Journal of Information & Knowledge Management, Vol. 5, No. 4 (2006) 303–313

Decision Support Systems Using Ensemble Genetic Programming

Ajith Abraham[*,‡] and Crina Grosan[†,§]
[*]*IITA Professorship Program, School of Computer Science*
Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Republic of Korea
[†]*Department of Computer Science*
Babes-Bolyai University, Cluj-Napoca, Romania
[‡]*ajith.abraham@ieee.org*
[§]*cgrosan@cs.ubbcluj.ro*

Abstract. This paper proposes a decision support system for tactical air combat environment using a combination of unsupervised learning for clustering the data and an ensemble of three well-known genetic programming techniques to classify the different decision regions accurately. The genetic programming techniques used are: Linear Genetic programming (LGP), Multi-Expression Programming (MEP) and Gene Expression Programming (GEP). The clustered data are used as the inputs to the genetic programming algorithms. Some simulation results demonstrating the difference of these techniques are also performed. Test results reveal that the proposed ensemble method performed better than the individual GP approaches and that the method is efficient.

Keywords: Decision support systems; genetic programming; ensemble systems; evolutionary multi-objective optimisation.

1. Introduction

Several decision support systems have been developed mostly in various fields including medical diagnosis, business management, control system, command and control of defense, air traffic control and so on (Leal de Matos and Powell, 2003; Moynihan *et al.*, 2002; Papamichail and French, 2003; Abraham *et al.*, 2005). Usually previous experience or expert knowledge is often used to design decision support systems. Several adaptive learning frameworks for constructing intelligent decision support systems have been proposed (Tran *et al.*, 2001, 2002a, b, 2004; Cattral *et al.*, 1999). To develop an intelligent decision support system, we need a holistic view on the various tasks to be carried out including data management and knowledge management (Eom, 1999; Nemati *et al.*, 2002). The focus of this paper is to develop a Tactical Air Combat Decision Support System (TACDSS) with minimal prior knowledge, which could also provide optimal decision scores. As shown in Fig. 1, we propose a concurrent unsupervised neural network to cluster the decision regions and genetic programming techniques to automatically generate the decision scores. Section 2 presents the problem of decision making in tactical air combat system. In Sec. 3, we introduce some theoretical concepts of Self Organising Map (SOM) (Kohonen, 1982) followed by the genetic programming techniques namely Linear Genetic programming (LGP) (Banzhaf *et al.*, 1998), Multi-Expression Programming (MEP) (Oltean and Grosan, 2003), Gene Expression programming (GEP) (Ferreira, 2001) and the proposed ensemble method. Experimentation results are provided in Sec. 4 and some conclusions are also provided towards the end.

2. The Tactical Air Combat Environment

Figure 2 presents a typical scenario of air combat tactical environment where the Airborne Early Warning and Control (AEWC) is performing surveillance operation using four fighter jets. An air-to-air fuel tanker is on the station and the location and status are known to the AEWC. Four fighter jets are on patrol in the area of Combat Air Patrol (CAP). Sometime later, the AEWC on-board sensor detects 2 hostile aircrafts. When the hostile aircrafts enter the surveillance region (shown as dashed circle) the mission system software is able to identify the enemy aircraft and its distance from the fighter jets in the ground base or in the CAP. The mission operator has few options to make a decision on the allocation of fighter jets to intercept the enemy aircrafts.

- Send the fighter jets directly to the spotted area and intercept.
- Call the fighter jets in the area back to the ground base and send another fighter jet from the ground base.
- Call the fighter jets in the area to refuel before intercepting the enemy aircraft.

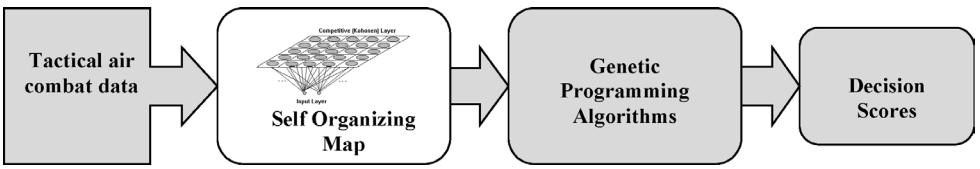


Fig. 1. Concurrent unsupervised and genetic programming models for decision support systems.

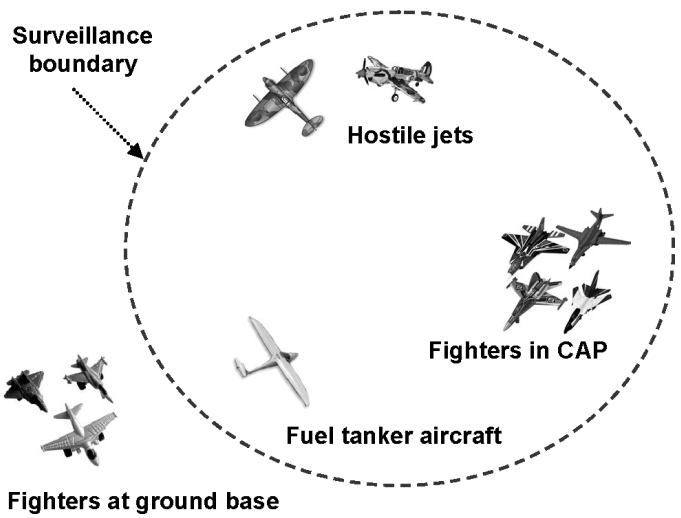


Fig. 2. A simple scenario of the air combat.

The mission operator will base the decisions on a number of decision factors, such as:

- fuel used and weapon status of fighter jets in the area;
- interrupt time required by the fighter jets in the ground base and the fighter jets at the CAP to stop the hostile;
- the speed of the enemy fighter aircraft and the type of weapons it possesses;
- the information of enemy aircraft on the type of aircraft, weapon and number of aircrafts.

From the above simple scenario, it is evident that there are several important decision factors of the tactical environment that might directly affect the air combat decision. In the simple tactical air combat, the four decision factors that could affect the decision options for calling the fighter jets in the CAP or the fighter jets at the ground base are the following:

- "fuel status" — quantity of fuel available to perform the intercept;
- "weapon possession status" — quantity of weapons available in the fighter jet;
- "interrupt time"— time required by the fighter jet to interrupt the hostile; and
- "danger situation" — information of the fighter jet and the hostile aircraft.

Each factor has different range of units such as the fuel status (0 to 1000 l), interrupt time (0 to 60 min), weapon status (0% to 100%) and danger situation (0 to 10 points). We used the following two expert rules for developing the fuzzy inference system.

- The decision selection will have small value if the fuel status is low, the interrupt time is slow, the fighter jet has low weapon status, and the danger situation is high.
- The decision selection will have high value if the fuel status is full, the interrupt time is fast, the fighter jet has high weapon status and the danger situation is low.

In a tactical air combat environment, decision-making is always based on all the states of decision factors. But sometimes, a mission operator or commander could make a decision based on a important factors, such as the fuel used is too low, the enemy has more powerful weapons, quality and quantity of enemy aircraft and so on. Table 1 depicts some typical scores (key decision selection criteria) taking into account of the various tactical air combat decision factors.

Table 1. Decision factors for the tactical air combat.

Fuel used	Time intercept	Weapon status	Danger situation	Decision
Full	Fast	Sufficient	Very dangerous	Good
Half	Normal	Enough	Dangerous	Acceptable
Low	Slow	Insufficient	Endanger	Bad

3. Learning Decision Regions Using Hybrid Unsupervised and Supervised Learning Paradigms

3.1. *Self-organising maps (SOM)*

The Kohonen's projection algorithm is the fundamental idea of unsupervised, competitive learning, self-organisation and global ordering (Kohonen, 1982). An input from the original high-dimensional space causes dominant response of one neuron in the 2D array of neurons, and only this "*winning*" neuron together with its *neighbouring neurons* get to adjust their weights. For example, adjusting weights of neurons in a local neighbourhood around the winning neuron leads to global ordering through continuous learning. This operation of the SOM algorithm shows the ability of biological neurons that perform global ordering based on local interactions. This global order leads to the creation of natural structures and biologically motivated configurations and shapes, which are created according to the laws of minimum energy, time, or complexity. The SOM algorithm for clustering the decision regions is given below:

1. Initialise the weight w_{ij}, neighbourhood size $N_m(0)$ and parameter functions $\alpha(t)$ and $\sigma^2(t)$.
2. Select the training term vector x_i at random for the input layer and calculate the similarity (distance) d of this input to the weight w of each node j.

$$d_j = \|x - w_j\| = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2} \tag{1}$$

3. Select the node with the minimum distance as the winner vector m.
4. Update the weights connecting the input layer to the winning node and its neighbouring nodes by the learning rule

$$w_{ij}(t+1) = w_{ij}(t) + c\lfloor x_i - w_{ij}(t) \rfloor, \tag{2}$$

where $c = \alpha(t) \exp(-\|r_i - r_m\| / \sigma^2(t))$ for all nodes j in $N_m(t)$.

5. Repeat steps 2–4 by increasing t by 1 at a time and decreasing the neighbourhood size, $\alpha(t)$ and $\sigma^2(t)$ until the weights are stabilised.
6. Map each term to a node on the SOM and label each winning node with an associated term.

A 2D map of decision regions (with different clusters) are formed after the training process. We adopted a trial and error approach by comparing the "normalised distortion" and "quantisation error" to decide the various parameter settings of the SOM algorithm. We finally decided the parameter setting, which could minimise both "normalised distortion" and "quantisation" errors.

3.2. *Genetic programming (GP)*

Once the clusters are defined using SOM, the next step is to apply the GP models to learn the different decision regions for the given input data. LGP, MEP and GEP are the three well-known genetic programming techniques explored in this paper.

3.2.1. *Linear genetic programming (LGP)*

LPG is a variant of the GP technique that acts on linear genomes (Banzhaf *et al.*, 1998). Its main characteristics when compared to the tree-based GP lies in that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like C/C++). An alternate approach is to evolve a computer program at the machine code level, using lower level representations for the individuals. This can tremendously hasten the evolution process as, no matter how an individual is initially represented, finally it always has to be represented as a piece of machine code, as fitness evaluation requires physical execution of the individuals. The basic unit of evolution here is a native machine code instruction that runs on the floating-point processor unit (FPU). Since different instructions may have different sizes, here instructions are clubbed up together to form instruction blocks of 32 bits each. The instruction blocks hold one or more native machine code instructions, depending on the sizes of the instructions. A cross-over point can occur only between instructions and is prohibited from occurring within an instruction. However, the mutation operation does not have any such restrictions. LGP uses a specific linear representation of computer programs. Instead of the tree-based GP expressions of a functional programming language (like LISP) programs of an imperative language (like C) are evolved. An LGP individual is represented by a variable-length sequence of simple C language instructions. Instructions operate on one or two indexed variables (registers) r, or on constants $c$from pre-defined sets. The result is assigned to a destination register, for example,

$ri = rj * c$. An example of an LGP program is illustrated below:

void LGP (double *v*[8])
{
$[0] = v[5] + 73;$
$v[7] = v[3] - -59;$
if $(v[1] > 0)$
if $(v[5] > 21)$
$v[4] = v[2].v[1];$
$v[2] = v[5] + v[4];$
$v[6] = v[7].25;$
$v[6] = v[4] - 4;$
$v[1] = \sin(v[6]);$
if $(v[0] > v[1])$
$v[3] = v[5].v[5];$
$v[7] = v[6].2;$
$v[5] = v[7] + 115;$
if $(v[1] <= v[6])$
$v[1] = \sin(v[7]);$
}

An LGP can be turned into a functional representation by successive replacements of variables starting with the last effective instruction. The maximum number of symbols in an LGP chromosome is 4 * number of instructions. Evolving programs in a low-level language allows us to run those programs directly on the computer processor, thus avoiding the need of an interpreter. In this way the computer program can be evolved very quickly. An important LGP parameter is the number of registers used by a chromosome. The number of registers is usually equal to the number of attributes of the problem. If the problem has only one attribute, it is impossible to obtain a complex expression such as the quartic polynomial. In that case we have to use several supplementary registers. The number of supplementary registers depends on the complexity of the expression being discovered. An inappropriate choice can have disastrous effects on the program being evolved. LGP uses a modified steady-state algorithm. The initial population is randomly generated.

The following steps are repeated until a termination criterion is reached:

Four individuals are randomly selected from the current population. The best two of them are considered the winners of the tournament and will act as parents. The parents are recombined and the offsprings are mutated and then replaced with the losers of the tournament.

We used an LGP technique that manipulates and evolves a program at the machine code level. The settings of various linear genetic programming system parameters are of utmost importance for successful performance of the system. The population space has been sub-divided into multiple sub-population or demes. Migration of individuals among the sub-populations causes evolution of the entire population. It helps to maintain diversity in the population, as migration is restricted among the demes. Moreover, the tendency towards a bad local minimum in one deme can be countered by other demes with better search directions. The various LGP search parameters are the mutation frequency, cross-over frequency and the reproduction frequency. The cross-over operator acts by exchanging sequences of instructions between two tournament winners. Steady-state genetic programming approach was used to manage the memory more effectively.

3.2.2. *Multi-expression programming (MEP)*

MEP genes are represented by sub-strings of variable lengths (Oltean and Grosan, 2003). The number of genes per chromosome is constant. This number defines the length of the chromosome. Each gene encodes a terminal or a function symbol. A gene that encodes a function includes pointers towards the function arguments. Function arguments always have indices of lower values than the position of the function itself in the chromosome. The proposed representation ensures that no cycle arises while the chromosome is decoded (phenotypically transcripted). According to the proposed representation scheme, the first symbol of the chromosome must be a terminal symbol. In this way, only syntactically correct programs (MEP individuals) are obtained. An example of chromosome using the sets $F = \{+, *\}$ and $T = \{a, b, c, d\}$ is given below:

1: a
2: b
3: + 1, 2
4: c
5: d
6: + 4, 5
7: * 3, 6

The maximum number of symbols in MEP chromosome is given by:

number of symbols = (n+1) * (number of genes – 1) + 1,

where n is the number of arguments of the function with the greatest number of arguments. The maximum number of effective symbols is achieved when each gene (excepting the first one) encodes a function symbol with the highest number of arguments. The minimum number of effective symbols is equal to the number of genes and it is achieved when all genes encode terminal symbols only. The translation of a MEP chromosome into a computer program represents the phenotypic transcription of the

MEP chromosomes. Phenotypic translation is obtained by parsing the chromosome top–bottom. A terminal symbol specifies a simple expression. A function symbol specifies a complex expression obtained by connecting the operands specified by the argument positions with the current function symbol. For example, genes 1, 2, 4 and 5 in the previous example encode simple expressions formed by a single terminal symbol. These expressions are:

$$E_1 = a,$$
$$E_2 = b,$$
$$E_4 = c,$$
$$E_5 = d,$$

Gene 3 indicates the operation + on the operands located at positions 1 and 2 of the chromosome. Therefore, gene 3 encodes the expression: $E_3 = a + b$. Gene 6 indicates the operation + on the operands located at positions 4 and 5. Therefore, gene 6 encodes the expression: $E_6 = c + d$. Gene 7 indicates the operation * on the operands located at positions 3 and 6. Therefore, gene 7 encodes the expression: $E_7 = (a + b) * (c + d)$. E_7 is the expression encoded by the whole chromosome. There is neither practical nor theoretical evidence that one of these expressions is better than the others. This is why each MEP chromosome is allowed to encode a number of expressions equal to the chromosome length (number of genes). The chromosome described above encodes the following expressions:

$$E_1 = a,$$
$$E_2 = b,$$
$$E_3 = a + b,$$
$$E_4 = c,$$
$$E_5 = d,$$
$$E_6 = c + d,$$
$$E_7 = (a + b) * (c + d).$$

The value of these expressions may be computed by reading the chromosome top–bottom. Partial results are computed by dynamic programming and are stored in a conventional manner. Due to its multi-expression representation, each MEP chromosome may be viewed as a forest of trees rather than as a single tree, which is the case of GP. The chromosome fitness is usually defined as the fitness of the best expression encoded by that chromosome. For instance, if we want to solve symbolic regression problems, the fitness of each sub-expression E_i may be computed using the formula:

$$f(E_i) = \sum_{k=1}^{n} |o_{k,i} - w_k|,$$

where $o_{k,i}$ is the result obtained by the expression E_i for the fitness case k and w_k is the targeted result for the fitness case k. In this case the fitness needs to be minimised. The fitness of an individual is set to be equal to the lowest fitness of the expressions encoded in the chromosome. When we have to deal with other problems, we compute the fitness of each sub-expression encoded in the MEP chromosome. Thus, the fitness of the entire individual is supplied by the fitness of the best expression encoded in that chromosome.

3.2.3. *Gene expression programming (GEP)*

The individuals of gene expression programming (Ferreira, 2001) are encoded in linear chromosomes which are expressed or translated into expression trees (branched entities). Thus, in GEP, the genotype (the linear chromosomes) and the phenotype (the expression trees) are different entities (both structurally and functionally) that, nevertheless, work together forming an indivisible whole. In contrast to its analogous cellular gene expression, GEP is rather simple. The main players in GEP are only two: the chromosomes and the Expression Trees (ETs), the latter being the expression of the genetic information encoded in the chromosomes. As in nature, the process of information decoding is called translation. And this translation implies obviously a kind of code and a set of rules. The genetic code is very simple: a one-to-one relationship between the symbols of the chromosome and the functions or terminals they represent. The rules are also very simple: they determine the spatial organisation of the functions and terminals in the ETs and the type of interaction between sub-ETs. GEP uses linear chromosomes that store expressions in breadth-first form. A GEP gene is a string of terminal and function symbols. GEP genes are composed of a *head* and a *tail*. The head contains both function and terminal symbols. The tail may contain terminal symbols only. For each problem the head length (denoted h) is chosen by the user. The tail length (denoted by t) is evaluated by:

$$t = (n - 1)h + 1,$$

where n is the number of arguments of the function with more arguments.

Let us consider a gene made up of symbols in the set S:

$$S = \{., /, +, -, a, b\}.$$

In this case, $n = 2$. If we choose $h = 10$, then we get $t = 11$, and the length of the gene is $10 + 11 = 21$. Such a gene is given below:

$$C_{\text{GEP}} = +*ab-+aab+ababbbababb.$$

The expression encoded by the gene C_{GEP} is:

$$E = a + b * ((a + b) - a).$$

GEP genes may be linked by a function symbol in order to obtain a fully functional chromosome. In the current version of GEP the linking functions for algebraic expressions are addition and multiplication. A single type of function is used for linking multiple genes. GEP uses mutation, recombination and transposition. GEP uses a generational algorithm and the initial population is randomly generated.

The following steps are repeated until a termination criterion is reached: A fixed number of the best individuals enter the next generation (elitism). The mating pool is filled by using binary tournament selection. The individuals from the mating pool are randomly paired and recombined. Two offsprings are obtained by recombining two parents. The offsprings are mutated and they enter the next generation.

There are some problems regarding multi-genic chromosomes. Generally, it is not a good idea to assume that the genes may be linked either by addition or by multiplication. Providing a particular linking operator means providing partial information to the expression which is discovered. But, if all the operators $\{+, -, ., /\}$ are used as linking operators, then the complexity of the problem substantially grows (since the problem of determining how to mix these operators with the genes is as difficult as the initial problem). Furthermore, the number of genes in the GEP multi-genic chromosome raises a problem. For some problems, the success rate of GEP increases with the number of genes in the chromosome. But, after a certain value, the success rate decreases if the number of genes in the chromosome is increased. This happens because we cannot force a complex chromosome to encode a less complex expression. A large part of the chromosome is unused if the target expression is short and the head length is large. Note that this problem arises usually in systems that employ chromosomes with a fixed length.

3.3. *Ensemble Modelling of LGP, MEP and LGP*

Our goal is to provide optimal decision scores. This could be achieved by optimising two error measures namely Root Mean Squared Error (RMSE) and Correlation Coefficient (CC):

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} |P_{\text{actual},i} - P_{\text{predicted},i}, |},$$

$$\text{CC} = \frac{\sum_{i=1}^{N} P_{\text{predicted},i}}{\sum_{i=1}^{N} P_{\text{actual},i}}.$$

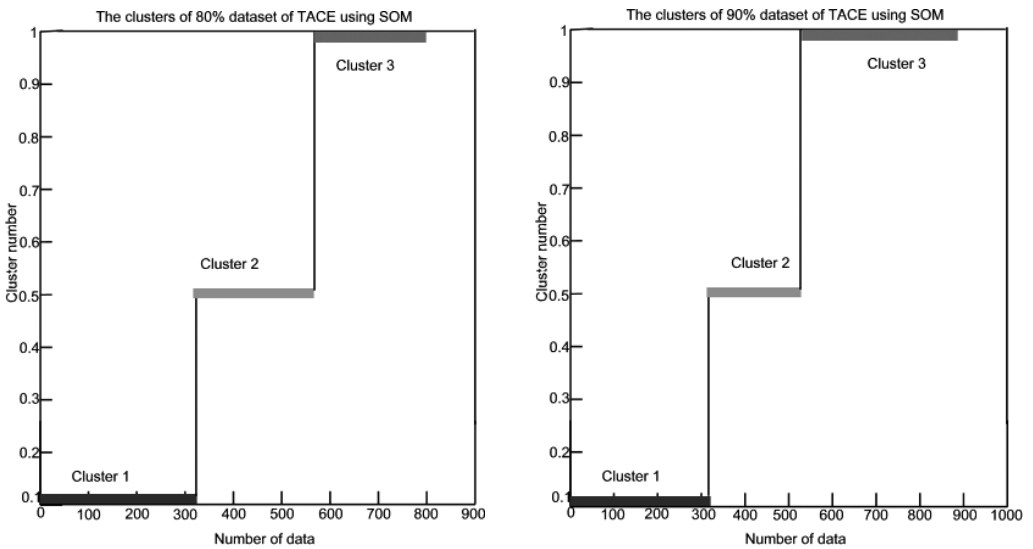
The task is to have minimal value for RMSE and a maximum value for CC. The objective is to carefully construct the different GP models to achieve the best generalisation performance. Test data is then passed through these individual models and the corresponding outputs are recorded. Suppose results obtained by LGP, MEP and GEP are a_n, b_n and c_n, respectively and the corresponding desired value is x_n. The task is to combine a_n, b_n and c_n so as to get the best output value that maximises the CC and minimises the RMSE values. We consider this problem as a multi-objective optimisation problem in which we want to find solution of this form: (coef_1, coef_2 and coef_3), where coef_1, coef_2 and coef_3 are real numbers between -1 and 1, so that the resulting combination, $\text{coef}_1{*}a_n + \text{coef}_2{*}b_n + \text{coef}_3{*}c_n$, would be close to the desired value x_n.

This means, in fact, to find a solution so as to simultaneously optimise RMSE and CC. This problem is equivalent to find the Pareto solutions of a multi-objective optimisation problem (objectives being RMSE and CC). We used the Multi-objective Evolutionary Algorithm (MOEA) — Non-dominated Sorting Genetic Algorithm II (NSGAII) (Deb *et al.*, 2002). A short description of this algorithm is given in Sec. 3.40.

3.4. *Non-dominated sorting genetic algorithm II (NSGA II)*

In the NSGA II algorithm, for each solution x the number of solutions that dominate solution x is calculated. The set of solutions dominated by x is also calculated. The first front (the current front) of the solutions that are non-dominated is obtained. Let us denote by S_i the set of solutions that are dominated by the solution x^i. For each solution x^i from the current front consider each solution x^q from the set S_i. The number of solutions that dominates x^q is reduced by one. The solutions which remain non-dominated after this reduction will form a separate list. This process continues using the newly identified front as the current front.

Let $P(0)$ be the initial population of size N. An offspring population $Q(t)$ of size N is created from current population $P(t)$. Consider the combined population $R(t) = P(t) \cup Q(t)$. Population $R(t)$ is ranked according to non-domination. The fronts F_1, $F_2, \ldots$ are obtained. New population $P(t+1)$ is formed by considering individuals from the fronts F_1, F_2, ... until the population size exceeds N. Solutions of the last allowed front are ranked according to a crowded comparison relation. NSGA II uses a parameter called *crowding distance* for density estimation for each individual. Crowding distance of a solution x



(a) (b)

Fig. 3. Developed clusters using SOM (a) Dataset A (b) Dataset B.

is the average side-length of the cube enclosing the point without including any other point in the population. Solutions of the last accepted front are ranked according to the crowded comparison distance.

NSGA II works as follows: Initially, a random population, which is sorted based on non-domination, is created. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Binary tournament selection, recombination and mutation are used to create an offspring population. A combined population is formed from the parent and offspring population. The population is sorted according to the non-domination relation. The new parent population is formed by adding the solutions from the first front and the followings until exceed the population size. Crowding comparison procedure is used during the population reduction phase and in the tournament selection for deciding the winner.

4. Experiment Results and Performance Analysis

Our master data set comprises of 1000 numbers. To avoid any bias on the data, from the master dataset, we randomly created two sets of training (Dataset A — 90% and Dataset B — 80%) and test data (10% and 20%). In addition to the four input variables (fuel used, time intercept, weapon status and danger situation) as illustrated in Table 1, we also used the cluster information generated using SOM algorithm to train the GP models. All the experiments were repeated three times and the average errors are reported.

4.1. *Unsupervised training of SOM*

The SOM algorithm provided three clusters C_1, C_2 and C_3 as depicted in Figs. 3(a) and 3(b).

4.2. *Learning the decision regions*

Parameters used by LGP, MEP and GEP and the ensembles between LGP, MEP and GEP using NSGA II are depicted in Tables 2, 3, 4 and 5, respectively. the RMSE and CC values obtained for Datasets A and B using LGP, MEP, GEP and the ensembles between LGP, MEP and GEP using NSGA II are presented in Table 6.

Table 2. Parameters used by LGP.

Parameters	Values
Population size	50
Mutation frequency	95%
Cross-over frequency	95%
Number of demes	10
Program size	
Initial	80
Maximum	1024

Table 3. Parameters used by MEP.

Parameters	Values
Population size	50
Number of mutations per chromosome	3
Cross-over probability	0.8
Code length	40
Number of generations	50
Tournament size	4

Table 4. Parameters used by GEP.

Parameter	Value
Population size	50
Mutation probability	0.044
Cross-over probability (one-point cross-over)	0.3
Number of genes	3
Genes recombination	0.1
Genes transposition	0.1
Inversion	0.1

Table 5. Parameters used by ensemble.

Parameters	Values
Population size	250
Number of generations	500
Cross-over probability	0.6

Table 6. RMSE of decision scores using GP and ensemble models for the training and test datasets.

Performance measure	LGP	MEP	GEP	Ensemble
		Dataset A		
RMSE Test	0.09989	0.07225	0.06693	**0.0485**
CC	0.993	0.994	0.989	**0.9981**
		Dataset B		
RMSE Test	0.056864	0.05927	0.07666	**0.0496**
CC	0.988	0.992	0.987	**0.9962**

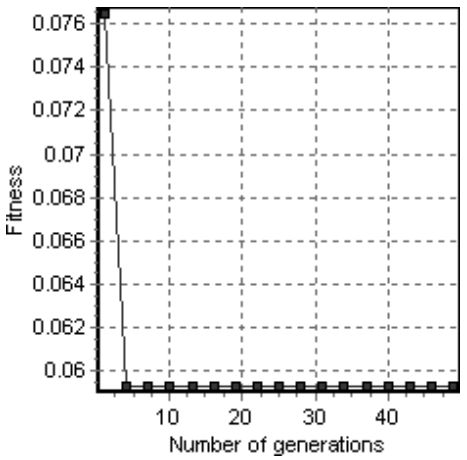


Fig. 4. Dataset A: relationship between fitness function and generations obtained by MEP.

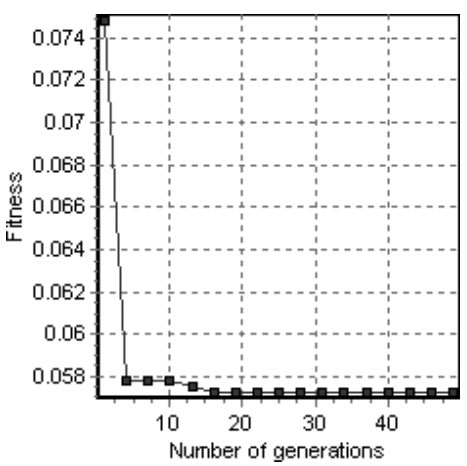
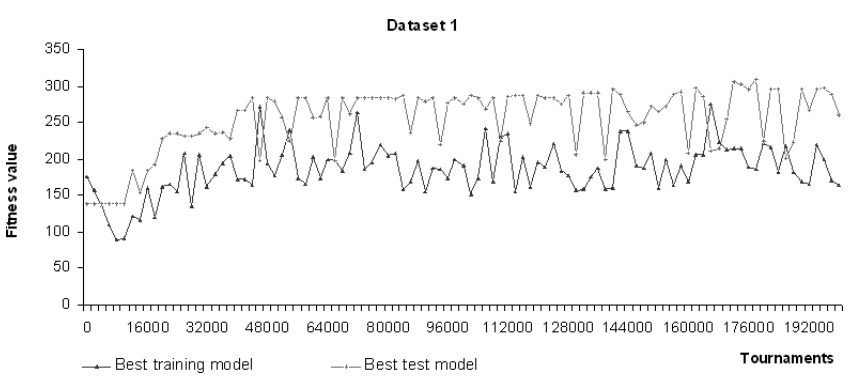
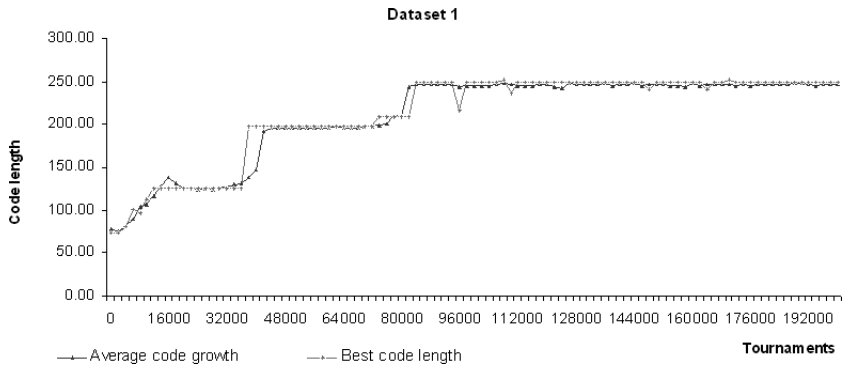


Fig. 5. Dataset B: relationship between fitness function and generations obtained by MEP.



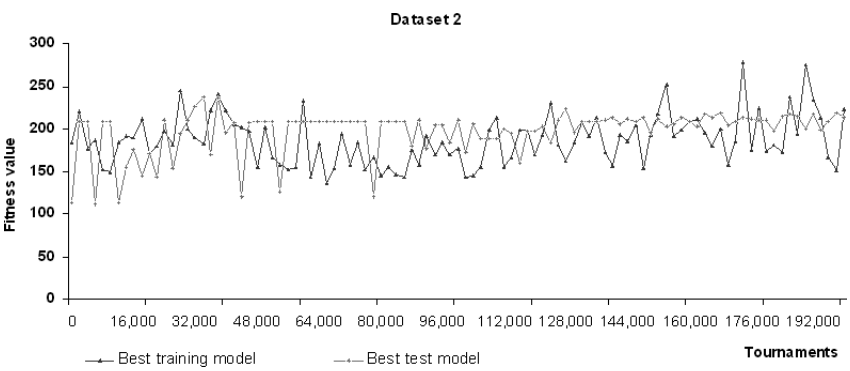
(a)

Fig. 6. (a) Dataset A: LGP models showing the evolution of best training and test fitness values during 200,000 tournaments; (b) average and best code length.

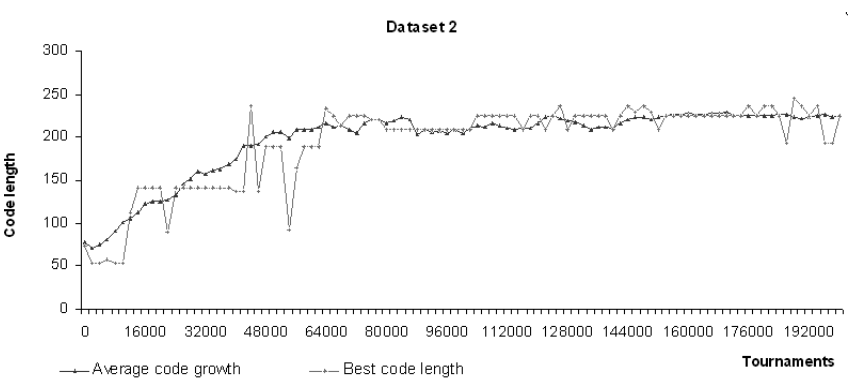


(b)

Fig. 6. (*Continued*)



(a)



(b)

Fig. 7. (a) Dataset B: LGP models showing the evolution of best training and test fitness values during 200,000 tournaments; (b) average and best code length.

The functions evolved by MEP (combining these variables and also using some constants) are reported below:

— for the first data set, the derived function is: var_3 (where var_1, var_2, var_3 and var_4 are the four input variables);
— for the second data set, the derived function is: $0.93786 + \text{var}_1 - 0.79537 - 0.044892$.

Relationship between the number of generations and the value of fitness function (RMSE) for training data obtained by MEP for the first data set and the second data set are depicted in Figs. 4 and 5, respectively. Figures 6 and 7 illustrate LGP models showing the evolution of best training and test fitness values and the average and best code length for the first and second data set,

respectively during 200,000 tournaments. From the experimental results, it is clear that the results obtained by the ensemble between GP techniques outperformed each of the individual techniques (LGP, MEP and GEP) in terms of lower RMSE and higher CC values.

5. Conclusion and Future Research

In this paper, we proposed a hybrid unsupervised–supervised training method to develop a decision support system where not much prior knowledge about the decision regions is available. We investigated the performance of three different GP techniques (LPG, MEP and GEP) to learn the different decision regions. LGP, MEP and GEP techniques were combined using an ensemble approach by an evolutionary multi-objective algorithm so as to simultaneously optimise the performance measures (RMSE and CC). We evolved a set of coefficients in order to obtain an ensemble combination of the two techniques by applying a multi-objective evolutionary algorithm.

Two data sets were considered in the experiments. Empirical results reveal that GEP outperforms LGP and MEP for the dataset A and for the dataset B, LGP obtains the best results. MEP performed well for both the data sets while GEP (the best algorithm for the first data set) performed worst for the second data set and LGP (the worst for the first data set) performed the best for the second data set. Empirical results also illustrate that a combination of these techniques is very useful. The results obtained by an ensemble of these paradigms clearly outperform results obtained by each technique individually.

Acknowledgement

This research was supported by the International Joint Research Grant of the IITA (Institute of Information Technology Assessment) foreign professor invitation programme of the MIC (Ministry of Information and Communication), Korea.

References

Tran, C, A Abraham and L Jain (2004). Modeling decision support systems using hybrid neurocomputing. *Neurocomputing Journal*, Elsevier Science, Netherlands, 61C, 85–97.

Tran, C, A Abraham and L Jain (2001). *Adaptive Database Learning in Decision Support System Using Evolutionary Fuzzy Systems: A Generic Framework, Hybrid Information Systems, Advances in Soft Computing*. Germany: Physica-Verlag, pp. 237–252.

Tran, C, Abraham A and L Jain (2002a). TACDSS: adaptation using a hybrid neuro-fuzzy system. In *7th Online World Conf. Soft Computing in Industrial Applications (WSC7), Advances in Soft Computing: Engineering Design and Manufacturing*, JM Benitez *et al.* (eds.), pp. 53–62. Germany: Springer-Verlag.

Tran, C, A Abraham and L Jain (2002b). Adaptation of mamdani fuzzy inference system using neuro-genetic approach for tactical air combat decision support system. In *15th Australian Joint Conf. Artificial Intelligence (AI'02), LNAI 2557*, pp. 672–679. Germany: Springer-Verlag.

Cattral, R, F Oppacher and D Deogo (1999). Rule acquisition with a genetic algorithm. In *Proc. Congress on Evolution Computation, CEC99*, Vol. 1, pp. 125–129.

Eom, SB (1999). Decision support systems research: current state and trends. *Industrial Management and Data Systems*, 99(5), 213–221.

Kohonen, T (1982). Self organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59–69.

Leal de Matos, PA and PL Powell (2003). Decision support for flight re-routing in Europe. *Decision Support Systems*, 34(4), 397–412.

Moynihan, GP, P Purushothaman, RW McLeod and WG Nichols (2002). DSSALM: a decision support system for asset and liability management. *Decision Support Systems*, 33(1), 23–38.

Nemati, HR, DM Steiger, LS Iyer and RT, Herschel (2002). Knowledge warehouse: an architectural integration of knowledge management, decision support. *Decision Support Systems*, 33(2), 143–161.

Papamichail, KN and S French (2003). Explaining and justifying the advice of a decision support system: a natural language generation approach. *Expert Systems with Applications*, 24(1), 35–48.

Parker, DB (1985). Learning-logic, Report TR-47. Cambridge, MA. Massachusetts Institute of Technology, Center for Computational Research in Economics and Management Science.

Oltean, M and C Grosan (2003). Evolving evolutionary algorithms using multi expression programming. In *Proc. 7th European Conference on Artificial Life*, pp. 651–658. Germany: Dortmund.

Banzhaf, W, P Nordin ER Keller and FD Francone (1998). *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. USA: Morgan Kaufmann Publishers, Inc.

Ferreira, C (2001). Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, 13(2), 87–129.

Deb, K, A Pratap, S Agarwal and T Meyarivan (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6(2), 181–197.

Abraham, A, C Grosan, C Tran and L Jain (2005). A concurrent neural network — genetic programming model for decision support systems. In *Knowledge Management Nurturing Culture, Innovation and Technology, 2005 International Conference on Knowledge Management (ICKM 2005), USA*, S. Hawamdeh (ed.), pp. 231–245. Singapore: World Scientific Press.

Ajith Abraham currently works as a Professor under the South Korean Government's Institute of Information Technology Assessment (IITA) Professorship programme at Yonsei University, Seoul. His primary research interests are in computational intelligence with a focus on using evolutionary computation techniques for designing intelligent paradigms. Application areas include Web services, information security, Web intelligence, financial modelling, multi-criteria decision-making, data mining applications, etc. He has authored/co-authored over 250 research publications in peer-reviewed reputed journals, book chapters and conference proceedings of which four have won "best paper" awards.

He is an editor of the *Journal of Information and Knowledge Management* (JIKM) and also serves the editorial board of over a dozen international journals. He has also guest-edited 16 special issues for reputed international journals. He received his PhD degree from Monash University, Australia.

Crina Grosan currently works as an Assistant Professor in the Computer Science Department of Babes-Bolyai University, Cluj-Napoca, Romania. She received her PhD degree from Babes-Bolyai University, Romania. Her main research area is in evolutionary computation, with a focus on evolutionary multi-objective optimisation and applications, and genetic programming. She is also interested in Swarm Intelligence (Particle Swarm Optimisation, Ant Colonies Systems) and applications in bioinformatics, internet security, financial modelling, etc. Dr. Grosan authored/co-authored over 70 articles in peer-reviewed international journals, proceedings of the international conferences and book chapters. She is co-author of two books in the field of computer science. She is the Managing Editor of the *International Journal of Computational Intelligence Research* (IJCIR). Dr. Grosan has guest-edited a special issue on "Soft Computing in Simulation and Modeling" for the *International Journal of Simulation Systems, Science and Technology*, UK. She has co-edited two books *Swarm Intelligence and Data Mining* and *Stigmergic optimisation* which were published by Springer-Verlag, Germany in 2006. Dr. Grosan is the co-editor of the following two books which will be published by Springer-Verlag, Germany in 2007: *Hybrid Evolutionary Systems* and *Engineering Evolutionary Intelligent Systems*.