

Evolving Weights and Transfer Functions in Feed Forward Neural Networks

M. Annunziato¹, I. Bertini¹, M. Lucchetti², S. Pizzuti^{1,3}

¹ENEA – Energy, New technologies, Environment Agency
'Casaccia' R.C., Via Anguillarese 301, 00060 Rome Italy

Phone: +39-0630484411, Fax: +39-0630484811

email: {mauro.annunziato, ilaria.bertini, stefano.pizzuti}@casaccia.enea.it

²University of Rome "La Sapienza"

Dept. of Computer and Systems Science, Via Eudossiana 18, 00184 Rome Italy

Phone: +39-06-44585938, Fax: +39-06-44585367

email: lucchetti@dis.uniroma1.it

³CS - Communication Systems s.r.l.

Piazza della Repubblica 32, Milan Italy

ABSTRACT: In this paper we show different evolutionary algorithms applied to the simultaneous off-line evolution of weights and transfer functions of feed-forward neural networks. Experimentation has been carried out with classical benchmarks when weights and both weights and transfer function are evolved and a comparison of the proposed evolutionary methods with classical methodologies (the back-propagation algorithm) are shown. Results are very promising and show the effectiveness of the addressed evolutionary methodologies to solve the problem of simultaneously finding the optimal weights and transfer functions of a neural network.

KEYWORDS: Evolutionary Neural Networks, Evolutionary Algorithms, Feed Forward Neural Networks, Smart Adaptive Systems

1. INTRODUCTION

Considerable research on the off-line evolution of Artificial Neural Networks (ANN) using Evolutionary Algorithms (EAs) has been carried out in recent years giving rise to a new branch of ANN known as Evolutionary Neural Networks [1]. In this context most of the research has concentrated on the evolution of weights and topological structures but relatively little has been done on the evolution of node transfer functions and the simultaneous evolution of both weights and node transfer functions.

Often the transfer function of each node in the architecture has been assumed to be fixed and predefined by human experts but the transfer function has been shown to be an important part of an ANN architecture which has significant impact on ANN's performance. Moreover the transfer function is often assumed to be the same for all the nodes in an ANN, at least for all the nodes in the same layer. In this way the application of EAs to the evolution of node transfer functions to get the optimal network's architecture seems to be a promising research field.

Stork et al. [2] were, to our best knowledge, the first to apply EAs to the evolution of both topological structures and node transfer functions even though only simple ANNs with seven nodes were considered. The transfer function was specified in the structural genes in their genotypic representation. It was much more complex than the usual sigmoid function because they tried to model a biological neuron in the tailflip circuitry of crayfish.

White and Ligomenides [3] adopted a simpler approach to the evolution of both topological structures and node transfer functions. For each individual (i.e., ANN) in the initial population, 80% nodes in the ANN used the sigmoid transfer function and 20% nodes used the Gaussian transfer function. The evolution was used to decide the optimal mixture between these two transfer functions automatically. The sigmoid and Gaussian transfer function themselves were not evolvable. No parameters of the two functions were evolved.

Liu and Yao [4] used EP to evolve ANNs with both sigmoidal and Gaussian nodes. Rather than fixing the total number of nodes and evolve mixture of different nodes, their algorithm allowed growth and shrinking of the whole ANN by adding or deleting a node (either sigmoidal or Gaussian). The type of node added or deleted was determined at random. Hwang et al. [5] went one step further. They evolved ANN topology, node transfer function, as well as connection weights for projection neural networks.

Sebald and Chellapilla [6] used the evolution of node transfer function as an example to show the importance of evolving representations. Representation and search are the two key issues in problem-solving. Co-evolving solutions and their representations may be an effective way to tackle some difficult problems where little human expertise is available.

The goal of this paper is to show that in general Evolutionary Computation (EC) is a good approach to solve the problem we are facing here.

2. THE EVOLUTIONARY ALGORITHMS

We conducted tests with three different types of evolutionary algorithms we carried out. They are all inspired by the artificial life metaphor and their basic principle consists in leaving the system a certain degree of freedom in order to develop an emergent behaviour by combining genetics with other peculiar aspects of life (interaction, competition, cooperation, food quest, etc.). In the proposed algorithms each individual represents a feed forward neural network in competition with the others by means of the proper fitness, which depends on the capability of reconstructing the training database. The genotype is composed by the synaptic weights and the transfer functions and reproduction operators simultaneously evolve them.

The fitness of the individuals is measured referring to the global error in modelling the training database with the following formula:

$$\text{Fitness} = 1 - \text{RMSE} \quad (1)$$

Where RMSE is the classical Root Mean Squared Error normalised in the lattice [0,1] used by the back-propagation (BP) algorithm. This cost function has been chosen in order to directly compare the results with the ones obtained with BP methodology.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_1^n (0.5 * \sum_1^m (y - y_t)^2)} \quad (2)$$

Where n is the dimension of the training data set, m is the number of output neurons, y is the estimated output and y_t is the corresponding target.

All the inputs and outputs of the networks are normalized between 0 and 1 (as well as the targets) and there are no differences between the activation functions for input, hidden and output nodes.

As a measure of the level reached in the training we take the fitness of the best individual corresponding to the best performing neural network.

During evolution we allow each neuron to have its own activation function. The activation functions we considered are :

$$y = 1/(1+e^{-x}) \quad (3)$$

$$y = \tanh(x) \quad (4)$$

$$y = \sin(x) \quad (5)$$

$$y = (1+\tanh(x))/2 \quad (6)$$

$$y = (1+\sin(x))/2 \quad (7)$$

$$y = x \quad (8)$$

$$y = 0.5 + \arctg(x)/\pi \quad (9)$$

$$\text{if } (\tanh(x) < 0.0) \text{ y} = 0.0 \text{ else } \text{y} = \tanh(x) \quad (10)$$

$$\text{if } (\sin(x) < 0.0) \text{ y} = 0 \text{ else } \text{y} = \sin(x) \quad (11)$$

$$\text{if } (x < 0.0) \text{ then } \text{y} = 0.0 \text{ else if } (x > 1.0) \text{ then } \text{y} = 1.0 \text{ else } \text{y} = x \quad (12)$$

$$\text{if } (x \leq 0.0) \text{ then } \text{y} = 0.0 \text{ else } \text{y} = 1.0 \quad (13)$$

The algorithms we developed, which are going to be described in the next sub-paragraphs, are Partial Emulation (PE), Artificial Societies (AS) and Chaotic Populations (CP).

2.2. CHAOTIC POPULATIONS

This algorithm is inspired by the fact that it is known that in natural environments population sizes, reproduction and competition rates, change and tend to stabilise around appropriate values according to some environmental factors. In this way it has been carried out a technique for setting the genetic parameters during the course of a run by *adapting* the population size and the operators rates on the basis of the environmental constrain of maximum population size. The main features of the algorithm are : meeting, mono-sexual and bi-sexual reproduction, and competition.

In this evolutionary algorithm the roulette wheel selection is replaced with the *meeting* concept. At each iteration we pick the i^{th} individual of the population, for i from 1 to the current population size, up and then we randomly look for a second individual. Therefore the meeting probability is defined as the population density. In this stage if someone is met then interaction (bi-sexual reproduction or competition) will start, else mono-sexual reproduction of the current individual might occur.

Bi-sexual reproduction is performed according to an adaptive rate and if it occurred then the resulting sons would not replace their parents, they would simply be added to the population. In this situation population increases of two new elements.

Mono-sexual reproduction is performed according to an adaptive reproduction rate and when it occurs an individual first clones itself and then mutates. The mutated individual doesn't replace the original one, it is simply added to the population and the population size increases of one unit.

Competition starts according to an adaptive rate and it means that when two individuals meet and they do not mate through bi-sexual reproduction then they'll fight for survival, the stronger will kill the weaker and this one will be kicked from the population off.

The resulting population dynamics are chaotic since the algorithm is a particular instance of a chaotic map similar to the well known logistic map. The reader interested in further details can refer to [7].

With respect to the original formulation two slight *ad hoc* modifications, mutation and freezing, have been introduced in order to improve the algorithm performance. The first one consists in trying a slight mutation on every individual and if it improves the individual's fitness then that mutation will be hold. This trick has shown to improve performance and accuracy but, as drawback, the fitness evaluations increase. The second one acts on the principle that when the optimal transfer function setting has been achieved it is no more useful to keep evolving transfer functions because it in that situation a lot of noise is added. In this way when the best individual reaches a certain very high fitness, typically around 0.95, then its architecture is considered frozen, that configuration is imposed to the whole population and it is not evolved anymore. Experimentation showed that with trick the optima training is achieved considerably faster than always keep evolving the transfer functions.

2.3. PARTIAL EMULATION AND ARTIFICIAL SOCIETIES

In this paragraph we illustrate the common features of the PE and AS algorithms and we'll describe the details of the two in the following paragraphs. In these algorithms, many individuals or *agents* move in a two-dimensional space. An individual is a point moving over the 2D lattice. The space is divided in cells and one cell can be occupied by only one individual. When an agent tries to enter into an occupied cell an interaction occurs. The type of interaction depends on the algorithm; the next paragraphs will explain these aspects. All the parameters for the dynamics, reproduction, life, interaction are recorded in a genetic map that is defined at the birth of an individual and remain unchanged throughout the individual's life.

Every life cycle all the individuals are moved into one of the surrounding cells. The movement of an individual is composed by a deterministic component and a random component. The reciprocal importance of the two components is regulated by a parameter, called *irrationality*. High values for this parameter cause totally random movement; low values cause totally deterministic movement. The dynamics of the individuals and the probability of interaction depends on several variables like number of agents, modalities of interaction, size of the discretization of the space (90x60). The space is toroidal: when an individual exits from a side it re-enters from the opposite side of the space. The parameters related to the dynamics and the other probabilistic models are affected by the random sequence that is identified by its *random initial seed*. The short term evolution of the population can be influenced by the initial seed, but after a while, the results are substantially independent on this choice.

In the following paragraph we explain the specific features of the two algorithms illustrating the meaning of the main parameters and indicating in parenthesis the most used values for each parameter.

2.3.1. Artificial Societies details

This algorithm is inspired to a society with a strong competition where the optimisation is obtained through the evolution mainly based on the genetic mutations during reproduction.

At the beginning, a number of individuals (256) are placed in the space and they begin reproducing and developing a population. A probabilistic test for self (haploid) reproduction is performed at every life cycle (probability=0.5). In the

haploid reproduction, a probabilistic-random mutation occurs in the weights in relation to a *mutation average* rate (1/10): only few weights are mutated in a mutation application. Another parameter is the mutation maximum percentage intensity: the mutated weights can be amplified or decreased by a delta corresponding in a percentage between 0 and the maximum percentage intensity (20 %). The mutation is also applied with a low probability (1/10) to the change of the type of activation function.

Reproduction is limited by a maximum number of individuals in the space. In order to avoid population saturation this number (1024) is chosen quite higher than the natural fluctuations in the population dynamics (50-500 individuals). Furthermore, the reproduction is conditioned by the presence of a free cell in the space surrounding the father. Upon the meeting of two individuals, a mechanism of competition is activated. The competition is based on the value of fitness of the two individuals: the individual that has a higher fitness survives. After a while, individuals that have higher fitness are able to survive and continue the evolution. In this way the best solution that corresponds to the individual with the highest fitness, increases continuously its presence in the population reaching the optimal values. Due to the hard competition mechanism (fatal interaction) the density of agents in the space is approximately uniform without formation of local groups.

In order to improve the performance of the artificial society we have included a modification in the algorithm that allowed the possibility to develop cycles of development of the biodiversity in the sense of different genomes and cycles of selection. This effect was achieved by including a condition in the reproduction: the son can survive after the birth only if his fitness is higher than the father's one. If this test is not passed the birth has a very low probability to occur (1/10000). The effect of this simple assumption is that the probability of reproduction depends on the fitness of the individual. High fitness means low probability of reproduction. During the evolution the population selects the best individuals by decreasing the probability of new births and consequently decreasing the number of living individuals (*selection period*). When a new born with low fitness escapes to the random roulette (1/10000) and survives to the father and to the quick interaction with the other individuals, he has a very high probability to generate offspring with low fitness and high probability of reproduction. These new individuals give an impulse to the number of living individuals and the biodiversity of the genetic has a quick explosion (period of *development of the biodiversity*). Using this approach, the population exhibits continuous oscillations periodically renewing its content of information.

No cross-over operator is present in this algorithm. Instead the mutation operator is used in two ways: the reproduction and the self-mutation. In the self-mutation mechanism, at every life cycle, the agent try to apply a mutation to its network. If the fitness of the muted network is higher, the mutation is really applied and the muted network substitute the previous one. Otherwise the mutation is rejected.

Finally no mechanism treating agents energy or reinizalitation of the network is considered in this algorithm.

To conclude, the most interesting new feature of this algorithm is the mechanism of the biodiversity cycle. This mechanism warrants a continuous production of new explorations and solutions to select. This corresponds in a strong ability to reach high levels of optimization. The general drawback can be identified in the need to explore a wide number of generations before reaching the maximum level.

2.3.2. Partial Emulation details

This algorithm is inspired by a cooperative-competitive society where optimization is obtained through a communication-emulation mechanism. The metaphors of evolution and genetics are not included because population is fixed and the agents cannot neither die nor reproduce. The optimization consists in a sort of development of the best adaptive behaviour.

At the beginning, a fixed number of individuals (256) are placed in the space and they begin moving in the space. As in the previous case, the individuals have the same rules for dynamics and self-mutation at every life-cycle.

A basic mechanism has been introduced in order to balance the competition effect with a cooperative effect. For this reason we introduced the energy variable. At the beginning any individual is initialised with a bonus of initial energy (500 units). At every life cycle a consumption of energy (2 units) is paid by each agent.

Upon the meeting of two individuals, a mechanism of competition is activated. The competition is based on the value of fitness of the two individuals: the individual with lower fitness transfers a quantum of its energy (1 unit) to the other individual. At the same time a cooperative mechanism is applied: the losing individual applies a partial emulation of the winner neural network. The partial emulation consists in the modification of its weights in relation to the following formula:

$$W_{li} = \alpha * W_{wi} + (1-\alpha) * W_{li} \quad (14)$$

Where W_{wi} is the generic weight (or threshold) of the agent with lower fitness and W_{li} is the generic weight of the agent with higher energy. α is the emulation factor (0.05). The emulation is applied also to the type of activation function with a probability of 1 over the number of nodes. The emulation corresponds to a sort of cross-over operator with an oriented direction.

When an agent loses energy up to a minimal zero level, its neural network is completely reinitialised and a new bonus of initialisation energy is assigned to it. The complete re-initialisation is important in order to have a basin of completely new solutions continuously improving the biodiversity of the environment. The initial bonus of energy is

important to give a time in which the re-initialised agents evolve through the self mutation and partial emulation mechanism. During this time these agents lose energy because of the interaction with well evolved agents. Most of the re-initialised agents come back at zero energy, but a few number of them (*emergent agents*) reach a good fitness level and are able to pass to the class of well evolved agents. The initialisation energy is necessary to partially protect the emergent agents during their initial climbing.

The dynamic of this algorithm is quite interesting and recalls the stock market competition. When an individual gains a high fitness, it increases its incoming of energy competing with the others agents. The other individuals try to emulate and learn from him. When the others reach its level and someone becomes better, the first individual starts having an attenuation of the incoming energy and then a drastic energy reduction up to finish its energy. At this point it is forced to change its behaviour to come back to a positive energy incoming. This form of learning is conceptually very different from the other case because of its feature of collective learning and *volatility*. In fact the produced knowledge is a product of the whole society and it is moved dynamically between the various individuals. Although the knowledge is generated during the life of the individuals, it can be transmitted through the generations. Another interesting aspect of this algorithm is the formation of local groups in the space. This effect is induced by the soft competitive-cooperative interaction. This behaviour allows the creation of niches of evolution of groups of individuals evolving together with high level of reciprocal interaction. After a while the group is dissolved and the generated information is transported in other evolving groups.

To conclude, the most interesting feature of this algorithm is the mechanism of the balance between the competitive and the cooperative behavior. This mechanism enhances the social component of the learning and the formation of niches of evolution. Also a good mechanism to promote new solution and protect emergent solution is included. The general drawback is similar to that one of the Artificial Societies algorithm: the need of a wide number of generations to reach the maximum level.

3. EXPERIMENTATION

Tests have been carried out on databases derived from very well known and widely studied benchmark problems. Table I shows a comparison of the experimental results when the evolution of weights is applied, keeping as fixed transfer functions the classic sigmoid (3), and when the evolution of both weights and transfer functions is applied using as initial condition the one with all sigmoidal functions. In the first case a comparison with the BP algorithm is reported and a direct comparison can be made. In parenthesis the input-hidden-output structure is described. It has to be underlined that all the networks' topologies have been chosen to be highly inadequate to solve the problems with BP and fixed sigmoidal transfer functions because we want to show that the proposed methodologies applied in this work can find good solutions in the situations where traditional techniques fail. Results are calculated according to the RMSE formula (2).

		XOR (2-0-1)	IRIS (4-1-3)	ADDITION (4-3-3)	6 BITS (6-1-1)	PARITY (4-1-1)
Weights	BP	0.38	0.33	0.47	0.36	0.35
	CP	0.35	0.31	0.3	0.33	0.32
	AS	0.35	0.29	0.31	0.33	0.32
	PE	0.35	0.29	0.23	0.33	0.32
Weights+transf.funct.	CP	0.0	0.07	0.0	0.0	0.0
	AS	0.0	0.07	0.0	0.0	0.0
	PE	0.0	0.05	0.0	0.0	0.0

Table I. Experimental results (RMSE)

Tables II and III show the experimental settings of all the methods applied in both cases. Finally table IV shows the best configurations found when evolving transfer functions. The numbers in parenthesis refer to the transfer functions previously numbered.

		XOR (2-0-1)	IRIS (4-1-3)	ADDICTION (4-3-3)	6 BITS (6-1-1)	PARITY (4-1-1)
BP	Cycles	1000000	1000000	1000000	1000000	1000000
	Generations	100	2000	2000	2000	300
CP	Fitness evaluations	11300	740000	256000	227000	35000
	Max population size	100	300	100	100	100
	Generations	100	610	1840	290	110
AS	Fitness evaluations	49200	868000	2165000	375000	155000
	Max population size	1024	1024	1024	1024	1024
	Generations	100	840	15450	740	310
PE	Fitness evaluations	26500	312000	6248000	273000	105000
	Max population size	256	256	256	256	256
	Generations	100	840	15450	740	310

Table II. Experimental settings used for training only weights

		XOR (2-0-1)	IRIS (4-1-3)	ADDICTION (4-3-3)	6 BITS (6-1-1)	PARITY (4-1-1)
CP	Generations	100	2000	3000	100	100
	Fitness evaluations	12000	740000	367000	12500	12500
	Max population size	100	300	100	100	100
AS	Generations	20	141700	740	35	10
	Fitness evaluations	16000	52900000	814000	47500	7700
	Max population size	1024	1024	1024	1024	1024
PE	Generations	22	6500	1650	140	30
	Fitness evaluations	6200	1670000	635000	45000	8000
	Max population size	256	256	256	256	256

Table III. Experimental settings used for evolving both weights and transfer functions

	XOR (2-0-1)	IRIS (4-1-3)	ADDICTION (4-3-3)	6 BITS (6-1-1)	PARITY (4-1-1)
CP	Out=(11)	Hidden=(8) Out=(12-11-12)	Hidden=(13-5-10) Out=(13-13-12)	Hidden=(5) Out=(12)	Hidden=(5) Out=(12)
AS	Out=(11)	Hidden=(8) Out=(12-5-12)	Hidden=(5-5-11) Out=(8-5-12)	Hidden=(11) Out=(8)	Hidden=(5) Out=(10)
PE	Out=(11)	Hidden=(5) Out=(12-5-12)	Hidden=(4-13-8) Out=(13-7-13)	Hidden=(7) Out=(8)	Hidden=(5) Out=(13)

Table IV. Optimal transfer functions

These experimental results let us draw some conclusions. We can see from Table I that EC based techniques perform slightly better than the classical BP algorithm when applied to train the network's weights. However the improvement given by EC techniques in this situation and for this problems is not remarkable and sometimes BP is faster. As result we might say that in this situation EC is not worthy to be applied.

A significant performance enhancement is achieved when EC algorithms are applied to simultaneously evolve both weights and transfer functions. In this situation the prediction error is generally cut from about 0.33-0.47 nearly to zero. This result is extremely remarkable and shows the EC capabilities to simultaneously and effectively train and design neural networks. In this situation EC is an approach which provide useful tools to solve problems which BP alone cannot and which are often left to human expertise.

The next step is to test the performance and the generalization ability of the addressed methods on testing data. Among the proposed problems the only one which has a testing set is the iris one. In the following table a comparison of testing results for this benchmark is proposed.

	BP	CP	AS	PE
Iris (4-1-3)	0.35	0.19	0.21	0.19

Table V. Testing results

The testing results show a remarkable improvement with respect to the BP algorithm. However it is still worth to underline that these results are not the optimal ones for this problem. This because we used only one hidden neuron. Adding the number of hidden neurons the BP is capable to solve the problem, but the goal of the proposed work was to show that the proposed methodologies can find good solutions when BP fails.

4. CONCLUSION

In this paper we showed different EC based algorithms applied to the simultaneous off-line evolution of weights and transfer functions of feed-forward neural networks. With respect to previous and related works what's new in this paper is that for the first time different EC based techniques have been compared and applied to different benchmarks in order to investigate the possibility of simultaneously evolving weights and transfer functions.

Experimentation concerned the evolution of weights alone and of both weights and transfer functions applied to classical benchmarks. In the first case a direct comparison with the classical Back-Propagation algorithm has been proposed. In this situation EC results don't significantly improve those obtained with BP which sometimes is faster. On the other hand results are very promising and much better when simultaneously evolving weights and transfer functions. In this situation errors in the training stage are driven nearly to zero and testing results show a remarkable improvement in the solution found with respect to BP. In general we can state that evolutionary based methods are a good approach to solve the problem of effectively training (weights) and designing (transfer functions) neural networks which Back-Propagation alone cannot. Moreover results stressed the importance of transfer functions in neural networks. It has to be underlined that all the networks' topologies have been chosen to be highly inadequate to solve the problems with BP and fixed sigmoidal transfer functions because we wanted to show that the methodologies proposed in this work find good solutions in the situations where traditional techniques fail.

Future work moves towards four directions. The first and simplest one is that of letting the functions' parameters evolve. The second one is the one which will introduce the evolution of the number and the selection of input and hidden nodes. In this way a complete design will be developed including the selection of the most relevant inputs (feature selection). The third is the one concerning the EC application to fully connected general neural networks (not only feed forward). The last line of research should move towards the on-line application of the proposed methodologies and concepts in order to carry out neural networks capable to dynamically adapt their topology to non stationary environments. This will lead to really adaptive and evolutionary neural networks.

REFERENCES

- [1] Yao X., "Evolving Artificial Neural Networks", *Proceedings of the IEEE*, **87(9)**:1423-1447, September 1999
- [2] Stork D. G., Walker S., Burns M. and Jackson B., "Preadaptation in neural circuits" in Proc. of Int'l Joint Conf. on Neural Networks, Vol. I, (Washington, DC), pp. 202--205, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.
- [3] White D. and Ligomenides P., "GANNet: a genetic algorithm for optimizing topology and weights in neural network design", in Proc. of Int'l Workshop on Artificial Neural Networks (IWANN'93), pp. 322--327, Springer-Verlag, 1993. Lecture Notes in Computer Science, Vol. 686.

- [4] Liu Y. and Yao X., "Evolutionary design of artificial neural networks with different nodes", in Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC'96), Nagoya, Japan, pp. 670--675, IEEE Press, New York, NY 10017-2394, 1996.
- [5] Hwang M. W., Choi J. Y., and Park J., "Evolutionary projection neural networks", in Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, ICEC'97, (Piscataway, NJ, USA), pp. 667--671, IEEE Press, 1997.
- [6] Sebald A. V. and Chellapilla K., "On making problems evolutionarily friendly, Part I: Evolving the most convenient representations", in Evolutionary Programming VII: Proc. of the 7th Annual Conference on Evolutionary Programming (V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, eds.), vol. 1447 of Lecture Notes in Computer Science, (Berlin), pp. 271--280, Springer-Verlag, 1998.
- [7] Annunziato M., Pizzuti S., "Adaptive Parameterisation of Evolutionary Algorithms and Chaotic Population Dynamics", International Journal of Knowledge-Based Intelligent Engineering Systems, 170-177, ISSN: 1327-2314, October 2002