. Although the fine-tuning of our algorithm was not the main concern of this paper, we investigated various initialisation approaches. Without claiming optimality, the GP parameters are adapted by [20]. This setup, together with function selection probability optimisation, offered for the presented grammar stable and effective runs throughout experiments. Although the initialisation of the population is random, using this probability bias the algorithm is forced to generate individuals of acceptable size. This optimisation was decided after experimentation, since it is not possible to obtain a general principle regarding the most proper probability values for every case. As it can be observed in [20], the setup denotes our preference for significantly high mutation rates, especially shrink mutation [17] that slows down the code bloat caused by crossover operations.

3.2 System Grammar and Operating Functions

The proposed system grammar is shown in Figure 2. Initial symbol (root) of a genetic programming tree can be a node of type <PROG>. The function set is as follows:

- Function PROG: The function PROG creates the embryonic network that is used later by the functions S1, S2, P1 and P2 to be expanded. An alternative name for this function, which is used throughout this paper, is the term "CNLN".
- Function S1: The function S1 enters a node in serial to the node that is applied, and is applied to input nodes.
- Function P1:The function P1 enters a node in parallel to



(CNLN (P1 (P1 (P1 (S1 (S1 (In T10) (Rule 0 0) E) (Rule 0 0) (S2 E (Rule 0 0) E)) (P1 (S1 (In T4) (Rule 0 0) (P2 E (Rule 10 3) (S2 E (Rule 10 3) E))) (In T11))) (P1 (P1 (In T3) (S1 (In T48) (Rule 12 8) E)) (P1 (P1 (P1 (S1 (S1 (In T10) (Rule 0 0) E) (Rule 0 0) (S2 E (Rule 0 0) E)) (P1 (S1 (S1 (In T4) (Link 50 0 (Rule 0 0)) E) (Rule 10 3) E) (P1 (S1 (In T4) (Rule 12 8) (P2 E (Rule 10 3) (S2 E (Rule 0 0) E))) (In T11)))) (P1 (P1 (In T58) (S1 (In T24) (Rule 12 8) (P2 E (Link 133 0 (Rule 0 0)) E))) (P1 (P1 (S1 (In T52) (Rule 0 0) E) (P1 (S1 (In T4) (Link 133 0 (Rule 10 3)) (P2 E (Rule 0 0) (S2 E (Rule 0 0) E))) (P1 (S1 (In T28) (Rule 0 0) (P2 E (Rule 10 3) (S2 E (Rule 0 0) E))) (In T11)))) (P1 (P1 (In T58) (S1 (In T24) (Rule 12 8) (P2 E (Link 133 0 (Rule 0 0)) E))) (P1 (P1 (S1 (In T31) (Rule 10 3) E) (In T49)) (In T42)))))) (In T4)))) (In T50)) (Rule 2 8))

Figure 3. Extracted solution and the corresponding neural logic network for the Active Sonar classification problem.

the node that is applied, and is also applied to input nodes.

- Function S2: The function S2 enters a node in serial to the node that is applied, and is used for hidden layer nodes.
- Function P2: The function P2 enters a node in parallel to the node that is applied, and is also used for hidden layer nodes. This mechanism, consisting of two different sets of expanding functions (P1 and S1 vs. P2 and S2), is used to ensure that population individuals will include at least one input node.
- Function IN: The operation of function IN is to assign a variable to the input node that it is applied.
- Function E: The operation of function E is to mark the end of the expansion of the network.
- Function LNK: This function provides the framework for the application of cut function. It actually enables the non-full connectivity of the network, a feature that offers larger solution search space.
- Function CNR: This function performs the node inference. Based on the first parameter, the corresponding calculation is performed. The second parameter assists the calculation for the at-least-k and majority-of-k operators. Possible computations are shown in Table I. An alternative name for this function, which is used throughout this paper, is the term “Rule”. In order to be able to process values other than true, false and don’t know, we applied the fuzzy propagation algorithm [19], which allows us to process any real valued variables (using proper normalization).
- Function NUM: The function NUM returns an integer in the interval [1,256] to be used by the calling LNK function.
- Function CUT: The function CUT returns an integer in the interval [0,1] to be used by the calling LNK function. If the returned value is 1, then the link will be ignored in the calculations (considered "cut").
- Function CNRSEL: The function CNRSEL returns an integer in the interval [0,8] to be used by the calling CNR function as its first parameter.
- Function K: The function K returns an integer in the interval [1,256] to be used by the calling CNR function if the returned value of the CNRSEL is 3,4 or 6 (corresponding to the calculation of the at least k-true, at least k-false and majority of k functions).

Having discussed the system design, in the following session we shall apply the methodology in the active sonar target identification domain.

4 RESULTS AND DISCUSSION

The remote detection of undersea mines in shallow waters using active sonar is a crucial capability required to maintain the security of important harbours ands coastline areas. It is often very difficult to distinguish active sonar returns from mines and return from clutter on the sea floor. There is currently no reliable signal classification scheme for automatically interpreting such sonar returns. Instead highly trained sonar operators must be relied upon to identify the presence of a mine. The present study was conducted to explore the use of neural networks as a means of automating mine-hunting operations. More specifically, the system develops a decision whether the signal corresponds to a cylinder (mine) or to a rock [7]. The database is consisted of two parts. The first part contains 111 records which are acquired by returning sonar signals from a metal cylinder in various positions. The second part is comprised by 97 records which correspond to sonar signals returned by rocks in similar situations. Training and testing data records were randomly selected by these sets. Also, in order to avoid overfitting during the training phase, we used a validation set. According to the literature, the target is to develop a system with high accuracy and potential knowledge interpretation. The evolved neural logic network and its graphical representation are depicted in Figure 3. This solution achieves a 86.27% (44/51) accuracy in unknown data. The accuracy in the training data was 88.24% (90/102) and in the validation data set it was 80.39% (41/51). In the literature, the derived accuracy for the various systems applied in the same data set ranges from 73.1% to 89.2%. Other experiments in [3] using neural logic networks by means of genetic programming offer for the same data set an equivalent classification score (86%) when *neulonet* association rules are applied, and a lower score (78.3%) when conjunctive association rules are used, a direct score comparison not being applicable however, since different training and test data sets have been used. The extracted neural logic network, due to the nature of the problem, maintains significant complexity, yet it achieves competitive to the literature results. It is worth to note also, that our solution can still be interpreted into a number of logical or Prolog rules, although solution interpretation was not among our primary targets for the specific problem.

4 CONCLUSION AND FURTHER RESEARCH

The aim of this paper was to demonstrate the effectiveness of the evolutionary neural logic networks paradigm into real-world problems, such as the active sonar target identification. In general, neural networks are powerful connectionist systems that have been introduced in areas where symbolic processing systems of traditional artificial intelligence used to be applied. As a tool of computational intelligence, the adaptation of the neural network to the problem domain using an inductive method, offers advantage over expert systems where the knowledge must be acquired first, before the system development. Ever since their first application, interpretation of the obtained knowledge was a research target for neural networks. In the scope of this area, the neural logic networks have been proposed as a class of networks that by their definition preserve their interpretability into symbolic knowledge.

Until recently however, the application of an effective training / production method within the CI framework has not been successful. A novel system that uses genetic programming with indirect encoding that has been proposed recently [20], overcomes these problems, producing automatically designed and tuned neural logic networks, which always preserve their interpretability. In this work we applied the system into a real-world problem, the Active Sonar classification problem. The system has been proved capable of producing competitive to the literature results. The acquired solution although being in a complicated form, it still maintains its interpretability. The complexity of the solution is rather straightforwardly related to the nature of the problem, i.e. physical measurements. However, as obviously seen, the solution interpretation could not be among the primary targets of this research, rather than the high classification rate. Hence, according to the experts, the application to a sonar classification problem shows that under particular circumstances the system can be implemented to some degree into this real situation problem.

Future work involves the application of the system in other sonar data sets, as well as in other areas, and the incorporation of recursive structures into the neural logic network architecture. Moreover, the *minimum description length* principle will be developed to be included as an anti-overfitting measure into the active sonar target identification problem. Finally, we believe that parameter-tuning optimisation of the underlying genetic programming algorithm will offer better efficiency; hence this will be of primary importance among our future work.

REFERENCES

[1] Alba E., Cotta C. and Troya J.M., "Evolutionary Design of Fuzzy Logic Controllers Using Strongly-Typed GP", Proc. 1996 IEEE Int'l Symposium on Intelligent Control, pp. 127-132. New York, NY., 1996.

[2] L. L. Burton and H. Lai. Active sonar target imaging and classication system. In Proceeedings of the SPIE International Symposium on Aerospace/Defence Sensing and Control, pages 19-33, Orlando, FL, April 1997.

[3] Chia H W-K and C-L Tan, Confidence and support classification using genetically programmed neural logic networks, Genetic and Evolutionary Computation Conference, GECCO 2004, 26-30 June 2004, Seattle, Washington, USA

[4] Chia H W-K and C-L Tan, Association-based evolution of comprehensive neural logic networks, GECCO 2004, 26-30 June 2004, Seattle, Washington, USA.

[5] Chia H.W-K. and Tan C-L.,"Neural logic network learning using genetic programming", Intl. Journal of Comp. Intelligence and Applications, 1:4, 2001, pp 357-368.

[6] Christian G. Hempel and Sheri L. Doran, A PMHT algorithm for active sonar, Proc. SPIE 5430, 132 (2004)

[7] Gorman, R. P., and Sejnowski, T. J. (1988). "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75-89.

[8] Gruau F., Whitley D. and Pyeatt L., "A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks", in Koza J.R., Goldberg D.E., Fogel D.B., Riolo R.L., Eds.,, Genetic Programming 1996: Proceedings of the First Annual Conf., pp 81-89, Cambridge, MA, MIT Press, 1996.

[9] Gruau F., "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm", Ph.D. Thesis, Ecole Normale Superieure de Lyon, anonymous ftp:lip.ens-lyon.fr (140.77.1.11) pub/Rapports/ PhD PhD94-01-E.ps.Z.

[10] Gruau F., "On Using Syntactic Constraints with Genetic Programming", in P.J.Angeline, K.E.Jinnear,Jr., Eds., Advances in Genetic Programming, MIT,1996.

[11] Hussain T., "Cellular Encoding: Review and Critique", Technical Report, Queen's University, 1997, http://www.qucis.queensuca/ home/hussain/web/ 1997_cellular_encoding_review.ps.gz

[12] J.Koza, F. Bennett, D. Andre and M. Keane, Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis, Morgan Kaufmann, 2003.

[13] Montana D.J., "Strongly Typed Genetic Programming", Evolutionary Computation, vol. .3, no. 2, 1995.

[14] Naur P., "Revised report on the algorithmic language ALGOL 60", Commun. ACM, Vol 6, No 1, pp 1-17, Jan 1963.

[15] N.Paterson and M.Livesey,"Evolving Caching Algorithms in C by

GP", Genetic Programming 1997, pp 262-267, MIT Press, 1997.
[16] Ratle A. and Sebag M., "Genetic Programming and Domain Knowledge: Beyond the Limitations of Grammar-Guided Machine Discovery", Schienauer et al., Eds., Proc. of the 6th Conf. on Parallel Problems Solving from Nature, LNCS, Springer, Berlin, 2000, pp 211-220
[17] Singleton A., "Genetic Programming with C++", BYTE Magazine, Feb 1994.
[18] Tan A-H. and Teow L-N., "Inductive neural logic network and the SCM algorithm", Neurocomputing, Vol. 14, 2 : 5, pp.157-176, 1997.
[19] Teh H.H., Neural Logic Networks: A New Class of Neural Networks, World Scientific Pub Co, 1995.
[20] Tsakonas A., Aggelis V., Karkazis I. and Dounias G., "An Evolutionary System for Neural Logic Networks using Genetic Programming and Indirect Encoding", Journal of Applied Logic, Special Issue on Neural-Symbolic Systems, Vol.2 (3), September 2004, pp.349-379, Elsevier.
[21] Tsakonas A. and Dounias G., Decision Making in the Medical Domain: Comparing the Effectiveness of GP-Generated Fuzzy Intelligent Structures, Proc. of Eunite-03, Oulou, Finland, 2003.
[22] Tsakonas A., Dounias G., "Hierarchical Classification Trees Using Type-Constrained Genetic Programming", Proc. of 1st Intl. IEEE Symposium in Intelligent Systems, Varna, Bulgaria, 2002.
[23] Tsakonas A., Dounias G., Axer H., and von Keyserlingk D.G., "Data Classification using Fuzzy Rule-Based Systems represented as Genetic Programming Type-Constrained Trees", Proc. of the UKCI-01, Edinbourgh, UK, pp 162-168, 2001.
[24] Tsakonas A. and Dounias G., "A Scheme for the Evolution of Feedforward Neural Networks using BNF-Grammar Driven Genetic Programming", Proc. of Eunite-02, Algarve, Portugal, 2002.
[25] F. Wallner, R. Graf, and R. Dillmann. Real-time map refinement by fusing sonar and active stereo-vision. In IEEE International Conference on Robotics and Automation, Nagoya, Japan, 1995.
[26] Whigham P., "Search Bias, Language Bias and Genetic Programming", Genetic Programming 1996, pp 230-237, MIT Press, 1996
[27] Wong M.L., "A flexible knowledge discovery system using genetic programming and logic grammars", Decision Support Systems, 31, 2001, pp 405-428.
[28] Yu T. and Bentley P., "Methods to Evolve Legal Phenotypes", Lecture Notes in Comp. Science 1498, Proc. of. Parallel Problem Solving from Nature V, pp 280-291,1998.