Fuzzy control in robot-soccer, evolutionary learning in the first layer of control

Peter J THOMAS
Faculty of Engineering and Physical Systems, Central Queensland University, Rockhampton, QLD 4702, Australia

and

Russel J STONIER
Faculty of Informatics and Communication, Central Queensland University, Rockhampton, QLD 4702, Australia

ABSTRACT

In this paper an evolutionary algorithm is developed to learn a fuzzy knowledge base for the control of a soccer playing micro-robot from any configuration belonging to a grid of initial configurations to hit the ball along the ball to goal line of sight. The knowledge base uses relative co-ordinate system including left and right wheel velocities of the robot. Final path positions allow forward and reverse facing robot to ball and include its physical dimensions.

Keywords: Evolutionary Algorithm, Fuzzy Logic Controller, Robot-Soccer.

1. INTRODUCTION

Determination of a fuzzy logic Knowledge Base (KB) to satisfactorily control a system is usually built from expert knowledge. If an expert is not available, some other method of building the KB is needed, whether the KB is a single layer, multi-layer or hierarchical structure. This is especially the case in robot-soccer. In the absence of expert knowledge, the KB of a fuzzy logic controller can be learnt using evolutionary algorithms.

The authors in [1] learnt a three-level hierarchical fuzzy controller system using an evolutionary algorithm to solve a point mass collision-avoidance problem in a simulated two robot system. Individual robot paths learnt using a complete KB can be combined by a *fuzzy amalgamation* post-evolutionary process [2, 3, 4, 5]. Combining individual robot paths learnt using a complete KB and using fuzzy amalgamation 'on-the-fly' during the evolutionary learning was shown in [6, 7].

Evolutionary learning of a fuzzy controller for a micro-robot soccer playing robot was presented in [8]. In this paper an evolutionary algorithm was used to learn the fuzzy rules of a three input two output fuzzy controller. Further, relative co-ordinates were used in preference to Cartesian co-ordinates for two main reasons: a reduction in the number of rules in the fuzzy KB, and learning can be performed using a single ball position. Simulation of the robot movement incorporated physical size and permitted forward and reverse facing impact with the ball as 'good' solutions. Special consideration was given for robot initial configurations touching the ball.

Input variables for the fuzzy control system were taken to be the position of the robot relative to the ball, described by $n = 3$ variables $x_1 = d^2$, $x_2 = \theta$ and $x_3 = \phi_r$ as shown in Figure 1.

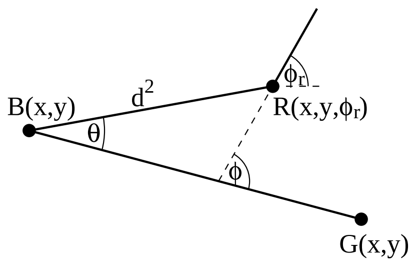

Figure 1: Relative co-ordinate parameters

Here x_1 is the square of the distance between the robot centre and the ball, x_2 is the angle between robot to ball line and the ball to goal line, x_3 is the projection of the robot heading angle ϕ_R onto the ball to goal line.

The velocity of the left and right wheels, v_L and v_R control the motion of the robot. These two variables were taken as output of the fuzzy control system described below (momentum and friction were ignored in the kinematics).

A key feature not included in this analysis is, that the left and right wheel velocities should also be inputs into the knowledge base. For inherent along any path followed by the robot, its current velocity is a key input variable to establish control.

In this paper we extend the results published in [8], to include the left and right wheel velocities as input variables to the fuzzy knowledge base defining the fuzzy controller. We shall first define the robot soccer system, the design of the new fuzzy controller and the design of the evolutionary algorithm to learn the fuzzy rules and finish with a short presentation of results for control of the robot from a far distance from the ball and from configurations close and touching the ball.

2. ROBOT SOCCER SYSTEM

The basic robot soccer system considered is that defined for the Federation of Robot-soccer Association, Robot World Cup (*www.fira.net*). Full specifications of hardware, software and basic robot strategies that are employed in this type of micro-robot soccer system can be found in [9]. All calculations for vision data processing, strategies and position control of the robots are performed on a centralised host computer.

Kinematics
The kinematics of a wheel chair style robot is given by Equation 1 from [10].

$$\begin{bmatrix} v_C \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \tag{1}$$

where v_L is the instantaneous speed at the left wheel of the robot, v_R is the instantaneous speed at the right wheel of the robot, L is the wheel base length, v_C is the instantaneous speed of the robot centre, ω is the instantaneous angular speed about the instantaneous point of rotation (x_0, y_0). The radius of the arc r is determined from $v_C = r\omega$, which is the distance between (x_0, y_0) and v_C.

Let the next robot position be approximated by a small time interval Δt. Assume v_L and v_R are constant over this interval. If $\omega = 0$, the robot is moving in a straight line. Equation 2 gives the next robot position using linear displacement $\Delta s = v_C \Delta t$.

$$\begin{bmatrix} x'_R \\ y'_R \\ \phi'_R \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ 0 \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\phi_R) \\ \Delta s \sin(\phi_R) \\ \phi_R \end{bmatrix} \tag{2}$$

When $\omega \neq 0$, the robot scribes an arc. Curvilinear robot paths are calculated using translation of the instantaneous point (x_0, y_0) of rotation to the origin, rotation about the origin and translation of the origin back to the point of instantaneous rotation. Figure 2 shows the symbols used in the curvilinear Equations 3 4 5.

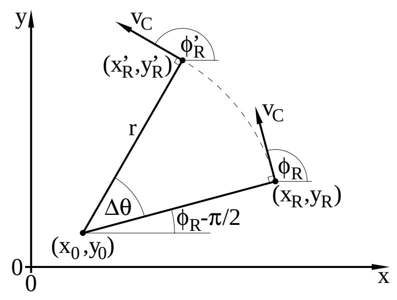

Figure 2: Curvilinear formulae symbols

$$\begin{aligned} x'_R &= x_R + r(\sin(\phi_R + \Delta\theta) - \sin(\phi_R)) && (3) \\ y'_R &= y_R - r(\cos(\phi_R + \Delta\theta) - \cos(\phi_R)) && (4) \\ \phi'_R &= \phi_R + \Delta\theta && (5) \end{aligned}$$

where $\phi'_R \in [0, 2\pi)$ is necessarily constrained for input into the fuzzy system. The following parameter values were used: $L = 68.5(mm), \Delta t = 1/60(s)$. The playable region was defined as a rectangle from coordinate $(0, 0)$ to $(1500, 1300)$ (measurements in mm.), ignoring the goal box at each end of the field.

3. FUZZY CONTROL SYSTEM DESIGN

The inputs into the fuzzy knowledge base are now variables x_1, x_2 and x_3 defined previously and the two wheel velocities $x_4 = v_L$ and $x_5 = v_R$. Figure 3 shows the fuzzy input sets used for each variable. There are seven linguistic membership sets defined for each variable. For both angles θ and ϕ: **VS** is Very Small, **S** is Small, **SM** is Small Medium, **M** is Medium, **ML** is Medium Large, **L** is Large and **VL** is Very Large. Distance squared x_1: **VC** is Very Close, **C** is Close, **CN** is Close Near, **N** is Near, **NF** is Near Far, **F** is Far and **VF** is Very Far. For left wheel velocity v_L and right wheel velocity v_R: **FR** is Fast Reverse, **MR** is Medium Reverse, **SL** is Slow Reverse, **S** is Stationary, **SF** is Slow Forward, **MF** is Medium Forward and **FF** is Fast Forward.

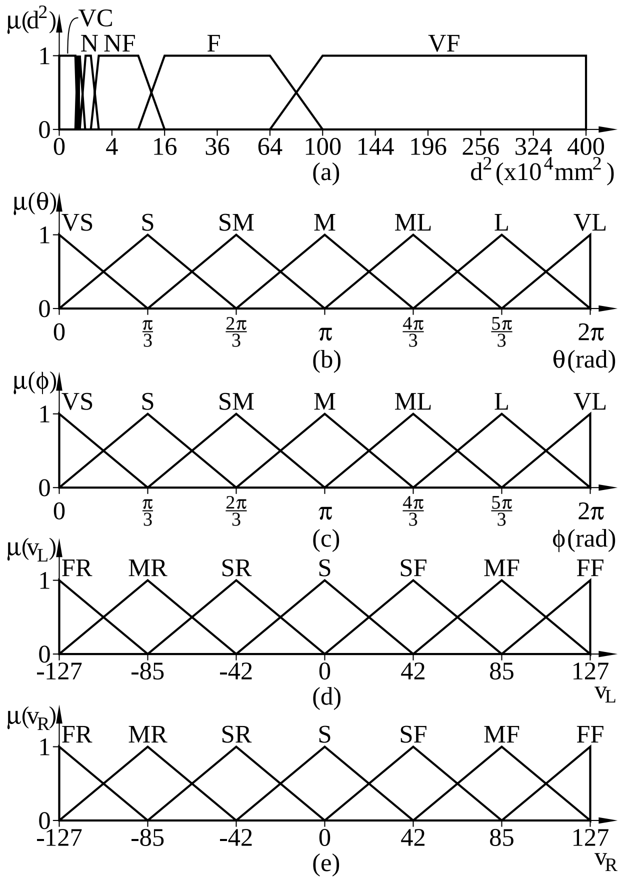

Figure 3: Fuzzy Input Sets

The new output variables are the changes in the two wheel velocities, that is, $y_1 = \Delta v_L$ and $y_2 = \Delta v_R$, and the new updated velocities are simply calculated as

$$v_L^{'} = v_L + \Delta v_L \quad \text{and} \quad v_R^{'} = v_R + \Delta v_R$$

We define eight membership sets for each output variable: **VFR** is Very Fast Reverse, **FR** is Fast Reverse, **MR** is Medium Reverse, **SR** is Slow Reverse, **SF** Slow Forward, **MF** is Medium Forward, **FF** is Fast Forward, and **VFF** is Very Fast Forward. These sets have centres: $\overline{y}_k^\ell = -28 + 8k$ for $k = 0, \cdots, 7$.

In all there are now $7^5 = 16807$ rules in a complete fuzzy knowledge base for this system. In general, the ℓ^{th} fuzzy rule has the form:

If (x_1 is A_1^ℓ and x_2 is A_2^ℓ and x_3 is A_3^ℓ and x_4 is A_4^ℓ and x_5 is A_5^ℓ)

Then (y_1 is B_1^ℓ and y_2 is B_2^ℓ).

where $A_k^\ell, k = 1, \cdots, 5$ are normalised fuzzy sets for input variables $x_k, k = 1, \cdots, 5$, and where $B_m^\ell, m = 1, 2$ are normalised fuzzy sets for output variables $y_m, m = 1, 2$.

Given a fuzzy rule base with M rules, a fuzzy controller as given in Equation 6 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

$$y_k = \frac{\sum_{\ell=1}^{M} \overline{y}_k^\ell (\prod_{i=1}^{n} \mu_{A_i^\ell}(x_i))}{\sum_{\ell=1}^{M} (\prod_{i=1}^{n} \mu_{A_i^\ell}(x_i))} \qquad (6)$$

where $\overline{y}_k^\ell$ are centres of the output sets B_k^ℓ

These values, 33614 of them, are typically unknown and require determination in establishing valid output for controls to each wheel of the robot. As there is no a-priori knowledge about the system control, we used evolutionary algorithms (EA) [11] to search for an acceptable solution.

A simple method of implementing Equation 6 is to use nested loops for the summation and product terms. This simple encoding requires the loops to consider antecedents and consequents of 16807 rules.

A special feature of overlapping sets using the Mamdani product inference engine is that a minimum of one and a maximum of two membership sets for each input variable will fire. A vertical slice through any variable membership in Figure 3 illustrates this property.

A maximum of $2^5 = 32$ out of the $7^5 = 16807$ will fire for any input into the fuzzy controller. With a sparse access to the rule base, it makes sense to access the rule base by developing pointers to the rule being used.

The number of membership sets was set at seven for each input variable to make the calculation of the pointer to the rule an easy radix calculation. It is not necessary to use the same number of memberships per variable the same, it was done simply to reduce errors made in programming and ease debugging.

Each input variable is tried on the input membership sets to find the one or two memberships that fire. The set identification is stored in an array with the membership value(s). Each array is terminated by a pointer value of -1. Five input variables require five nested loops to calculate the pointer reference using the antecedents in accordance with Equation 6. The pointer refers to the consequent $\overline{y}_k^\ell$ in the string for inclusion into Equation 6.

4. EVOLUTIONARY LEARNING

Our objective here is to learn a rule base for the control of this system. The first problem is how to formulate the knowledge base as a string in the population.

Each output fuzzy set is represented by an integer in the interval $[0, 7]$ corresponding to the fuzzy output sets $\{$**VFR**, **FR**, **MR**, **SR**, **SF**, **MF**, **FF**, **VFF**$\}$. So we can uniquely represent a potential knowledge base solution as an individual string $\underset{\sim}{s}$ contains $2M = 33614$ integer represented consequents of the form:

$$\underset{\sim}{s} = \{s_1^1, s_2^1, \cdots, s_1^k, s_2^k, \cdots, s_1^M, s_2^M, \},$$

where $s_j, j = 1, 2$ is an integer in the interval $[0, 7]$.

The population at generation t, $P(t) = \{\underset{\sim}{s}^n : n = 1, \cdots, N\}$, where $N = 2000$ is the number of individuals in the population. The population at the next generation $P(t+1)$ was built using a full replacement policy, tournament selection with size $n_T = 3$, and one point crossover with probability $p_c = 0.6$. Elitism was used, with the 10 best individuals carried from population $P(t)$ to population $P(t+1)$. An incremental mutation operator with probability $p_m = 0.01$, replaced the binary mutation used previously. This mutation operator increments/decrements s_k by one with equal probability using bounds checking, that is, if $s_k = 0$, it was incremented to $s_k = 1$, and if $s_k = 7$, it was decremented to $s_k = 6$.

Fitness evaluation of each individual was calculated by scribing a path using the fuzzy controller and stopping when either:

(i) iteration (time steps) reached a prescribed limit (500), or

(ii) the path exceeded the maximum allowable distance from the ball, or

(iii) the robot collides with the ball.

In (iii) care has been taken to recognise the finite size of the robot. The robot is a square with side of 80mm and the ball has a diameter of 42.7mm. Detecting a collision of the robot and ball is made by calculating the distance of the ball centre $(x_B, y_B) = (750, 650)$ perpendicular to the line in the direction of the robot d_{NL} passing through the centre of the updated position of the robot (x'_R, y'_R), and the distance of the ball d_{AL} projected onto that line. These values are determined as:

$$d_{AL} = \frac{|(x_B - x'_R) + m(y_B - y'_R)|}{\sqrt{m^2+1}}.$$

$$d_{NL} = \frac{|(y_B - y'_R) - m(x_B - x'_R)|}{\sqrt{m^2+1}}.$$

where m is the gradient of the line passing through the robot centre, in the direction of the robot. The following quantity is used to define an exclusion region determined by the physical size of the robot:

$$r_{corner}^2 = (d_{AL} - 40)^2 + (d_{NL} - 40)^2.$$

A flag "HitBall" is raised when the following condition is true:

IF $(((d_{NL} < 40)$ AND $(d_{AL} < 61.35))$ OR
$((d_{NL} < 61.35)$ AND $(d_{AL} < 40))$ OR
$(r_{corner}^2 < 61.35^2))$
THEN (HitBall = TRUE).

A grid of initial configurations was defined: $x = -750 + 100(k-1)$ for $k = 1, \cdots, 31$ and $y = -650 + 100(k-1)$ for $k = 1, \cdots, 27$ excluding the ball position. Each grid point has five angles: $\theta = 2(k-1)\pi/5$ for $k = 1, \cdots, 5$. The total number of initial configurations is therefore $C = 5(31 \times 27 - 1) = 4180$. All initial configurations start with zero, left and right, wheel velocity.

The final position of the path obtained by running the fuzzy controller defined by the knowledge base (represented by each string) from a given grid configuration was used to evaluate the fitness of each individual:

$$f_i = \alpha_1 T_1 + \alpha_2 T_2 + \alpha_3 T_3 \quad (7)$$

where $\alpha_1 = 1.0$, $\alpha_2 = 1.0$, $\alpha_3 = 100.0$. The terminal angle coefficients was heavily weighted to ensure the correct final alignment of the robot to the ball and the target goal.

The first quantity in the fitness sum is $T_1 = d^2(R, DP)$. It is the final squared distance between the robot centre R and the destination point $DP = (688.5, 650)$ when the path is terminated as described above. It determines the accuracy of the fuzzy controller to control the system to the desired destination configuration.

Term T_2 is the iteration count for each path and is used to minimise the time taken to reach the desired destination configuration.

The third quantity $T_3 = 1000\sin^2(\phi)$ where ϕ is the final angle of the robot relative to line $\overline{BG}$. This term is included to enable forward facing and reverse facing solutions to be accepted at the final destination.

The evolutionary algorithm was terminated after a prescribed number of generations. The best individual, that is, the one with the minimum fitness, is taken as the "best" fuzzy logic controller or defining the "best" knowledge base determined by the algorithm for this problem.

In [8] the learning of the knowledge base took place by summing the fitness evaluations for ALL starting configurations in the grid. This led to heavy computation in fitness evaluation, and difficulties in achieving convergence to acceptable paths in some instances from the initial configuration to the target point. See [8], for a detailed discussion.

A new method for learning a knowledge base across the grid configuration, is employed and reported here. The evolutionary algorithm was run sequentially through the full number of initial configurations, being allowed to run for 10 generations at each configuration before moving to the next. That is, each individual in the population was evaluated with fitness derived from a single grid point for 10 generation, at which instance, fitness was then calculated at another point in the grid for the next 10 generations and so on. The evolutionary process was stopped after a total of 500 000 generations in all were completed.

5. RESULTS

The results obtained in the final "best" fuzzy knowledge were excellent, obtaining very smooth continuous paths to the target with both forward and reverse facing in the final position depending on the initial configuration. Only a very small number of aberrations existed but the paths to the target were still acceptable. We show a number of the many images obtained in Figures 4, 5, 7 and 6.

In Figures 4 and 6 the robot has final approach to the ball forward facing, and in the other two figures, reverse facing.

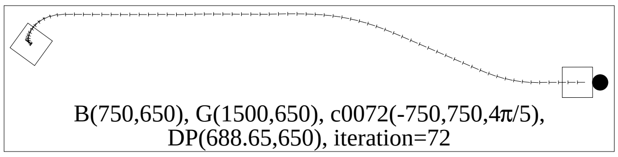
B(750,650), G(1500,650), c0072(-750,750,4π/5),
DP(688.65,650), iteration=72

Figure 4: Long distant path from left

Compared to [8], the trajectories obtained are very smooth, there is much less tendency for the robot to execute high momentum turns resulting in a "cusp" along the trajectory.

A problem with pre-mature convergence of the evolutionary algorithm was reported in [8]. The algorithm

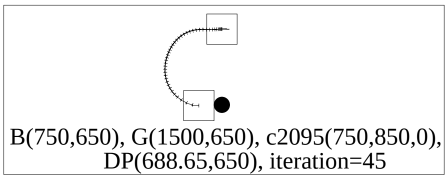
B(750,650), G(1500,650), c2095(750,850,0),
DP(688.65,650), iteration=45

Figure 5: Long distant path from right

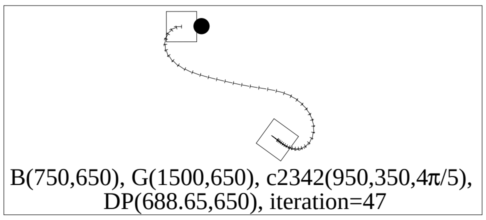
B(750,650), G(1500,650), c2342(950,350,4π/5),
DP(688.65,650), iteration=47

Figure 6: Short distant path from bottom

now outlined in this paper has shown no pre-mature convergence problems.

6. CONCLUSION

In this paper we have shown how to design a complicated fuzzy controller to control a micro-robot that incorporates both forward and reverse impact with the ball, and precise directional control to a destination behind the ball, from configurations far from the ball as well as close and touching the ball. The fuzzy controller incorporates as input current left and right wheel velocities and it's output determines changes in these two velocities. We have also shown that it is capable to learn the rules of such a knowledge base using an evolutionary algorithm.

The present analysis has been undertaken from initial configurations in which starting left and right wheel

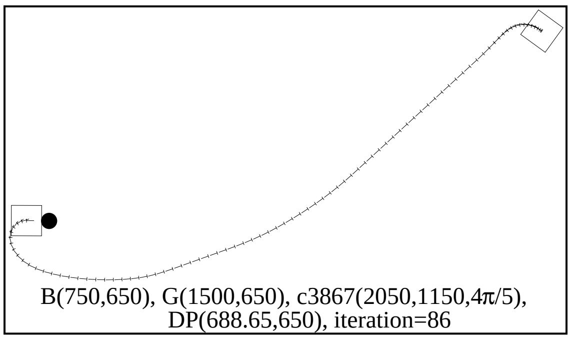
B(750,650), G(1500,650), c3867(2050,1150,4π/5),
DP(688.65,650), iteration=86

Figure 7: Short distant path from top

velocities are zero. To properly model this problem, initial configurations with non-zero velocities should be incorporated.

This requires extending the grid configuration to include the extra two variables of v_L and v_R, and in so doing drastically increases the already heavy computation in the evolutionary learning of the fuzzy rules. Research in this direction is currently being undertaken.

It has already been noted that by increasing the number of input variables, the fuzzy rule base has increased in size considerably. Further research is being undertaken on this problem by decomposing the fuzzy knowledge base into a three layered hierarchical fuzzy knowledge bases, [12], which considerably reduces the number of rules in the overall fuzzy control system and may reduce the already high computation associated with learning the fuzzy controller.

7. REFERENCES

[1] M.Mohammadian & R.J. Stonier, Hierarchical Fuzzy Control, *Proceedings of the 7th International conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, 621-629, Paris, 1998.

[2] M. Mohammadian & R.J. Stonier, Generating Fuzzy Rules by Genetic Algorithms, *Proceedings of 3rd International Workshop of Robot and Human Communication*, Nagoya, Japan, 362-367, 1994.

[3] R.J. Stonier & M. Mohammadian, Self Learning Hierarchical Fuzzy Logic Controller in Multi-Robot Systems, *Proceedings of the IEA Conference Control95*, Melbourne Australia, 381-386, October 1995.

[4] R.J. Stonier, Adaptive Learning using Genetic Algorithms and Evolutionary Programming in Robotic Systems, *First Korea–Australia Joint Workshop on Evolutionary Computing*, 183-198, 1995.

[5] M. Mohammadian, Genetic learning of fuzzy control Rules in hierarchical and multi-layer fuzzy logic systems with application to mobile robots, *PhD Thesis*, Central Queensland University, Rockhampton, Australia, 1998.

[6] M. Mohammadian & R.J. Stonier, Fuzzy Rule Generation by Genetic Learning for Target Tracking, *Proceedings of the 5th International Intelligent Systems Conference*, Reno, Nevada, 10-14, June, 1996.

[7] R.J. Stonier & M.Mohammadian, Knowledge Acquisition for Target Capture, *Proceedings of the International Conference on Evolutionary Computing ICEC'98*, 721-726, Anchorage, Alaska, 1998.

[8] P.J. Thomas & R.J. Stonier, Evolutionary learning of fuzzy control in robot-soccer, *Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'2001)*, ISBN 0 85889847 0 on CD, M. Mohammadian (Ed), Las Vegas, pp 494-503, 2001.

[9] J. H. Kim, *Lecture Notes on "Multi-Robot Cooperative System Development*, Green Publishing Co., Feb. 1998.

[10] M.-J. Jung, H.-S. Shim, H.-S. Kim, & J.-H. Kim, The Omni-directional mobile robot OmniKity-Y(OK-I) for RoboSOT category, *Proceedings of the FIRA Robot World Cup 1998*, R.J. Stonier and J. H. Kim (Eds), 37-42, 1999.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd Edition, Springer Verlag, 1994.

[12] G.V.S. RAJU & J. ZHOU, Adaptive Hierarchical Fuzzy Controller, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, No. 4, pp 973-980, 1993.