STUDIES IN *FUZZINESS*
AND *SOFT COMPUTING*

Studies in Fuzziness and Soft Computing

Bhanu Prasad (Ed.)

Soft Computing Applications in Industry

Springer

Bhanu Prasad (Ed.)

Soft Computing Applications in Industry

Bhanu Prasad (Ed.)

Soft Computing Applications in Industry

Springer

Editor

Prof. Bhanu Prasad
Florida A&M University
Department of Computer and
Information Sciences
Tallahassee, FL 32307
USA
E-mail: bhanu.prasad@famu.edu

Associate Editors

Jason Black
Florida A&M University, Tallahassee, FL, USA

Bobby Granville
Florida A&M University, Tallahassee, FL, USA

Zoran Majkic
University of Belgrade, Belgrade, Serbia

Yenumula B. Reddy
Grambling State University, Grambling, LA, USA

ISBN 978-3-540-77464-8 e-ISBN 978-3-540-77465-5

DOI 10.1007/978-3-540-77465-5

Studies in Fuzziness and Soft Computing ISSN 1434-9922

Library of Congress Control Number: 2007943077

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Contents

Optimization of Industrial Processes Using Improved and Modified Differential Evolution

B.V. Babu[1] and Rakesh Angira[2]

[1] Chemical Engineering Department & Educational Hardware Division (EHD)
`bvbabu@bits-pilani.ac.in`
[2] Chemical Engineering Department
`angira@bits-pilani.ac.in`
[1,2] Birla Institute of Technology and Science (BITS), Pilani – 333 031 (Rajasthan), India

1 Introduction

Optimization refers to finding one or more feasible solutions, which correspond to extreme values of one or more objectives. The need for finding such optimal solutions in a problem comes mostly from the extreme purpose of either designing a solution for minimum possible cost of fabrication, or for maximum possible reliability, or others. Because of such extreme properties of optimal solutions, optimization methods are of great importance in practice, particularly in engineering design, scientific experiments and business decision-making.

In most of the gradient-based traditional optimization methods, in moving along the gradient, we are guided by the extrapolation of the derivatives of the objective function with respect to appropriate variables. However, the shape of the response surface may change, thereby necessitating a change in the direction of search. In other words, we cannot move on the surface for any considerable length of time. Another biggest limitation of traditional optimization methods is that these methods are applicable only to find local minima. Hence these traditional optimization techniques do not ensure the global optimum and also have limited applications.

In the recent years, non-traditional search and optimization methods based on natural phenomena of evolution, such as Simulated Annealing (SA), Genetic algorithms (GA), Differential evolution (DE), Self Organizing Migrating Algorithms (SOMA), Particle Swarm Optimization (PSO), Tabu Search (TS), Scatter Search (SS), and Ant Colony Optimization (ACO) have been developed to overcome these problems of traditional optimization methods (Onwubolu and Babu 2004; Babu 2004; Corne et al. 1999). These algorithms are stochastic in nature, with probabilistic transition rule. These are comparatively new and are gaining popularity due to certain properties which the deterministic algorithms do not have. These are found to have better global perspective than the traditional methods.

Previous studies (Storn 1995; Wang and Chiou 1997; Babu & Sastry 1999; Babu and Angira 2003; Angira and Babu 2005a; Angira 2005) have shown that DE is an efficient, effective and robust evolutionary optimization method. Still DE takes large computational time for optimizing the computationally expensive objective functions. And therefore, an attempt to speed up DE is considered necessary. Modified Differential Evolution (MDE), an evolutionary optimization technique, is evaluated by applying to four non-linear chemical processes (Angira and Babu 2005a, 2006a, 2006b, 2006c; Babu and Angira 2005, 2006). DE has also been extended to solve multi objective optimization problems. Three such extended strategies namely MODE (Babu et al. 2005a, 2005b, 2005c), NSDE (Angira and Babu 2005b) and MNSDE (Angira and Babu 2006a) are discussed in brief.

2 DE in Brief

Differential Evolution (DE), a recent optimization technique, is an exceptionally simple evolution strategy, which is significantly faster & robust at numerical optimization and is more likely to find a function's true global optimum (Price and Storn 1997). Simple GA (Goldberg1989; Deb 1996) uses a binary coding for representing problem parameters whereas DE uses real coding of floating point numbers. Among the DE's advantages are its simple structure, ease of use, speed and robustness. The details of DE algorithm and pseudo code are available in literature (Price and Storn 1997; Babu and Sastry 1999; Onwubolu and Babu 2004; Angira 2005; Angira and Babu 2006b; Price and Storn, 2007 etc.).

Original DE dealt with a single strategy (Price and Storn 1997). Later on ten different strategies have been suggested by Price and Storn (2007). A set of control parameters that works out to be the best for a given problem may not work well when applied for a different problem. The best value of control parameters to be adopted for each problem is to be determined separately by trial & error. Similarly, the strategy that works out to be the best for a given problem may not be effective when applied to some other problem.

DE has been successfully applied in various fields. Some of the successful applications of DE include: digital filter design (Storn 1995), Batch fermentation process (Wang and Chen 1999), Estimation of heat transfer parameters in trickle bed reactor (Babu and Sastry 1999), Dynamic Optimization of a Continuous Polymer Reactor using a Modified Differential Evolution (Lee et al. 1999), Optimal design of heat exchangers (Babu and Munawar 2000), Synthesis and optimization of heat integrated distillation system (Babu and Singh 2000), Optimization of an alkylation reaction (Babu and Gaurav 2000), optimization of Low Pressure Chemical Vapor Deposition Reactors Using Hybrid Differential Evolution (Lu and Wang 2001), optimization of non-linear functions (Babu and Angira 2001a), Optimization of thermal cracker operation (Babu and Angira 2001b), Global optimization of MINLP problems (Babu and Angira 2002), Optimization of water pumping systems (Babu and Angira 2003), Supply chain Planning (Pinto 2002), Optimization of Non-Linear Chemical Processes (Babu and Angira 2006), Process Synthesis and design (Angira and Babu 2006a), Optimal Design of Complex and Non-Linear Chemical Processes (Angira and Babu 2006b) etc.

3 Improvements on DE

When using any population based search algorithm in general and DE in particular to optimize a function, an acceptable trade–off between convergence (with reference to locating optimum) and robustness (with reference to not missing the global optima) must generally be determined. Convergence implies a fast convergence although it may be to a local optimum. On the other hand, robustness guarantees a high probability of obtaining the global optimum. A few attempts have already been made to achieve this trade-off (Chiou and Wang 1999; Babu and Angira 2003; Tasoulis et al 2004; Bergeya and Ragsdaleb 2005).

Chiou and Wang (1999) embedded accelerated phase and migration phase into the original algorithm of DE. These two phases are used to improve the convergence speed without decreasing the diversity among individuals. Also, several alternate methods are compared. Babu and Angira (2003) proposed a variation of mutation and crossover scheme and investigated its effectiveness by applying to liquid extraction problem. Tasoulis et al. (2004) explored how Differential Evolution can be parallelized in a virtual parallel environment so as to improve both the speed and the performance of the method. Bergeya and Ragsdaleb (2005) proposed a DE with greedy random strategy for genetic recombination. They found that modified algorithm has higher convergence velocity than original DE still maintaining the robustness. In this paper an attempt has been made to increase the convergence speed of DE without compromising with the robustness. A modified DE is proposed and evaluated in the present work to achieve this trade-off.

4 Modified Differential Evolution (MDE)

The principle of modified DE (Angira and Babu 2005a, 2005c; 2006a, 2006b, 2006c; Babu and Angira 2006) is same as DE. The major difference between DE and MDE is that MDE maintains only one *array*. The array is updated as and when a better solution is found. Also, these newly found better solutions can take part in mutation and crossover operation in the current generation itself as opposed to DE (where another array is maintained and these better solutions take part in mutation and crossover operations in next generation). Updating the single array continuously enhances the convergence speed leading to less function evaluations as compared to DE. This modification enables the algorithm to get a better trade-off between the convergence rate and the robustness. By choosing the key parameters (NP, CR, and *F*) wisely/appropriately, the problem of premature convergence can be avoided to a large extent. Such an improvement can be advantageous in many real world problems where the evaluation of a candidate solution is a computationally expensive operation and consequently finding the global optimum or a good sub-optimal solution with the original differential evolution algorithm is too time consuming, or even impossible within the time available. This has been found to be very true in examples such as optimization in the field of computational mechanics, computational magnetics, computational fluid dynamics and unsteady solidification.

5 Application to Industrial Chemical Processes

The optimization of non-linear constraint problems is relevant to chemical engineering practice (Floudas 1995). Non-linearities are introduced by process equipment design relations, by equilibrium relations and by combined heat and mass balances. There are many chemical processes which are highly nonlinear and complex with reference to optimal operating conditions with many equality and inequality constraints. In this paper the following processes are considered for applying DE and MDE: (1) Optimal operation of alkylation unit, and (2) Heat exchanger network design (3) Reactor network design (4) Dynamic optimization of a batch reactor.

5.1 Optimal Operation of Alkylation Unit

Alkylation process is common in the petroleum industry. A simplified process flow diagram of an alkylation process is shown in Fig. 1. The process model was described and solved in literature (Sauer et al. 1964) using successive linear programming. The process model seeks to determine the optimum set of operating conditions for the process, based on a mathematical model, which allowed maximization of profit. Bracken and McCormick (1968) formulated the problem as a direct nonlinear programming model with mixed nonlinear inequality and equality constraints and a nonlinear profit function to be maximized. They used Sequential Unconstrained Minimization Technique (SUMT) for solving the same. Later, Dembo (1976) transformed the NLP problem with ten variables which Bracken and McCormick (1968) derived, into a problem with seven variables. All equality constraints are eliminated and the problem has been formulated as a signomial optimization problem. This problem involves seven variables subject to twelve nonlinear and two linear inequality constraints. Edger and Himmelblau (1989) used sequential quadratic programming to solve the problem as formulated by Bracken and McCormick (1968). Maranas and Floudas (1997) used generalized geometric programming to solve the seven variables problem as formulated by Dembo (1976). Adjiman et al. (1998) used αBB algorithm (for general twice –differentiable constraint NLPs) for solving this problem.

As shown in Fig. 1, an olefin feed (100% butane), a pure isobutane recycle and a 100% isobutane make-up stream are introduced in a reactor together with an acid catalyst. The reactor product stream is then passed through a fractionator where the isobutane and the alkylate product are separated. The spent acid is also removed from the reactor. The variables are defined as shown in Table-1 along with the upper and lower bounds on each variable. The bounds represent economic, physical and performance constraints. In the present study, the problem formulation is same as that of (Maranas and Floudas 1997; Adjiman et al. 1998). The problem is briefly discussed below.

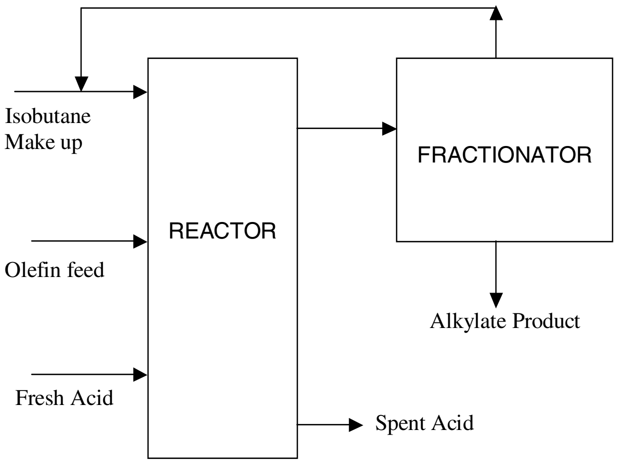


Fig. 1. Simplified Alkylation Process Flow sheet

Table 1. Variables and their bounds

Symbol	Variable	Lower Bound	Upper Bound
x_1	Olefin feed rate (barrels/day)	1500	2000
x_2	Acid addition rate (thousands of pounds/day)	1	120
x_3	Alkylate yield (barrels/day)	3000	3500
x_4	Acid strength (wt. %)	85	93
x_5	Motor octane no.	90	95
x_6	External Isobutane-to-olefin Ratio	3	12
x_7	F-4 performance no.	145	162

5.1.1 Profit Function

The objective is to improve the octane number of some olefin feed by reacting it with isobutane in the presence of acid. The product of the reaction is distilled and the un-reacted is recycled back to the reactor. The objective function was defined in terms of alkylate product, or output value minus feed and recycle costs. Operating costs were not reflected in the function. The total profit ($ per day), to be maximized (Adjiman et al. 1998), is given as follows:

$$\text{Max.} \quad \text{Profit} = 1.715x_1 + 0.035x_1x_6 + 4.0565x_3 + 10.0x_2 - 0.063x_3x_5$$

Subject to the following constraints:

$$0.0059553571x_6^2x_1 + 0.88392857x_3 - 0.1175625x_6x_1 - x_1 \le 0$$
$$1.1088x_1 + 0.1303533x_1x_6 - 0.0066033x_1x_6^2 - x_3 \le 0$$
$$6.66173269x_6^2 + 172.39878x_5 - 56.596669x_4 - 191.20592x_6 - 10000 \le 0$$
$$1.08702x_6 + 0.32175x_4 - 0.03762x_6^2 - x_5 + 56.85075 \le 0$$
$$0.006198x_7x_4x_3 + 2462.3121x_2 - 25.125634x_2x_4 - x_3x_4 \le 0$$
$$161.18996x_3x_4 + 5000.0x_2x_4 - 489510.0x_2 - x_3x_4x_7 \le 0$$
$$0.33x_7 - x_5 + 44.333333 \le 0$$
$$0.022556x_5 - 0.007595x_7 - 1.0 \le 0$$
$$0.00061x_3 - 0.0005x_1 - 1.0 \le 0$$
$$0.819672x_1 - x_3 + 0.819672 \le 0$$
$$24500.0x_2 - 250.0x_2x_4 - x_3x_4 \le 0$$
$$1020.4082x_4x_2 + 1.2244898x_3x_4 - 100000x_2 \le 0$$
$$6.25x_1x_6 + 6.25x_1 - 7.625x_3 - 100000 \le 0$$
$$1.22x_3 - x_6x_1 - x_1 + 1.0 \le 0$$

The maximum profit as reported in Adjiman *et al.* (1998) is: \$1772.77 per day, and the optimal variable values are $x_1 = 1698.18$, $x_2 = 53.66$, $x_3 = 3031.3$, $x_4 = 90.11$, $x_5 = 95.0$, $x_6 = 10.50$, $x_7 = 153.53$.

5.1.2 Results and Discussion

Table-2 presents the comparison of results obtained in earlier studies and, those obtained using DE and MDE in present study. These results cannot be compared because of the different platform used for computation. But, there is a variation in value of objective function obtained using different optimization methods. The optimal solution obtained using DE is: $x_1 = 1698.256922$; $x_2 = 54.274463$; $x_3 = 3031.357313$; $x_4 = 90.190233$; $x_5 = 95.0$; $x_6 = 10.504119$; $x_7 = 153.535355$; and the value of objective function is 1766.36. This solution satisfies all the inequality constraints to six decimal places (i.e., 0.0000001) while solution reported by Maranas and Floudas (1997) and Adjiman et al. (1998) violates the first, third and sixth constraints. The values of first, third and sixth constraints are found to be 0.016501, 4.752125, and 1727.867362 respectively instead of zero or less than zero. Hence the better solution (possibly global optimum) is 1766.36 that satisfy all the constraints to at least six decimal places.

Table 2. Comparison of Various Methods

S. No.	Method	Objective function (f)	CPU-time (s)
1	SUMT (Bracken and McCormick 1968)	1769	Not reported
2	NPSOL (Edger and Himmelblau, 1989)	1768.75	Not reported
3	GGP in GAMS (Maranas and Floudas, 1997)	1773	30[1]
4	αBB algorithm (Adjiman et al., 1998)	1772.77	13.6[2]
5	Differential Evolution (present study)	1766.36	5.77[3]
6	Modified Differential Evolution (present study)	1766.36	5.43[4]

[1] CPU-time on HP-730 workstation.
[2] CPU-time on HP9000/730 using scaled Gerschgorin theorem method.
[3] CPU-time on PC with Pentium PIII, 500 MHz/128 MB RAM/ 10 Gb HD (strategy DE/rand/1/bin).
[4] CPU-time on PC with Pentium PIII, 500 MHz/128 MB RAM/ 10 Gb HD (strategy DE/rand/1/bin).

Table-3 shows the results obtained using DE and MDE and their comparison in terms of the number of objective function evaluations, CPU-time and the percentage of convergencies to the optimum, i.e., success rate. The termination criterion used is an accuracy value of 1×10^{-6}. *NFE, NRC* and CPU-time in Table-3 represents, respectively the mean number of objective function evaluations over all the 10 experiments (with different seed values), success rate, and the average CPU time per experiment. The key parameters used are *NP* = 10*D*, *CR* = 0.8, *F* = 0.5. Both DE and MDE are

Table 3. Results of DE and MDE for alkylation Problem

Method	NFE	NRC (%)	CPU-time (s)
DE	114895	100	5.77
MDE	108103	100	5.43

able to locate the global optimum in all the experiments, as *NRC* is 100%. MDE takes 5.9% less CPU-time than DE. Results in Table-3 clearly indicate that MDE is faster than DE and takes less CPU-time to locate the global optimum solution.

5.2 Heat Exchanger Network Design (HEND)

This problem addresses the design of a heat exchanger network as shown in Fig. 2. It has been taken from Floudas & Pardalos (1990). Also, it has been solved by Adjiman et al. (1998) using αBB -Algorithm. One cold stream must be heated from 100 ^{0}F (37.78 ^{0}C) to 500 ^{0}F (260 ^{0}C) using three hot streams with different inlet temperatures. The goal is to minimize the overall heat exchange area.

$$\text{Min} f = x_1 + x_2 + x_3$$

Subject to

$$0.0025(x_4 + x_6) - 1 = 0$$

$$0.0025(- x_4 + x_5 + x_7) - 1 = 0$$

$$0.01(- x_5 + x_8) - 1 = 0$$

$$100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0$$

$$x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0$$

$$x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0$$

$$100 \leq x_1 \leq 10000$$

$$1000 \leq x_2, x_3 \leq 10000$$

$$10 \leq x_4, x_5, x_6, x_7, x_8 \leq 1000$$

The global optimum is: $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8; f)$ = (579.19, 1360.13, 5109.92, 182.01, 295.60, 217.9, 286.40, 395.60; 7049.25).

The above problem can be reformulated by eliminating equality constraint as given below:

$$\text{Min} \quad f = x_1 + x_2 + x_3$$

Subject to

$$100x_1 - x_1(400 - x_4) + 833.33252x_4 - 83333.333 \leq 0$$

$$x_2 x_4 - x_2(400 - x_5 + x_4) - 1250x_4 + 1250x_5 \le 0$$

$$x_3 x_5 - x_3(100 + x_5) - 2500x_5 + 1250000 \le 0$$

$$100 \le x_1 \le 10000$$

$$1000 \le x_2, x_3 \le 10000$$

$10 \le x_4, x_5 \le 1000$; Global optimum is same after reformulation.

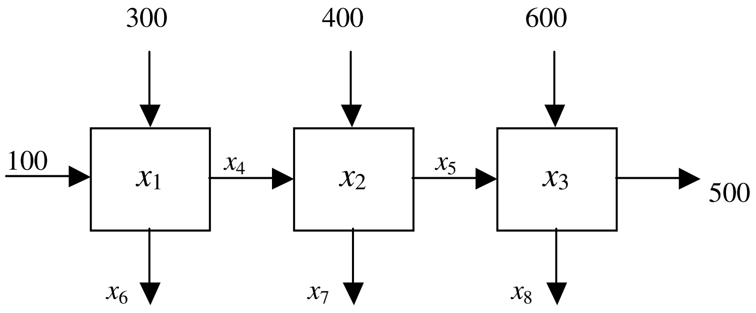


Fig. 2. Heat exchanger network design problem

5.2.1 Results and Discussion

Table-4 shows the results obtained using DE with variable violations relocating to boundary/relocating to interior, and the comparison of DE performance with that of αBB algorithm. The stopping criteria adopted for DE is to terminate the search process when one of the following conditions is satisfied: (1) the maximum number of generations is reached (assumed 5000 generations), (2) $|f^k_{\max} - f^k_{\min}| < 10^{-5}$ where f is the value of objective function for k-th generation. In Table-4 & 5, NFE^{100} & NRC^{100} represent respectively, the mean number of objective function evaluations and the percentage of runs converged to the global optimum in all the 100 executions (with different seed values).

In this problem, NFE^{100} in case of relocating to interior method is 2.68% less than NFE^{100} with relocating to boundary method (Table-4). Also, the NRC^{100} is 100% and 89% respectively for relocating to interior and relocating to boundary methods. Convergence tolerance in the present case is 10^{-5} as compared to 10^{-3} in Adjiman et al. (1998).

The time taken by DE is much less than that of αBB-Algorithm. Of course the CPU-times cannot be compared directly because different computers are used. However, a comparison can be made after considering a factor of 10 (high enough) i.e. had the same problem would have been solved on HP9000/730 (66MHz) using DE it might have taken ten times more of CPU-time than by Pentium-III, 500MHz. As we know processor speed of HP9000/730 machine is 66 MHz while for P-III used in present study is 500 MHz, therefore a factor of 10 seems to be appropriate for comparison purpose.

Even then the CPU-time using DE is 73.44% less for the HEND problem and 92% less for RND problem than that in αBB-Algorithm respectively (Table-4).

Table-5 shows the results obtained using MDE (relocating to boundary/relocating to interior) and its comparison to DE. The parameters used are CR = 0.8, F = 0.5 and NP = 10D. The performance of MDE, as is evident from results presented in Table-5, is better than that of DE (a saving of 16.32% CPU-time). The reliability of DE and MDE is nearly same as indicated by NRC100, which is same in both the cases (100% for relocating to interior and 88% for relocating to boundary method). The convergence history of HEND problem is shown in Fig. 3. It represents the averaged results of 100 experiments. The error obtained after 200 generations is 0.986 for MDE while it is 2.344 for DE. MDE is able to locate the global optimal solution for the test problem considered faster than DE.

The results of 100 different experiments are plotted in Fig. 4 as fractional differences of optimal objective function value from its best-found optimal value (in all 100

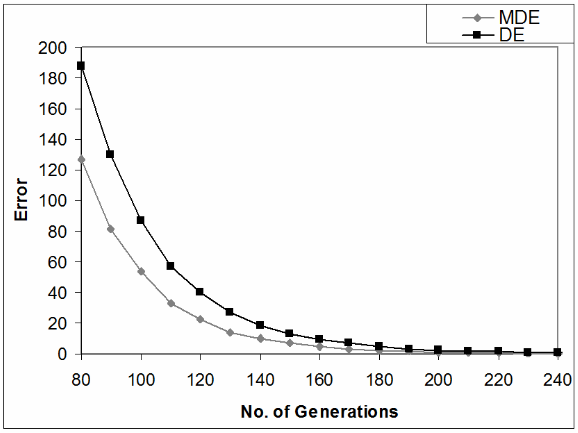


Fig. 3. Convergence History of HEND Problem

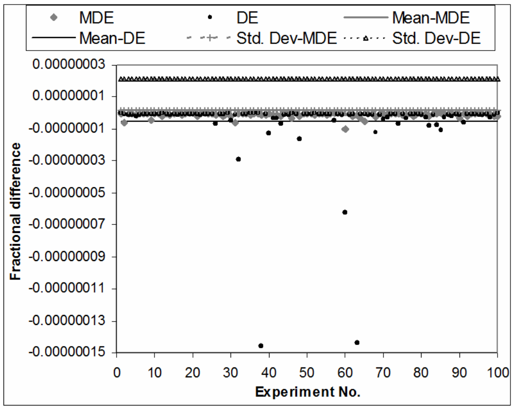


Fig. 4. Fractional difference variation for HEND Problem

experiments) versus experiment number. Thus, in the figure, a point closer to zero value of d_f denotes a more accurate result. The best optimal value (f_{best}) of 7049.248022 was obtained from the 100 experiments using DE and MDE algorithms. The overall accuracy of the 100 values of d_f is given by low mean value of -1.16×10^{-9} and -5.61×10^{-9} respectively for MDE and DE. The precision of 100 values of d_f is expressed in terms of low standard deviation of 1.45×10^{-9} and 2.12×10^{-9} respectively for MDE and DE.

5.3 Reactor Network Design (RND)

This example, taken from Ryoo and Sahinidis (1995), is a reactor network design problem, describing the system shown in Fig. 5. It involves the design of a sequence of two CSTR reactors where the consecutive reaction $A \rightarrow B \rightarrow C$ takes place. The goal is to maximize the concentration of product B ($x_4 = C_{B2}$) in the exit stream. This problem is known to have caused difficulties for other global optimization methods.

Min $f = -x_4$

Subject to

$$x_1 + k_1x_2x_5 = 1$$

$$x_2 - x_1 + k_2x_2x_6 = 0$$

$$x_3 + x_1 + k_3x_3x_5 = 1$$

$$x_4 - x_3 + x_2 - x_1 + k_4x_4x_6 = 0$$

$$x_5^{0.5} + x_6^{0.5} \le 4$$

$$(0, 0, 0, 0, 10^{-5}, 10^{-5}) \le (x_1, x_2, x_3, x_4, x_5, x_6) \le (1, 1, 1, 1, 16, 16).$$

Where $k_1 = 0.09755988$

$k_2 = 0.99k_1$

$k_3 = 0.0391908$

$k_4 = 0.9k_3$

The global optimum is: (x_1, x_2, x_3, x_4, x_5, x_6; f) = (0.771462, 0.516997, 0.204234, 0.388812, 3.036504, 5.096052; -0.388812).

Ryoo and Sahinidis (1995) solved globally the same problem by convex lower bounding the bilinear terms for a different set of reaction constants. A number of variations of the main algorithm were considered and the best one yielded a CPU-time of 23s on SPARC2. Maranas and Floudas (1997) used geometric programming approach for the above problem and obtained the global optimum after 299 iterations and 20s of CPU-time.

This example constitutes a very difficult test problem, as it possesses a local minimum with an objective function value that is very close to that of the global

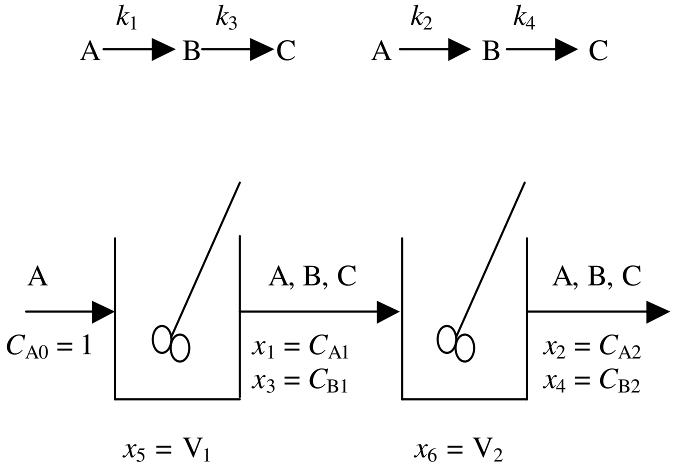


Fig. 5. Reactor network design problem

solution. The local solutions are with f = -0.37461 and f = -0.38808. Interestingly enough, the two local solutions utilize only one of the two reactors whereas the global solution makes use of both reactors.

This problem can be reformulated by eliminating equality constraint as follows:

$$\text{Max } f = \frac{k_2x_6(1+k_3)+k_1(1+k_2x_6)}{(1+k_1x_5)(1+k_2x_6)(1+k_3x_5)(1+k_4x_6)}$$

$$\text{Subject to} \qquad x_5^{0.5}+x_6^{0.5} \le 4$$

$$(10^{-5}, 10^{-5}) \le (x_5, x_6) \le (16, 16).$$

Global optimum is same after reformulation.

5.3.1 Results and Discussion

RND is a difficult problem as mentioned already having two local optima near global optimum. Still MDE and DE are able to locate the global optimum although the success rate is 44 to 57% (Table-5). For RND problem, the first termination condition is the same as above but second condition is $|f_{\max}^k - f_{\min}^k| < 10^{-6}$. In this problem, NFE^{100} with relocating to boundary method is 2.87% and 9.33% (for MDE and DE respectively) more than NFE^{100} for relocating to interior method (Table-5). It is important to note that in this problem, NRC^{100} with relocating to boundary method is just 8% and 10% respectively for MDE and DE (Table-5). It is because when the upper limit of variable is violated, the value of variable is relocated to the upper limit that resulted in convergence to non-optimal solution. This happened as one of the local solutions near to global optimum is lying on the bound, and hence trapped at local optimum.

Further, in order to enhance the robustness of the DE and MDE algorithms, *CR* value is increased. And it is found that *CR* alone is not affecting the value of *NRC*. An increase in F value significantly affects the NRC value. Table-6 shows the comparison of DE, and MDE algorithm for the increased values of CR and F. Also, it has been found out that at $CR = 1.0$ and $F = 0.9$, both DE and MDE are able to achieve almost 100% NRC^{100} although the computational time is 46% and 53% more than that shown in Table-5 respectively for DE (RI) and MDE (RI) algorithms. Further, it is clear from Table-6 that MDE is taking 13.33% less CPU-time as compared to DE. Therefore, MDE seems to be computationally efficient.

The results of 100 different experiments carried out are plotted in Fig. 6 as fractional differences of optimal objective function value from its best-found optimal value (in all 100 experiments) versus experiment number. Thus, in the figure, a point closer to zero value of fractional difference denotes a more accurate result. The best optimal value (f_{best}) of 0.388811 was obtained from the 100 experiments using DE and MDE algorithms. The overall accuracy of the 100 values of d_f is given by low mean value of 5.97×10^{-7} and 4.89×10^{-7} respectively for MDE and DE. The precision of 100 values of d_f is expressed in terms of low standard deviation of 2.00×10^{-6} and 1.30×10^{-6} respectively for MDE and DE.

Table 4. Results of DE and its comparison with αBB Algorithm

Problem	DE CPU-time^{100}/ NFE^{100}/NRC^{100} (RB)	DE CPU-time^{100}/ NFE^{100}/NRC^{100} (RI)	Key Parameters of DE (NP/CR/F)	αBB Algorithm (CPU-time)
HEND	1.513**/38824/89	1.477**/37810/100	50/0.8/0.5	54.4* s
RND	0.049**/1605/10	0.041**/1468/57	20/0.8/0.5	5.5* s

* CPU-time obtained using HP9000/730 (66 MHz) with convergence tolerance of 0.001 (Adjiman et al., 1998).
** CPU-time obtained using Pentium-III (500 MHz) with convergence tolerance of 0.00001 (present study).

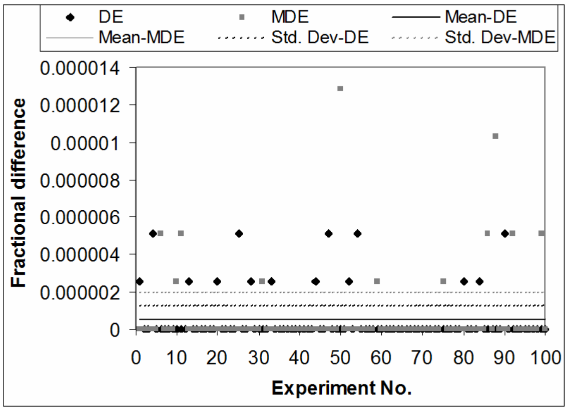


Fig. 6. Fractional difference variation for Reactor Network Design Problem

Table 5. Results of MDE** and its comparison with DE**

Problem	MDE (RB)	MDE (RI)	DE (RB)	DE (RI)
HEND	1.292/33146/88	1.236/31877/100	1.513/38824/89	1.477/37810/100
RND	0.041/1289/08	0.034/1253/44	0.049/1605/10	0.041/1468/57

** CPU-time^{100}/*NFE*^{100}/*NRC*^{100} obtained using Pentium-III (500 MHz).

Table 6. Comparison of DE, and MDE for RND problem

Methods	CPU-time^{100} (s)*	NFE^{100}/ NRC^{100}	Key Parameters (CR/F)
DE (RI)	0.060	2074/100	1.0/0.9
MDE (RI)	0.052	1860/99	1.0/0.9

*CPU-time obtained using Pentium-III (500 MHz).

5.4 Dynamic Optimization of a Batch Reactor

In this problem, we consider the consecutive reaction: $A \xrightarrow{k_1} B \xrightarrow{k_2} C$ in a batch reactor (Ray, 1981). The objective is to obtain the optimal reactor temperature progression, which maximizes the intermediate product *B* for a fixed batch time. The dynamics are given by the following design equations:

$$\frac{dC_A}{dt} = -k_1 C_A^2 ;$$

$$\frac{dC_B}{dt} = k_1 C_A^2 - k_2 C_B ;$$

Where $k_1 = 4000 * \exp\left(\frac{-2500}{T}\right)$ $\qquad$ $k_2 = 620000 * \exp\left(\frac{-5000}{T}\right)$

and $298\,\text{K} \le T \le 398\,\text{K}$ **;** with the initial conditions C_A (0) = 1.0, C_B (0) = 0.0, and the objective function is *f* = Maximize C_B. We need to find out optimal temperature profile, which gives maximum intermediate product concentration. This problem has been solved by Renfro et al. (1987) using piecewise constant controls. Dadebo and Mcauley (1995) used dynamic programming for solving this problem and reported results for different number of stages.

In this problem our parameters are temperatures at different time intervals. With these temperatures we need to find out the optimal final concentration of *B* (by solving the above model equations along with DE and MDE).

5.4.1 Results and Discussion

In this problem, we need to find out optimal temperature profile, which gives maximum intermediate product concentration. This problem has been solved by Renfro et al. (1987) using piecewise constant controls. They reported a value of 0.61 for the objective function. Logsdon and Biegler (1989) obtained a value of 0.610767. Dadebo and Mcauley (1995) used dynamic programming for solving this problem. They reported results for different number of stages. Dadebo and Mcauley (1995) reported a yield of 0.610070 for 10 stages which is same as shown in Table-7 for DE and MDE. In the Table-7, CPU-time[20] is average of 20 executions. It is evident that MDE takes about 13.7% (20 s) of CPU-time less than that of DE. Also, it is to be noted that time saving is highly desirable in this type of problems, as though the percentage is 13.7 which appears to be less, but the amount is 20s, which is significant amount as compared to other problems where the saving is of order of 1.0s or less. Also, the *NRC*[20] is 100% using both MDE and DE algorithms.

The results of the 100 different experiments are plotted in Fig. 7 as fractional differences of optimal objective function value from its best-found optimal value (in all 100 experiments) versus experiment number. Thus, in the figure, a point closer to zero value of fractional difference denotes a more accurate result. The best optimal value (f_{best}) of 0.610079 was obtained from the 100 experiments using DE and MDE algorithms. The overall accuracy of the 100 values of d_f is given by low mean value of 0.0 and 0.0 respectively for MDE and DE. The precision of 100 values of d_f is expressed in terms of low standard deviation of 0.0 and 0.0 respectively for MDE and DE.

Fig. 8 shows the optimal temperature profile obtained using DE and MDE for 10 intervals of total time (i.e., for 10 stages). Both the profiles are exactly same. Having obtained the same profile in lesser CPU-time establishes the fact that MDE is able to find the optimal temperature profile faster than DE.

Table 7. Results of DE and MDE for 10 intervals

Methods	Yield	Optimal Initial Temperature T (0)	CPU-time[20] (s)*	NRC[20] (%)
DE	0.610079	361.4	151.28	100
MDE	0.610079	361.4	130.55	100

* Pentium -4/2.4 GHz/256 MB RAM using RK Method of 4th order with step size of 0.0001.

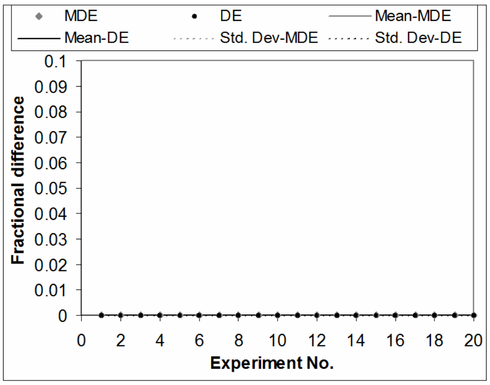


Fig. 7. Fractional difference variation for Dynamic Optimization of a Batch Reactor Problem

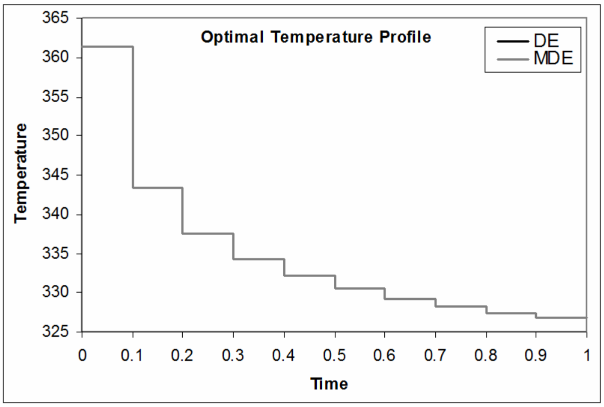


Fig. 8. Optimal Temperature Profile for 10 intervals

6 Multi-Objective Optimization

Multi-objective optimization has created immense interest in engineering in the last two decades. Most of the real world optimization problems have more than one objective to be optimized, and hence they are called Multi-objective Optimization Problems (MOOPs). Ideally, MOOPs require multiple trade-off solutions (a set of Pareto optimal solutions) to be found. The presence of multiple conflicting objectives makes the problem interesting to solve. Due to conflicting objectives, no single solution can be termed as an optimum solution. Therefore, the resulting multi-objective optimization problem resorts to a number of trade-off optimal solutions.

Classical techniques can at best find one solution in a single run, on the other hand evolutionary algorithms can find multiple optimal solutions in a single run due to their population based search approach. This characteristic enables the search for an entire set of Pareto optimal solutions in a single run. A detailed account of multi-objective optimization using evolutionary algorithms and some of the applications of Multi-objective Differential Evolution (MODE) algorithms can be found in literature (Deb 2001; Babu and Anbarsu 2005; Babu et al. 2005a, 2005b; Gujarathi and Babu 2005, 2006).

In recent years, many evolutionary algorithms for multiobjective optimization have been introduced. Among these, the NSGA-II by Deb et al. (2002) and SPEA2 by Zitzler et al. (2001) are the most popular. A particular EA that has been used for multiobjective optimization is Differential Evolution (DE). DE is a simple yet powerful evolutionary algorithm given by Price and Storn (1997) that has been successfully used in solving single-objective optimization problems. Hence, several researchers have tried to extend it to solve MOOPs. Abbass et al (2001) first extended DE to solve MOOPs and proposed Pareto Differential Evolution (PDE) algorithm. PDE was compared to SPEA (the predecessor of SPEA2) on two test problems and found to outperform it. Madavan (2002) achieved good results with the Pareto Differential Evolution (PDE). Babu et al (2005a, 2005b) also studied the extension of DE to solve MOOPs in continuous domain and proposed multi-objective differential evolution (MODE), nondominated sorting differential evolution (NSDE) and modified nondominated sorting differential evolution (MNSDE). In this chapter, MODE, NSDE and MNSDE are discussed briefly.

6.1 Multi-Objective Differential Evolution (MODE)

Babu and his co-workers (Babu et al. 2005a, 2005b) introduced multi-objective differential evolution (MODE) for solving MOOPs. MODE is an extension of Differential Evolution (DE). Successful application of MODE algorithm is made to carry out the multi-objective optimization of Styrene Reactor. Five combinations of the objectives are considered. Pareto set (a set of equally good solutions) obtained for all the cases was compared with results reported using non-dominated sorting genetic algorithm (NSGA). The results show that all objectives besides profit can be improved compared to those reported using NSGA and current operating conditions.

In the multi-objective optimization of styrene reactor, an adiabatic reactor with plug flow and radial uniformity was considered for simulation. Sheel and Crowe (1969) considered a pseudo-homogeneous model with axial dispersion ignored, and any limitations of mass or heat transfer to the catalyst pellet or diffusion within the pellet were lumped into the rate constant. The models developed on this basis are only valid for the particle size and operating conditions used on obtaining the data, and hence are not valid over a wide range of design and operating conditions. Later, Abdalla et al. (1994) and Elnashaie et al. (1993) used this model as well as a more detailed heterogeneous model, which takes into account diffusion in the catalyst pellet. Yee et al. (2003) showed that the predictions by both the models are comparable, but pseudo-homogeneous model took significantly less computational time than heterogeneous model. Hence for this study pseudo-homogeneous model is selected for simulating the industrial adiabatic reactor, whose design and operating conditions were reported by Sheel and Crowe (1969).

Parametric study of MODE algorithm on benchmark test problems is also carried out by Babu et al. (2007a). Babu et al. (2007b) also carried out multi-objective optimization of polyethylene terepthalate (PET) reactors. PET is one of the most important thermoplastic engineering polymers and is manufactured commercially by semi-batch or continuous process. They are extensively used in the manufacture of fibres, films, bottles, etc. Optimal design and operation of the wiped PET reactor are required, as it is the critical equipment in PET manufacturing process.

Modeling of PET reactor was done using pre-validated model and simulation runs were carried out using ODE23S subroutine (MATLAB Library). Simulations were carried out for the input conditions and parameters available in literature. The results were found to be identical to those reported by these workers. The equations were solved manually using Euler's method for a few iterations. The results thus obtained match well with the values provided with the computer code. Based on the simulation results the following significant points were observed:

- The degree of polymerization (DP) is found to increase along the dimensionless distance. The PET formed is also found to increase along the length.
- The profiles of acid end group concentration, vinyl end group concentration, and other by products produced shows a decreasing trend.
- The simulations were carried for three sets of data. The results are helpful to tune the model. Results of all the three sets match exactly with those reported in the literature further confirming the validity of the code.

Several cases of industrial importance of multi-objective optimization problems for PET reactors were formulated, and then solved by MODE. The results are found to be extremely encouraging as smooth and wide spread Pareto set was obtained and not a single solution as reported by Bhaskar et al. (2001) using NSGA. The Pareto set and optimal operating conditions for the first three combinations are same. The trend of decision variables at the optimum in all cases can be explained qualitatively, which shows that the multi-objective optimization results obtained by MODE are reliable.

6.2 Non-dominated Sorting Differential Evolution (NSDE)

NSDE algorithm is a simple extension of DE for solving multi-objective optimization problems. The working of NSDE and DE is similar except the selection operation that is modified in order to solve the multi-objective optimization problems. The details of the NSDE algorithm is as follows (Angira and Babu 2005b):

First of all set the set the key parameters, i.e., *CR* - crossover constant, *F* - scaling factor, *NP* - population size, *Max_gen* – maximum number of generations of NSDE algorithm. And then randomly initialize the population points within the bounds of decision variables. After initialization of population, randomly choose three mutually different vectors for mutation and crossover operation (as is done in DE algorithm) to generate trial vector. Evaluate the trial and target vector and perform a dominance check. If trial vector dominates the target vector, the trial vector is copied into the population for next generation otherwise target vector is copied into population for next generation. This process of mutation, crossover, and dominance check is repeated for specified number of generations. Evaluate and then sort this final population to obtain the non-dominated solutions. Sorting can be done using any of the standard approaches reported in Deb (2001). Angira (2005) and Angira & Babu (2005a) used naïve and slow approach. In this approach, each solution *i* is compared with every other solution in the population to check if it is dominated by any solution in the population. If no solution is found to dominate solution *i*, it is member of the non-dominated set otherwise it does not belong the non-dominated set. This is how any other solution in the population can be checked to see if it belongs to the non-dominated set. More details with application to test problems are given in Angira (2005), Angira and Babu (2005a) and Angira and Babu (2006).

The stopping criteria for the algorithm can be any one of the following conditions:

(a). There is no new solution added to the non-dominated front for a specified number of generations.
(b). Till the specified number of generations.

However, in this study, the second condition is used as termination criterion.

6.3 Modified Non-dominated Sorting Differential Evolution (MNSDE)

In the previous section, it has been found that MDE took less computational time due to the use of single array of population. In this section, MDE is extended for solving multi-objective optimization problems and the extended algorithm is called as MNSDE (Modified Non-dominated Sorting Differential Evolution). MNSDE is similar to NSDE except for the selection criterion (Angira 2005). Also, MNSDE maintains only one set of population as against two sets in NSDE. The selection criterion used in MNSDE is different from that of NSDE and is as follows:

After mutation & crossover the trial solution is generated. Selection is made between this trial solution and target solution. If trial solution dominates the target solution, then the target solution is replaced by the trial solution in the population of current generation itself otherwise the target solution is kept as it is. The remaining procedure is same as that of NSDE. The use of single array of population in MNSDE as against two in NSDE may lead to reduction in memory and computational efforts required as is found for MDE.

Both NSDE and MNSDE are used to solve non-linear and complex test problems (Angira 2005; Babu 2007; Angira and Babu 2005a, 2006). Also, the effect of various parameters have been discussed and analyzed. Angira and Babu (2006) found number of Pareto set solution to be increasing with maximum number of generations up to a certain value which is problem dependent. It is reported that a high value of CR (0.7 or more) is suitable and scaling factor is found to affect not only number of solutions in Pareto set but also the shape and spread of Pareto optimal front. Based on the results, Angira and Babu (2006) recommended to use a high CR value for both NSDE & MNSDE and lower value of scaling factor for MNSDE and a value of 0.5 for MNSDE. Maximum number of generation is found to be problem dependent.

7 Conclusions

The Modified Differential Evolution (MDE) algorithm has been compared to Differential Evolution (DE) for global optimization of benchmark test functions and selected nonlinear chemical processes. Extensive computational comparisons have been made for all the chemical engineering problems considered using standard statistical hypothesis testing methods such as t-test. The results stated above clearly show the improvement upon the performance characteristics of DE with regard to the number of function evaluations (*NFE*)/CPU-time required to find the global optimum. The enhancement was accomplished by using single array in MDE as compared to two arrays in DE. The second method, i.e., relocating to interior is found to be better than the relocating to boundary, as it avoids trapping at local optimal solution if they are present at extreme (upper or lower bound). RND problem is an example of such a situation. The Differential Evolution algorithm has also been extended to solve multi-objective optimization problems. Three modified versions (MODE, NSDE, and MNSDE) have been proposed and applied successfully on various bench mark test functions and two industrial multi-objective optimization problems for producing Styrene and PET.

References

Abbass, H.A., Sarkar, R., Newton, C.: PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of IEEE congress on Evolutionary Computation, pp. 971–978 (2001)

Abdalla, B.K., et al.: Intrinsic kinetics and industrial reactors modeling for the dehydrogenation of ethyl benzene to styrene on promoted iron oxide catalysts. Applied Catalysis A: General 113, 89–102 (1994)

Adjiman, C.S., et al.: A global optimization method for general twice – differentiable constrained NLPs – I. Theoretical Advances. Computers and Chemical Engineering 22, 1137–1158 (1998)

Angira, R.: Evolutionary Computation for Optimization of Selected Non-linear Chemical Processes. Ph.D. thesis, BITS-Pilani, India (2006)

Angira, R., Babu, B.V.: Optimization of Non-Linear Chemical Processes Using Modified Differential Evolution (MDE). In: Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, India, pp. 911–923 (2005a)

Angira, R., Babu, B.V.: Non-dominated Sorting Differential Evolution (NSDE): An Extension of Differential Evolution for Multi-objective Optimization. In: Proceedings of The 2nd Indian International Conference on Artificial Intelligence, Pune, India, pp. 1428–1443 (2005b)

Angira, R., Babu, B.V.: Process Synthesis and Design Using Modified Differential Evolution (MDE). In: Proceedings of International Symposium & 58th Annual Session of IIChE in association with International Partners, New Delhi (2005c)

Angira, R., Babu, B.V.: Optimization of Process Synthesis and Design Problems: A Modified Differential Evolution Approach. Chemical Engineering Science 61, 4707–4721 (2006a)

Angira, R., Babu, B.V.: Multi-Objective Optimization using Modified Differential Evolution (MDE). International Journal of Mathematical Sciences: Special Issue on Recent Trends in Computational Mathematics and Its Applications 5, 371–387 (2006b)

Angira, R., Babu, B.V.: Performance of Modified Differential Evolution for Optimal Design of Complex and Non-Linear Chemical Processes. Journal of Experimental & Theoretical Artificial Intelligence 18, 501–512 (2006c)

Babu, B.V.: Process Plant Simulation. Oxford University Press, New York (2004)

Babu, B.V.: Improved Differential Evolution for Single- and Multi-Objective Optimization: MDE, MODE, NSDE, and MNSDE. In: Deb, K., et al. (eds.) Advances in Computational Optimization and its Applications, pp. 24–30. Universities Press, Hyderabad (2007)

Babu, B.V., Sastry, K.K.N.: Estimation of Heat-transfer Parameters in a Trickle-bed Reactor using Differential Evolution and Orthogonal Collocation. Computers and Chemical Engineering 23, 327–339 (1999)

Babu, B.V., Gaurav, C.: Evolutionary Computation Strategy for Optimization of an Alkylation Reaction. In: Proceedings of International Symposium & 53rd Annual Session of IIChE, Calcutta (2000)

Babu, B.V., Munawar, S.A.: Differential Evolution for the Optimal Design of Heat Exchangers. In: Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, Bhuvaneshwar (2000)

Babu, B.V., Singh, R.P.: Synthesis & optimization of Heat Integrated Distillation Systems Using Differential Evolution. In: Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, Bhuvaneshwar (2000)

Babu, B.V., Angira, R.: Optimization of Non-linear functions using Evolutionary Computation. In: Proceedings of 12th ISME Conference on Mechanical Engineering, Chennai, pp. 153–157 (2001a)

Babu, B.V., Angira, R.: Optimization of Thermal Cracker Operation using Differential Evolution. In: Proceedings of International Symposium & 54th Annual Session of IIChE, Chennai (2001b)

Babu, B.V., Angira, R.: Optimization of Water Pumping System Using Differential Evolution Strategies. In: Proceedings of The Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems, Singapore (2003)

Babu, B.V., Angira, R.: Optimal Design of an Auto-thermal Ammonia Synthesis Reactor. Computers and Chemical Engineering 29, 1041–1045 (2005)

Babu, B.V., Anbarasu, B.: Multi-Objective Differential Evolution (MODE): An Evolutionary Algorithm for Multi-Objective Optimization Problems (MOOPs). In: Proceedings of The Third International Conference on Computational Intelligence, Robotics, and Autonomous Systems, Singapore (2005)

Babu, B.V., Angira, R.: Modified Differential Evolution (MDE) for Optimization of Non-Linear Chemical Processes. Computers and Chemical Engineering 30, 989–1002 (2006)

Babu, B.V., Mubeen, J.H.S., Chakole, P.G.: Multiobjective Optimization Using Differential Evolution. TechGenesis-The Journal of Information Technology 2, 4–12 (2005a)

Babu, B.V., Chakole, P.G., Mubeen, J.H.S.: Multiobjective Differential Evolution (MODE) for Optimization of Adiabatic Styrene Reactor. Chemical Engineering Science 60, 4822–4837 (2005b)

Babu, B.V., Chakole, P.G., Mubeen, J.H.S.: Differential Evolution Strategy for Optimal Design of Gas Transmission Network. Journal of Multidisciplinary Modeling in Materials and Structures 1, 315–328 (2005c)

Babu, B.V., et al.: Strategies of Multi-Objective Differential Evolution (MODE) for Optimization of Adiabatic Styrene Reactor. In: Proceedings of International Conference on Emerging Mechanical Technology-Macro to Nano, BITS-Pilani, pp. 243–250 (2007a)

Babu, B.V., Mubeen, J.H.S., Chakole, P.G.: Simulation and Optimization of Wiped Film Poly Ethylene Terephthalate (PET) Reactor using Multiobjective Differential Evolution (MODE). Materials and Manufacturing Processes: Special Issue on Genetic Algorithms in Materials (in press, 2007b)

Bergey, P.K., Ragsdale, C.: Modified differential evolution: a greedy random strategy for genetic recombination. Omega 33, 255–265 (2005)

Bhaskar, V., Gupta, S.K., Ray, A.K.: Multi-objective optimization of an industrial wiped film PET reactor. AIChE J. 46, 1046–1048 (2001)

Bracken, J., McCormick, J.P.: Selected Applications of Nonlinear Programming. John Wiley & Sons Limited, New York (1968)

Chiou, J.P., Wang, F.S.: Hybrid Method of Evolutionary Algorithms for Static and Dynamic Optimization Problems with Application to a Fed-batch Fermentation Process. Computers and Chemical Engineering 23, 1277–1291 (1997)

Chiou, J.P., Wang, F.S.: Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed batch fermentation process. Computers and Chemical Engineering 23, 1277–1291 (1999)

Corne, D., Dorigo, M., Glover, F.: New Ideas in Optimization. McGraw-Hill Publications, London (1999)

Dadebo, S.A., Mcauley, K.B.: Dynamic optimization of constrained chemical engineering problems using dynamic programming. Computers and Chemical Engineering 19, 513–525 (1995)

Deb, K.: Optimization for engineering design: Algorithms and examples. Prentice-Hall, New Delhi (1996)

Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons Limited, New York (2001)

Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)

Dembo, R.S.: A set of geometric programming test problems and their solutions. Math. Program. 10, 193–213 (1976)

Edgar, T.F., Himmelblau, D.M.: Optimization of Chemical Processes. McGraw-Hill, Inc., Singapore (1989)

Elnashaie, S.S.E.H., Abdalla, B.K., Hughes, R.: Simulation of the industrial fixed bed catalytic reactor for the dehydrogenation of ethyl benzene to styrene: heterogeneous dusty gas model. Industrial and Engineering Chemistry Research 32, 2537–2541 (1993)

Floudas, C.A., Pardalos, P.M.: A collection of test problems for constrained global optimization algorithms. LNCS. Springer, Germany (1990)

Floudas, C.A.: Non-linear and mixed-integer optimization. Oxford University Press, New York (1995)

Gujarathi, A.M., Babu, B.V.: Multi-Objective Differential Evolution (MODE): A New Algorithm of Solving Multi-Objective Optimization Problems. In: Proceedings of International Symposium & 58th Annual Session of IIChE in association with International Partners, New Delhi (2005)

Gujarathi, A.M., Babu, B.V.: Multi-Objective Optimization of Styrene Reactor Using Multi-Objective Differential Evolution (MODE): Adiabatic vs. Steam Injected Operation. In: Proceedings of International Symposium & 59th Annual Session of IIChE in association with International Partners, Bharuch (2006)

Goldberg, D.E.: Genetic Algorithms for Search, Optimization, and Machine Learning Reading. Addison-Wesley, Reading, MA (1989)

Lee, M.H., Han, C., Chang, K.S.: Dynamic optimization of a continuous polymer reactor using a modified differential evolution algorithm. Industrial and Engineering Chemistry Research 38, 4825–4831 (1999)

Logsdon, J.S., Biegler, L.T.: Accurate solution of differential algebraic equations. Industrial and Engineering Chemistry Research 28, 1628–1639 (1989)

Lu, J.C., Wang, F.S.: Optimization of Low Pressure Chemical Vapor Deposition Reactors Using Hybrid Differential Evolution. Canadian Journal of Chemical Engineering 79, 246–254 (2001)

Madavan, N.K.: Multi-objective optimization using a pareto differential evolution approach. In: Congress on Evolutionary Computation, New Jersey, vol. 2, pp. 1145–1150 (2002)

Maranas, C.D., Floudas, C.A.: Global optimization in generalized geometric programming. Computers and Chemical Engineering 21, 351–370 (1997)

Onwubolu, G.C., Babu, B.V.: New Optimization Techniques in Engineering. Springer, Heidelberg (2004)

Pinto, E.G.: Supply Chain Optimization using Multi-Objective Evolutionary Algorithms, Technical Report (2002), available at:
http://www.engrpsu.edu/ce/Divisions/Hydro/Reed/Reports.htm

Price, K.V., Storn, R.: Home page of differential evolution (2007),
http://www.ICSI.Berkeley.edu/storn/code.html

Price, K.V., Storn, R.: Differential evolution – a simple evolution strategy for fast optimization. Dr. Dobb's Journal 22, 18–24 (1997)

Ray, W.H.: Advanced process control. McGraw-Hill, New York (1981)

Renfro, J.G., Morshedi, A.M., Osbjornsen, O.A.: Simultaneous optimization and solution of systems described by differential/algebraic equations. Computer and Chemical Engineering 11, 503–517 (1987)

Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers and Chemical Engineering 19, 551–566 (1995)

Sauer, R.N., Coville, A.R., Burwick, C.W.: Computer points way to more profits. Hydrocarbon Processing Petroleum Refiner 43, 84 (1964)

Sheel, J.G.P., Crowe, C.M.: Simulation and optimization of an existing ethyl benzene dehydrogenation reactor. Canadian Journal of Chemical Engineering 47, 183–187 (1969)

Storn, R.: Differential Evolution design of an IIR-filter with requirements for magnitude and group delay. International Computer Science Institute (1995)

Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution (2004), available at: http://www.math.upatras.gr/~dtas/papers/TasoulisPPV2004.pdf

Wang, F.S., Chiou, J.P.: Optimal control and optimal time location problems of differential-algebraic systems by differential evolution. Industrial & Engineering Chemistry Research 36, 5348–5357 (1997)

Wang, F.S., Cheng, W.M.: Simultaneous optimization of feeding rate and operation parameters for fed-batch fermentation processes. Biotechnology Progress 15, 949–952 (1999)

Yee, A.K.Y., Ray, A.K., Rangiah, G.P.: Multi-objective optimization of industrial styrene reactor. Computers and Chemical Engineering 27, 111–130 (2003)

Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland (2001)

Noise-Robust Tool Condition Monitoring in Micro-milling with Hidden Markov Models

K.P. Zhu, Y.S. Wong, and G.S. Hong

Department of Mechanical Engineering,
National University of Singapore, Kent Ridge Crescent, Singapore 11926

1 Introduction

Tool condition monitoring is crucial to the efficient operation of machining process where the cutting tool is subject to continuous wear. In particular, in micro machining, the tolerances, depth of cut, and even workpiece sizes are in micro scale. Micro-machining can overcome the shortcomings of micro fabrication techniques (such as lithography and etching) with limitation of work materials (mostly on silicon) and geometric forms (2 or 2.5 dimensions) (Byrne et al. 2003; Liu et al. 2004). One very versatile micro-machining process is micro-milling. Micro-milling has advantages over other micro-machining techniques with respect to the types of workable materials and the free-form 3D micro structures with high aspect ratios and high geometric complexity. However, in micro-milling, with the miniaturisation of the cutting tool (<1 mm in diameter), and the use of high speed (>10,000 rpm), the tool wears quickly. It is critical to monitor the tool wear in micro-machining due to the high precision required. Compared to conventional machining, the noise component in the signal for monitoring micro-machining is usually very high and difficult to separate (Tansel et al 1998; Zhu et al. 2007). This phenomenon makes it difficult to apply TCM in micro-machining.

This chapter presents the development of a general tool for signal denoising when the desired signal is contaminated with non-Gaussian noise and the noise spectrum is distributed widely in the frequency domain. It assumes that the noise is source and treats the signal denoising as blind source separation (BSS). A noise-robust multi-category tool wear state classification system has been developed for micro-milling. It is designed with left-right Hidden Markov Models (HMM) to provide a localized model of the tool condition, and to be noise-robust and not oscillating between different tool states in the classification.

2 Forms of Tool Wear

Tool wear is the common cause of, and inevitable precursor, to tool failure in machining processes. The extent of the tool wear has a strong influence on the surface finish and dimensional integrity of the workpiece. In general, tool wear can be viewed as the change of shape of the tool from its fresh state. The types of tool wear can generally be categorized as flank wear and crater wear, as shown in Fig. 1. Wears on the flank

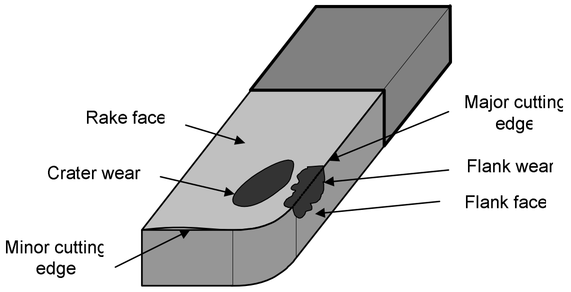


Fig. 1. Tool geometry

face and rake face are called flank wear and crater wear respectively. The most commonly observed phenomenon is the flank wear and is usually quantified by the width of the wear land. It can be easily measured under microscope. However, such operation will inevitably have to stop the cutting operation. As such, an indirect approach which relies on changes in the signal of a sensor associated with the wear of the tool face, is more commonly used for tool condition monitoring (TCM).

The aim of TCM is to apply appropriate sensor signal processing and pattern recognition techniques to identify and predict the cutting tool state, so as to reduce loss brought about by tool wear or tool failure. Effective tool condition monitoring (TCM) systems can improve productivity and ensure workpiece quality, and hence, has a major influence in machining efficiency (Shaw 2005). TCM is widely studied in conventional machining (Byrne et al. 1995). Generally, indirect measurement methods such as force, acoustic emission, and vibration are used to capture machining signals, and approaches such as autoregressive models (Liang and Dornfeld 1987), neural networks (Hong et al. 1996; Niu et al. 1998; Tansel et al. 1998; Saglam and Unuvar 2003; Sun et al. 2004; Wang et al. 2007), and pattern recognition (Heck and Chou 1994; Du et al. 1995; Li and Elbestawi 1996; Kumar et al. 1997; Li et al. 2000), are then applied to estimate the tool wear states. Hidden Markov models (HMMs) have recently been employed for TCM lately (Wang et al. 2002; Fish et al. 2003; Kassim et al. 2006) due to their excellent representation of local dynamics of signals and reasoning property in speech recognition (Rabiner 1989; Li and Douglas 2003). Table 1 lists them according to their classification approaches. Note that the list is not totally exhaustive but serves to be representative of known TCM approaches.

Table 1. Tool state estimation approaches

	Approa-hes	References	Comments
Time series	AR, ARMA	Liang and Dorn-feld, 1987	Linear, good for stationary machining like turning, not good for non-stationary machining like milling; need set threshold for classification
	MLP	Hong et al. 1996; Niu et al 1998; Tansel et al. 1998	Iterative MSE optimization; sensitive to network structure; nonlinear classification, no good generalization

Table 1. (*continued*)

Neural Networks	SOM	Wang et al. 2007	Nonlinear and iterative clustering; suitable for low dimensional feature vectors
	ART	Saglam and Unuvar 2003	Based on competitive learning; good self-adaptive ability; fast incremental learning ability,
	SVM	Sun et al. 2004	Maximizing the margin between classes; metric dependent, nonlinear, good generalization; suitable for lack of feature sets
Pattern Recognition	k-means	Du et al. 1995	K-clusters, the nearest mean decides the cluster, good for Gaussian signal with equal covariance
	Fuzzy Methods	Li and Elbestawi 1996; Li et al. 2000	Need initializing clusters and class membership
	Gaussian Mixture Models	Heck and Chou 1994	Each state is assumed to be drawn number of underlying Gaussian distributions; need estimate components,
	LDA	Kumar et al. 1997	Supervised linear classifier, using MSE for optimization, need Gaussian assumption of signal
Stochastic Models	Hidden Markov Models	Wang et al. 2002; Fish et al. 2003; Kassim et al 2006	Simple structure; Good in generalization; good in nonlinear and non-stationary machining signals; need to train many small models

AR: Autoregressive; ARMA: Autoregressive Moving Average; MSE: Minimum Square Estimation; LDA: Linear Discriminant Analysis; MLP: Multilayer Perceptron; SOM: Self-Organizing Map; ART: Adaptive Resonance Theory; SVM: Support Vector Machines, GMM: Gaussian Mixture Models; HMM: Hidden Markov Models.

3 TCM as a Pattern Recognition Problem

The problem of TCM can be considered as a typical pattern recognition problem. The objective of a TCM system can be achieved in a three-step procedure, as shown in Fig. 2. The sensor signal $x(t)$ is firstly pre-processed to remove the noise and prepare the data for feature extraction. Then the information relevant to pattern classification is extracted from $x'(t)$ to a feature vector $y(t)$. The task of feature extraction is to

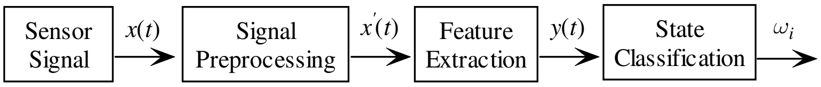


Fig. 2. Data flow in TCM system

enhance the characteristics of the various tool wear classes and suppress or filter off the normal background. The final stage is state classification. Feature vector *y* is assigned to one of the *K* tool wear state, $\omega_1, \omega_2, \ldots, \omega_i, \ldots \omega_K$, by the classifier based on a certain type of classification criteria, such as Bayes decision rule, neural networks, clustering, and hidden Markov models suitable for this purpose.

We can formally specify the objectives of TCM to be a search for the most probable state ω_i given the extracted measurable signal feature y . This is a dynamic inference problem since we do not estimate the tool state only with prior knowledge, but also adapt to the current features. Hence, the aim of TCM is to find,

$$\text{TCM:}\ \arg\max_{i} p(w_i \mid y) \tag{1}$$

or in the physical form,

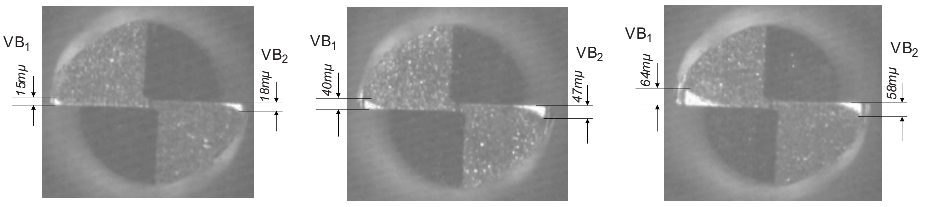
$$\text{TCM:}\ \arg\max_{tool\ state\ i} p(tool\ state \mid signal\ features) \tag{2}$$

4 Definition of Tool Wear State in Micromachining

Compared to the standard maximum flank wear of 0.3mm in conventional machining (ISO 8688-2: 1989), there is no general definition of tool wear level in micro maching. This has to be redefined in micro-machining because the total cutting edge is less than of 0.3mm generally. According to Tansel et al. (1998, 2000), and Rahman et al. (2001), it is a matter of different degrees of wear and hence, instead of a single

Table 2. Definition of tool state

Tool diameter	threshold	State 1	State 2	State 3
500 μm	Estimated	17.9μm	17.9-53.8μm	53.8μm
500 μm	Final	0-20μm	20-50μm	>50μm
800 μm	Estimated	23.2μm	23.2-57.5μm	57.5μm
800 μm	Final	0-20μm	20-60μm	>60μm



a) slight wear b) Medium wear c) Severe wear

Fig. 3. Flank wear a mill tool with diameter Φ800μm

indicator, multi-category identification and classification of tool wear for monitoring progressive wear of the tool is more appropriate.

In order to establish various wear state for micro-machining, we refer to the Taylor's tool life curve (Shaw 2005). There are initial wear region, gradual wear region and the accelerated wear regions. We take the average of the two flute flank wears $VB=1/2(VB_1+VB_2)$ as the final flank wear value (Fig. 3). In this chapter, the progressive change in the flank wear is approximated by a B-spline curve, and the cross over points of the second order derivative of the Taylor tool life curve is chosen to be the boundary between different regions. The average values measured by the thresholds of these three states are listed in Table 2. We then choose their closest 10μm as the final threshold for convenient representation. This rounding of the threshold is based on the belief that the threshold is roughly estimated while the features can represent their corresponding wear level well.

5 Experiment Setup

Experiments were conducted to verify the application of the proposed algorithm for micro-milling tool wear state estimation. The machine used in the experiments is a MAKINO V55 vertical milling machine driven by a 22kw spindle drive motor. The cutting force was measured with a Kistler 9256A 3-channel dynamometer where the workpiece was mounted (Fig. 4). Tool wear was measured using the Olympus Toolmakers microscope (213 times enlargement). The cutting force output was recorded on a Sony digital tape recorder. The sampling rate of the cutting force is 6,000 Hz. The force signal data are processed in windows of 360 points, with each window duration equivalent to 20 revolutions of the cutter and 0.06 second. The micro-machining conditions are listed in Table 3. A total of 27 test experiments were conducted with different spindle speed, depth of cut, radial depth of cut, and feed rate for *Φ500μm* and *Φ800μm* tools. The materials used are either copper and steel.

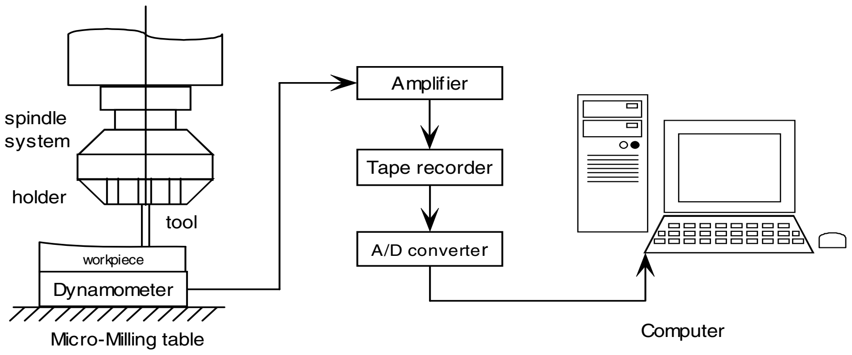


Fig. 4. Experiment setup

Table 3. Experiment setup

Test	Material	Tool diameter (μm)	Spindle Speed (rpm)	Depth of cut (μm)	Radial depth of cut (μm)	Feed rate (mm/min)
1	Copper	500	20,000	30	75	80
2	Copper	500	20,000	30	75	120
3	Copper	500	20,000	40	75	100
4	Copper	500	20,000	60	75	120
5	Copper	500	20,000	60	75	150
6	Copper	500	20,000	80	75	150
7	Copper	500	20,000	80	75	180
8	Copper	500	18,000	60	75	120
9	Copper	500	18,000	80	150	150
10	Copper	500	18,000	100	150	150
11	Copper	500	18,000	100	225	150
12	Copper	500	18,000	100	225	180
13	Copper	500	18,000	120	225	150
14	Copper	500	18,000	100	300	180
15	Copper	500	18,000	120	300	180
16	Copper	500	18,000	150	225	150
17	Copper	500	18,000	120	300	150
18	Steel	800	18,000	80	240	120
19	Steel	800	18,000	80	80	100
20	Steel	800	18,000	80	120	100
21	Steel	800	18,000	80	80	120
22	Steel	800	18,000	80	100	100
23	Steel	800	18,000	80	80	80
24	Steel	800	18,000	60	80	80
25	Steel	800	18,000	60	80	60
26	Steel	800	18,000	60	60	60
27	Steel	800	18,000	60	80	100

6 TCM in Micro Machining

As discussed in above sections, the flank wear length has to be redefined in micro machining. Because the machining process is in micro-scale, the signal noise associated with the cutting force is also significantly different from the traditional machining processes that non-Gaussian noise is also observed. As such, a general tool is proposed and developed for signal denoising when the desired signal is contaminated with non-Gaussian noise and the noise spectrum distributed widely in the frequency domain. The proposed approach assumes that the noise as a source and treating the denoising problem as blind source separation problem (BSS), and then developing a noise-robust multi-category tool wear state classification system in micro-milling. It is designed with left-right Hidden Markov Models (HMM) to provide a localized model of the tool condition, and to be noise-robust and not oscillating between different tool states in the classification. Key components of the proposed and developed architecture and the methodology are shown in Fig. 5.

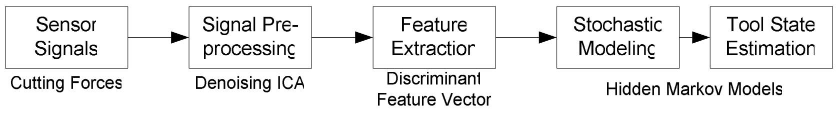


Fig. 5. System architecture

Firstly, the sensor signal has to be de-noised. In micro-milling, the comparatively small cutting force signal is prone to contamination by relatively large noises, and as a result it is important to de-noise the force signal before further processing. In the ICA (define – mentioned first time here?) approach, noise is treated as a source and sensor signals as an instantaneous mixture of sources (both noise and force signal). The signal de-noising process is treated as blind source separation (BSS), which overcomes the traditional Gaussian noise assumption.

The next is feature extraction. The features from both wavelet packets and from time domain, i.e. mean, standard deviation, skew, kurtosis and dynamic component, are combined to form a 37-dimension feature vectors. Furthermore, the dimensionality of parameter vectors is normally very high and needs to be reduced for the sake of less computational cost and system complexity. Fisher's linear discriminant analysis (FDA) is adapted for this purpose. In the discriminant selection, features are chosen to maximize class separation and are ranked by their separation ability between different classes.

The final stage is state classification. Continuous Hidden Markov models (HMMs) are adapted for stochastic modeling of the tool wear process in micro-milling, and estimation of the tool wear state based on the output probability of the HMMs given the cutting force features. Different tool wear states are modeled as separate HMMs. The tool wear state is then classified by the HMM that has the maximum probability to indicate the test features.

7 ICA for Cutting Force Denoising in Micro-milling

Cutting force has been found to correlate well with tool conditions in machining and most effective as sensor signal for tool wear monitoring (Byrne, 1995; Wu et al. 2001). However, the noise component in the signal for monitoring micro-machining is usually very high and difficult to separate (Tansel et al. 1998; Zhu et al. 2007). Fig. 6 shows a typical cutting force and noise in micro-milling. It can be observed that the force signal is very low and highly non-stationary (periodically involves impulse), and the signal-to-noise ratio is relatively very low. Effective signal processing for this

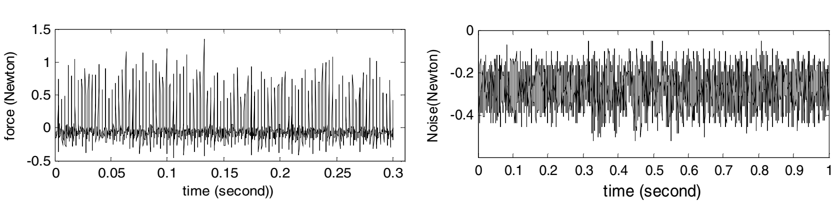


Fig. 6. Typical cutting force and noise in micro-milling

Table 4. The statistics of reference noise

Statistics	Function	Value
Mean	$\mu = E[x(t)]$	-0.2776
Variance	$\sigma^2 = E[(x(t)-\mu)^2]$	0.2631
Skew	$Skew = E[(x(t)-\mu)^3/\sigma^3$	0.0004
Kurtosis	$Kurtosis = E[(x(t)-\mu)^4/\sigma^4 - 3$	1.2621

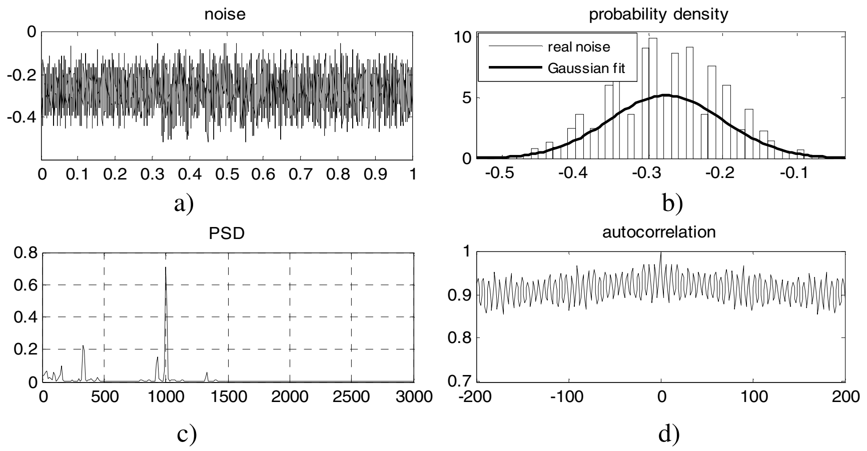


Fig. 7. Illustrative of the statistics of noise: a) noise signal, b) noise distribution compared to Gaussian distribution, c) the corresponding power spectrum density, d) the autocorrelation coefficients

highly non-stationary and noisy signal to provide a robust approach for tool condition monitoring (TCM) is very important in micro machining.

The reference noise is collected when there is no machining. For a typical sample of noise from the experiments (Fig. 7), their statistics are computed for reference in table 4. It is super-Gaussian, with a longer tail than Gaussian distribution (See Fig. 7b)). The eigen-frequency of the cutting force is Fr=20000 rpm /60 s =333.3 Hz. The integer times of *Fr*, *nFr* is the characteristic frequencies of cutting force. The PSD and auto-correlations show that the noise is not random. The PSD harmonics peaks at the characteristic frequencies (*nFr*) and the autocorrelations is very high. Typically it is like a component of cutting force. The traditional approaches such as digital filtering and wavelet thresholding (Donoho, 1994) are not effective in this condition.

7.1 Independent Component Analysis and FastICA

Independent component analysis (ICA) (Bell and Sejnowski 1995; Hyvarinen 1999; Hyvarinen et al 2001) is a recently developed method for blind source separation (BSS) (Cardoso 1998). ICA intends to recover the sources using only the observed signals and on the general assumption that the expected sources are mutually independent. ICA has found wide applications in biomedical systems, speech separation and array processing. It was applied to signals generated by mechanical components and systems lately (Gelle and Colas 2001; Roan et al. 2002; Tian et al 2003; Serviere and Fabry 2004). Little has been investigated with the application of the ICA for signal denoising and to the analysis of signals such as those from cutting forces. This section introduces a FastICA (Hyvarinen 1999) algorithm as a preprocessor to provide noise free forces for later correlation to tool flank wear. This approach assumes that the noises are sources and treat the denoising process as blind source separation. We apply the FastICA for these blind sources separation and then discard the separated noise components.

In general, m mixed signals $x_i(k)$ $(i = 1, 2, ...m)$ are linear combinations of n unknown mutually statistically independent, source signals $s_i(k)$ $(i = 1, 2, ...n)$, and noise $v_i(k)$ $(i = 1, 2, ...m)$. This can be expressed as

$$x_i(k)\ (i = 1, 2, ...) = \sum_{j=1}^{n} w_{ij} s_j(k) + v_i(k) \tag{3}$$

Our aim is to find some methods to reverse this mixing process (un-mixing) and find the original sources. Assume that there is a matrix H and,

$$H\mathbf{x}(t) = H[W\mathbf{s}(t)] = (HW)\mathbf{s}(t) \tag{4}$$

$$\textit{If}\ \ HW = I,\ \textit{i.e.}\ H = W^{-1}, \tag{5}$$

$$\textit{then}\ H\mathbf{x}(t) = I\mathbf{s}(t) = \mathbf{s}(t)$$

In this way, if we can find some un-mixing matrix H, we will recover the sources $\mathbf{s}(t)$. This whole separation process is illustrated as Fig. 8.

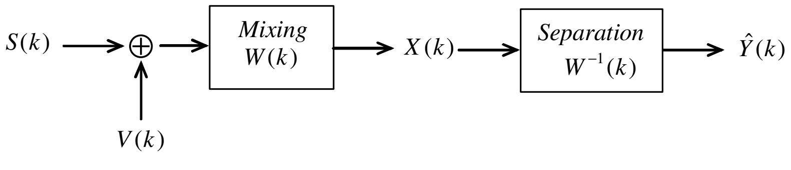


Fig. 8. General model of ICA: $X(k)$ observations, $Y(k)$ estimated sources, $S(k)$ sources

Generally the unmixing matrix can be found by virtue of the non-Gaussian nature of the underlying sources. Depending on the different measurements and optimization functions of nonGaussianity in the ICA, there are various approaches of ICA (Hyvarinen et al. 2001). In this approach, we apply the FastICA (Hyvarinen 1999) and measure the nonGaussianity with negentropy. Take $f(y)$ as the probability density function (PDF) of random variable y, the entropy of $\boldsymbol{y}$ is then defined as:

$$H(Y) = -\int f(y)\log(f(y))dy \tag{6}$$

Gaussian variable has the largest entropy among all random variables of equal variance, and this means that negentropy could be used as a measure of non-Gaussianity. Negentropy is defined by:

$$J(Y) = H(Y_{gauss}) - H(Y) \tag{7}$$

where $Y_{Gaussian}$ is a multivariate normal with the same covariance matrix as *Y*. *J(Y)* is approximated by:

$$J(Y) \approx k[E(G(Y)) - E(G(v))]^2, \tag{8}$$

For some $k \geq 0$, $v \sim N(0,1)$ and *Y* is standardized. $G(v) = tanh(u)$ in this study.

To estimate several independent components, we need to run the one-unit FastICA algorithm using several units (e.g. neurons) with weight vectors $w_1, \ldots w_n$. To prevent different vectors from converging to the same maxima we decorrelate the outputs $w_1^T x, \ldots, w_n^T x$ by deflation after each iteration. The procedure can be continued until all the estimated source signals are recovered. This process is implemented as:

$$1.\ \text{Let } w_{p+1} = w_{p+1} - \sum_{j=1}^{p} w_{p+1}^T w_j w_j \tag{9}$$

$$2.\ \text{Let } w_{p+1} = w_{p+1} / \sqrt{w_{p+1}^T w_{p+1}} \tag{10}$$

7.2 Noise Separation in Micro-milling

Now we apply the FastICA to the cutting forces separation. Three sources are expected: one force source, one non-Gaussian source for machining noise and one Gaussian source of environmental noise. Noises are treated as sources here. The sensor outputs and their corresponding power spectrum are illustrated in Fig. 9. As can be seen from the figure., the low-frequency components have the highest peaks all along the time axis and as a result the desired harmonics *nFr* are immersed.

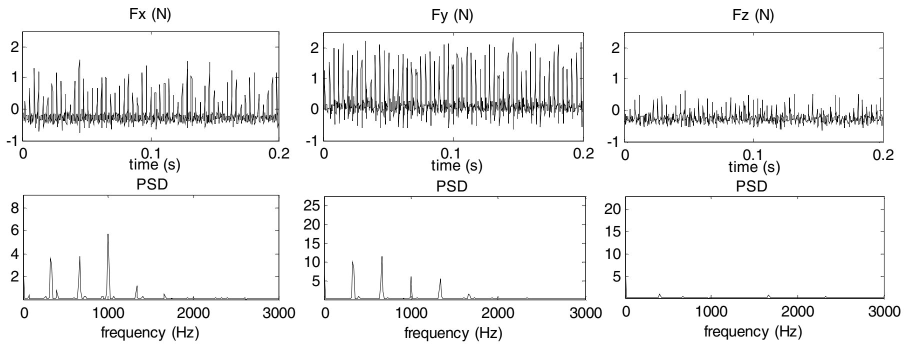


Fig. 9. The sensor output and their corresponding power spectrum

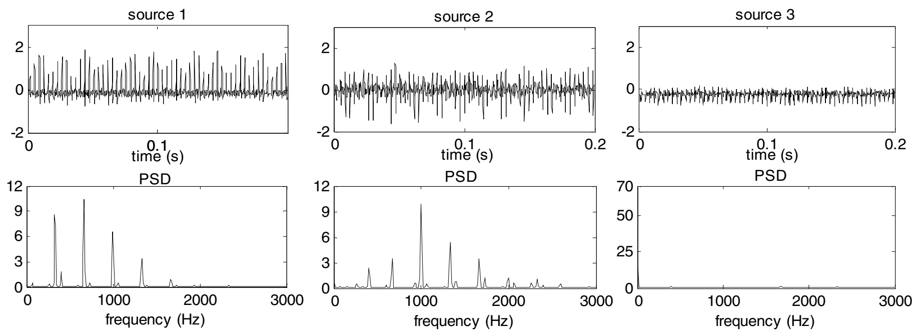


Fig. 10. Cutting forces reconstructed

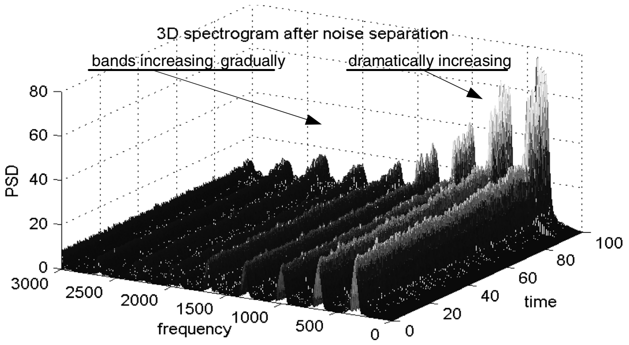


Fig. 11. Tool wear with ICA reconstructed force

And now we apply the FastICA. The reconstructed sources and their corresponding power spectrum are illustrated in Fig. 10. It is clear that the reconstructed force peaks are at the first five harmonics, while the machining noise peaks are at low frequency and a wide range of harmonics, and the environmental noise peaks are at very low frequency.

We now apply the Short Time Fourier Transform (STFT) to demonstrate the full-pass results. All the reconstructed force are added together and illustrated by spectrogram in Fig. 11. It is observed that after 80 second *1Fr*, *2Fr*, and *3Fr* of the feed force increased dramatically, corresponding to the sharp increase of tool flank wear. To show the effectiveness of FastICA in the removal the noise, we compare it with the signal denoised by wavelet.

For comparison we apply wavelet thresholding for denoising the same forces. Here we choose soft thresholding with feed force *Fx*, as feed force is more correlated to the flank wear in these experiments (Zhu et al. 2007). The same passes were added together to show their overall denoising performance in Fig. 12. Comparing the Fig. 11 and Fig. 12, we can see Fig. 11 shows better diagnostic information since harmonics stand out and the first three harmonics increasing dramatically according to the fast increase of flank wear.

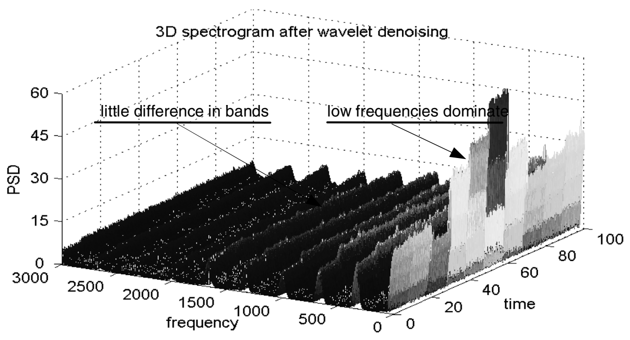


Fig. 12. Tool wear with wavelet denoised force

From above, it can be observed that FastICA shows better force denoising performance than wavelet for use in micro milling tool wear monitoring. In micromachining there are machining noises that correlate to the machining conditions and have wide harmonics and non-Gaussian distribution. FastICA denoises the force signal efficiently by directly separating Gaussian and non-Gaussian noises as signal sources, under the measure of negnentropy. In this process, the non-Gaussian noise or noise harmonics are removed as source separation. Wavelet thresholding estimates too small threshold in this case (Zhu et al. 2007), and does not work well by discarding smaller coefficients since the non-Gaussian noise coefficients are not small and evenly distributed in certain bands.

8 Hidden Markov Models (HMMs) Based Tool Condition Monitoring

Like that of conventional milling, tool wear increases progressively with machining time, given specific working conditions in micro-milling. Generally the dynamic wearing process is expressed as,

$$w(t+\Delta t) = w(t) + v(t)\Delta t \tag{11}$$

where $w(t)$ is the tool wear value and $v(t)$ wear rate at machining time t. The tool wear value $w(t+1)$ at time $t+1$ is to be predicted given the current value $w(t)$ and wear rate $v(t)$. This is a first-order Markov process where the current state only depends on the immediately preceding state in time. More formally,

$$p(w(t+1) \mid w(t), w(t-1), \ldots w(1)) = p(w(t+1) \mid w(t)) \text{ (hidden process)} \tag{12}$$

This process is hidden to us, and hence classed as *hidden process*, which we try to uncover. For the practical situation, we cannot measure the current tool wear value directly, but estimate it from the observation of the force features $y(t)$. This relationship is very complex, and typically non-stationary and uncertain. It is a function of $w(t)$:

$$y(t) = f(w(t)) \qquad \text{(observation process)} \tag{13}$$

which represents the relationship of a certain feature $y(t)$ and a tool wear value $w(t)$. $y(t)$ is generally extracted from measurable sensor signals, so we call it the *observation process*.

The hidden Markov model (HMM) (Rabiner 1989; Li and Douglas 2003) is a double-layered stochastic process, with an underlying finite-state hidden Markov process that associates with the observation process. For our application, the force features $Y = \{y_1, y_2, ..., y_T\}$ are the observation process, and the tool state sequence $S = \{s_1, s_2, ..., s_T\}$ is the hidden process. The probability of the observations Y is conditionally dependent on state S,

$$P(Y) = \sum_S P(Y, S) = P(y_1|s_1)P(s_1)\prod_{t=2}^{T} P(y_t|s_t)P(s_t|s_{t-1}) \tag{14}$$

In order to characterize an HMM completely, the following elements are needed. An HMM is identified with the parameter set $\lambda = (\pi, A, B)$ where

π is the initial state distribution vector, e.g. π is the probability of state i at the same arbitrary time $t = 0$.

A is the state transition matrix, where $A = [a_{ij}]$, with a_{ij} being the probability of transiting to state j given current state i.

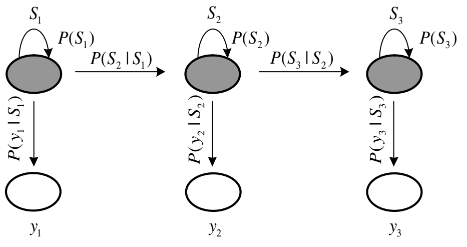
B is the output distribution matrix, where $B = [b_{jk}]$, with b_{jk} being the probability of observing feature k given current state j.

The estimation and selection of parameter set $\lambda = (\pi, A, B)$ are the essence of HMM modeling, which are discussed in the following sections.

8.1 Hidden Markov Model Structure Selection: Continuous Left-Right HMMs

8.1.1 Left-Right HMM

The most widely used HMMs are the ergodic and the left-right HMMs. The ergodic HMM is fully connected with all states linked together so that every state can be



Transition matrix:

$$\begin{bmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix} \quad \begin{array}{l} a_{ij} = P(s_j \mid s_i) = 0 \ \ for \ \ |i-j| > 1 \,\&\, i < j \\ \text{Initialization: } \pi = [1 \ \ 0 \ \ 0] \end{array}$$

Fig. 13. Left-right HMM tool state model

reached from every other state. Ergodic HMMs are not feasible for TCM, however, because in machining the tool wear is always increasing and wear state can not change back. The left-right HMM can transit to itself and later state, but not to the former state, and it is suitable for describing tool wear process. A 3-state left-right HMM is shown in Fig. 13. This transition matrix is thus constrained to an upper triangular matrix. The initial state of left-right HMM is $\pi = [1 \;\; 0 \;\; 0]$ since the tool state starts at the first state.

Unlike that of (Fish et at, 2003), the HMMs in this study does not involve jumping between states but progresses from one to another, i.e. the tool starts in the initial wear state, moves into the gradual wear state, which is relatively long in micro-milling, before advancing into the accelerated wear state.

8.1.2 Continuous HMMs and Gaussian Mixtures Modeling

Besides classification according to structures, HMMs can also be classified into discrete HMMs or continuous HMMs, depending on whether the observation features are modeled as continuous or discrete. In the case of a discrete HMM, the features are degraded significantly by Vector Quantization (VQ) (Li and Douglas 2003). Since it is an average over large regions. Moreover, when a new class is added, it is necessary to reconstruct all of the features and retrain all of the models. We choose continuous HMMs for our study.

In a continuous HMM, the feature (observation) space *Y* is considered to be infinite and continuous. Gaussian Mixtures (GM) are typically used to model the variation feature vectors which correspond to a state due to its variability and accuracy. By varying the number of Gaussians *K*, the weights c_{ij} , and the mean μ_{ij} and covariance matrix Σ_{ij} of each Gaussian density function $N(\mu_{ij}, \Sigma_{ij})$, GM can be used to describe any complex density function. The observation $\{y_i\}$ is modeled as a *K*-Gaussian mixture as,

$$y_i = \sum_{j=1}^{K} c_{ij} N(\mu_{ij}, \Sigma_{ij}) \tag{15}$$

where the weights are all positive and sum to one.

8.2 Selection of the Number of Hidden States in Each HMM

The selection of appropriate number of hidden states in the HMM is from a physical point of view. Generally with one HMM modeled for one tool state, the states of HMMs lack physical meaning as a single HMM for different tool wear states. The tool wear state is characterized by the corresponding entire HMM in our models. The states in that HMM are the Markov transition state within the HMM, but not the tool wear states. In milling, the feature patterns are quite different even in the same tool wear state but among different cutting stage. So we train HMMs with the data based on each cutting pass, and constrain every HMM with three states as: entry, in-progress, and exit. In this sense, the HMMs can estimate both the tool wear state (i.e., which HMM matches most) and the cutting state (i.e., which state in that HMM is highly probable).

8.3 Estimation of the Hidden Markov Model Parameters

For the Gaussian mixture HMMs, $B=B(c, \mu, \Sigma)$; then the HMMs are identified with the parameter set $\lambda=(\pi, A, c, \mu, \Sigma)$. Once the number and topology of HMM states are chosen as discussed above, the distribution parameters are to be trained. The training problem can be described as: Given the observed features $\{y_i\}$, determine the HMM parameters λ that best characterize the features. These parameters can be iteratively estimated by the Expectation Maximization (EM) algorithm (Baum-Welch algorithm) (Rabiner 1989).

The probability of being at state S_i at time t and state S_j at time $t+1$ is $\gamma_t(i,j)$. Each of the parameter in the model is maximized separately with EM, by setting its respective partial derivative equal to zero. The results are:

$$\pi_i = [1\ 0\ 0] \tag{16}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \gamma_t(i,j)} \tag{17}$$

$$\bar{c}_{ij} = \frac{\sum\limits_{t=1}^{T-1} \gamma_t(i,j)}{\sum\limits_{t=1}^{T} \sum\limits_{j=1}^{M} \gamma_t(i,j)} \tag{18}$$

$$\bar{\mu}_{ij} = \frac{\sum\limits_{t=1}^{T-1} \gamma_t(i,j) Y_t}{\sum\limits_{t=1}^{T} \gamma_t(i,j)} \tag{19}$$

$$\bar{\Sigma}_{ij} = \frac{\sum\limits_{t=1}^{T-1} \gamma_t(i,j)[(Y_t - \mu_{ij})(Y_t - \mu_{ij})']}{\sum\limits_{t=1}^{T} \gamma_t(i,j)} \tag{20}$$

9 HMM for Micro-milling TCM

9.1 The Framework of HMM for Micro-milling TCM

The tool state estimation problem can be formalized as: find λ = *argmax* $P(Y|\lambda)$. Given a fixed HMM with parameters $\lambda=(\pi, A, c, \mu, \Sigma)$, determine the most likely hidden states $\{s_i\}$ for an observed set of features $\{y_i\}$. The framework of HMMs for TCM is illustrated in Fig. 14. The features from both time and wavelet domain are extracted and selected to train the HMMs. We build an HMM for each tool state for classification, and each HMM is trained with its own feature set. Once the models are trained, we get the λ=(π, A, c, μ, Σ) that represent different tool wear states. In the tool state recognition state, the extracted features from an unknown state are extracted and input to the HMMs to match the trained states. The Viterbi algorithm (Rabiner 1989) is implemented to estimate the most probable tool wear state sequence.

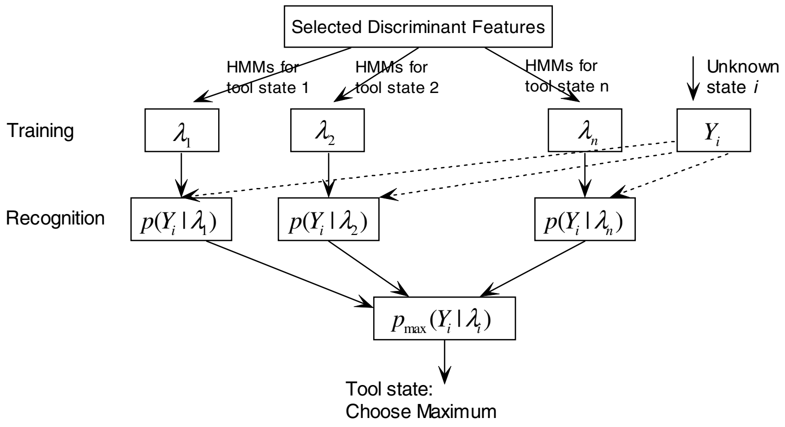


Fig. 14. Framework of Hidden Markov Models for tool wear classification

9.2 Feature Extraction for HMM Modeling

Before we could model the tool condition with cutting forces, force features have to be extracted for robust and effective representation. Because of the intermittent cutting process of micro-milling, the cutting forces are typically non-stationary. Wavelet analysis (Mallat 1998) overcomes the pitfalls of Fourier methods and is most suitable for processing milling forces. For convenience, each coefficient vector is represented by the average energy of the wavelet packet coefficient $d^n_{j,k}(t)$,

$$E_j = \frac{1}{N_j}\sum_{k=1}^{N_j}\left[d^n_{j,k}(t)\right]^2 \tag{21}$$

where $d^n_{j,k}(t)$ is the wavelet packet coefficient localized at $2^j k$ in scale 2^j, and n is the oscillation parameter $n = 1, 2, \ldots$, N_j is the number of coefficients at each scale.

The wavelet Daubeches 8 is used to decompose cutting forces into 5 levels in this study. We use all the level-5 wavelet packets, but none from the other levels, since this level segments the force into the largest parts (32 wavelet packets) and separates the most apart of the signal's low frequency and high frequency. Different frequency bands have been proven to be effective to represent the force from different wear level. The features are represented by the mean energy of the respective packet node,

Table 5. Feature extraction scopes

	Features	Formula
Time domain	Mean	$\mu = E[f(t)]$
	Variance	$\sigma^2 = E[(f(t)-\mu)^2]$
	Skew	$Skew = E[(f(t)-\mu)^3 / \sigma^3$
	Kurtosis	$Kurtosis = E[(f(t)-\mu)^4 / \sigma^4 - 3$
	Dynamic Component	$\Delta F_a(t) = \lvert F_a(t) - F_{med}(t) \rvert$
Wavelet domain	Wavelet Packet Coefficients	$E_j = \frac{1}{N_j}\sum_{k=1}^{N_j}\left[d_{j,k}^n(t)\right]^2$

$F_a(t)$: amplitude of raw force; $F_{med}(t)$: mean force of the segment.

with $d_{5,1}$ represents first nods of level 5 decomposition. Time domain features (Goebel and Yan 2001) are also extracted to show the force statistics. They are mean (T_1), standard deviation (T_2), skew (T_3), kurtosis (T_4) and dynamic component (T_5), as defined in the Table 5. Note that the dynamic component one is different from the variance: It measures how far the signal from its medium and it is useful when there are spikes. To be comparable with the features from the wavelet packets, they are rescaled to their 1/32 times. The features from wavelet packets and from time domain are combined to form a 37-dimension feature vectors. These features are normalized between [0 1] to eliminate the effect of different working conditions.

Tool wear classification algorithms should consider whether the features are discriminating enough, and avoid unnecessary computation or hinder the subsequent classification process. A feature selection algorithm that deals with this problem is necessary. We adapt the Fisher's linear discriminant analysis (FDA) (Theodoridis and Koutroumbas 2003) for feature selection that ranks features by their class-discriminant ability. The *d*-dimensional sample mean vector μ_i and the covariance matrix S_i of class *i*. For the multi-class case, the *Fisher's discriminant ratio* (*FDR*) is defined as:

$$FDR(m) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K}\left|\mu_i^m - \mu_j^m\right|^2}{\sum_{i=1}^{k} s_i^m} \tag{22}$$

where the subscripts i , j refer to the mean and variance corresponding to the feature under investigation for the class i , class j respectively, and *m* is feature index. Features are then ranked in order of descending values of *FDR(m)*. The selected top discriminant features are then modeled and recognized by the Hidden Markov Models (HMMs). Table 6 shows the top 8 discriminant features for the three tool wear states. Most of the features are from wavelet domain, and the band of which include the force harmonics. The only selected time feature is the dynamic force (T_5). Fig. 15 shows the FDA score of each features for the three tool wear states. As shown in the Fig., some of the features are quite different (e.g. feature 8, 10 and 21) from the three

Table 6. Top discriminant features

	Top 8-discriminant features							
Copper	$d_{5,16}$	$d_{5,3}$	$d_{5,5}$	$d_{5,6}$	T_5	$d_{5,7}$	$d_{5,1}$	$d_{5,13}$
Steel	$d_{5,12}$	$d_{5,2}$	$d_{5,4}$	$d_{5,8}$	T_5	$d_{5,6}$	$d_{5,16}$	$d_{5,1}$

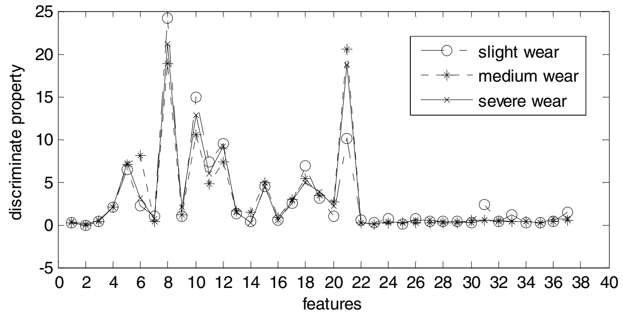


Fig. 15. Discriminant feature selection

classes while others are similar (e.g. feature 27, 28, and 29). These reduced feature vectors are then used as the input to the Hidden Markov Models (HMMs) for classification.

9.3 HMM Training for TCM

To train the continuous HMM for TCM, the feature vectors are modeled as Gaussian Mixture Model (GMM). The number of mixture components has to be decided. Theoretically a GMM can approximate any signal to a certain precision when provided with enough mixture components. Too many components may not be desirable, however, as they over fit the data and involve needless computation. The Bayesian Information Criterion (BIC) provides a reliable approximation to the integrated likelihood and is a most popular criterion (Fraley, and Raftery, 1998). We apply the BIC for the selection in this study. The preferred model is the one with the highest value of the criterion. The number of Gaussian components is identified to be 9 among the

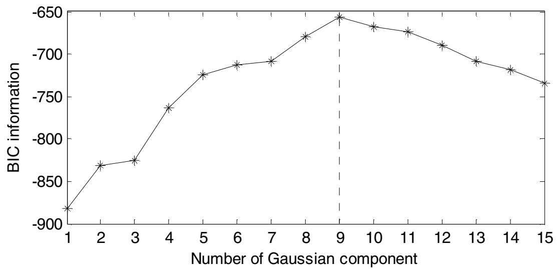


Fig. 16. Bayesian information criterion for mixture components

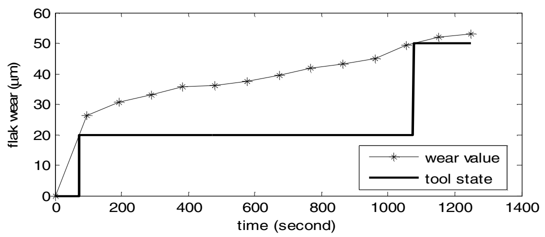
training tests for the 8-dimension feature vectors. The average BIC score over the tests reaches the highest as illustrated in the Fig. 16.

After the selection of the structures of GMM, each HMM is trained with its own training sets. To train the HMM that represents the fresh tool state, we only use the feature vectors from that state, and similarly for the medium wear state and severe wear state. We apply the EM algorithm to estimate the HMM parameter $\lambda = (\pi, A, c, \mu, \Sigma)$. The algorithm is implemented to interactively adjust the model parameter to maximize the probability of the observation sequence. The EM method works as discussed in section 4.4. Practically we need to set a threshold for the end of training.

```
Expectation-Maximization
    1. Estimate an initial HMM model as $\lambda = (\pi, A, c, \mu, \Sigma)$.
    2. Given λ and the observation sequence Y, we calculate a new model
           $\lambda = (\pi, A, c, \mu, \Sigma)$ such that $P(Y \mid \bar{\lambda}) > P(Y \mid \lambda)$.
    3. If the improvement $\frac{P(Y \mid \bar{\lambda}) - P(Y \mid \lambda)}{P(Y \mid \bar{\lambda})} < \mathit{threshold}$, then stop
           Put $\bar{\lambda}$ instead of $\lambda$ and go to step 1.
```

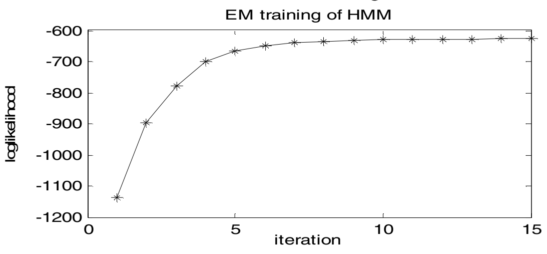
Fig. 17 a) shows one of the training set for the HMM modeling for copper. The selection of the training set is based on the distance to the mean working parameters. The EM algorithm converges quickly as illustrated in Fig. 17 b) from test 4.

Other working conditions are also trained to build HMMs in the same way. They are the test 10 and test 13 for copper, and test 20 and test 24 for steel, which are listed in the Table 3.



Test 4, Copper
Spindle speed: 20,000rpm
DOC: 60μm
Radial DOC: 75μm
Feed rate: 120mm/min

a) Tool wear state for training the HMM



b) Convergence of EM for HMMs training

Fig. 17. HMM training for test 4

9.4 HMM for Tool Wear State Estimation

Because there are 5 HMMs trained, we need to decide which HMMs suit best for the tool state estimation. Given the working condition, a similarity measure based on the Euclidian distance of the working condition is first determined as follows:

$$d_i = \left[(\frac{v - v_{ri}}{v_{ri}})^2 + (\frac{f - f_{ri}}{f_{ri}})^2 + (\frac{d_v - d_{vri}}{d_{vri}})^2 + (\frac{d_r - d_{rri}}{d_{rri}})^2 \right]^{1/2} \tag{23}$$

where d_i is the distance, v_{ri} is the *i-th* reference spindle speed, f_{ri} is the *i-th* reference feed rate, d_{ri} is the *i-th* reference vertical depth of cut, and d_{rri} is the *i-th* reference radial depth of cut. Our aim is to find the one with the minimum distance, $\min d_i$. The other two variables, i.e. work piece materials and tool diameters, are not included here as they are usually predetermined and assumed to be the same.

The features from both time and wavelet domain are extracted and input to the trained HMM for recognition. We build an HMM for each tool state for classification, and each HMM is trained with its own feature set. To obtain the HMM for the fresh tool state, for example, we only select the features with tool wear from 0-20μm, and likewise for the subsequent wear states. We then get $HMM_1=\lambda_1=(\pi_1, A_1, c_1, \mu_1, \Sigma_1)$, $HMM_2=\lambda_2=(\pi_2, A_2, c_2, \mu_2, \Sigma_2)$ …, for the corresponding tool wear states. When all the HMMs are trained, we obtain the HMM represented by the parameters: i.e. $HMMs=(\lambda_1, \lambda_2...)$. The Viterbi algorithm is then implemented to estimate the most probable tool wear state sequence. For an observation feature set ***Y***, $p_i = p(Y \mid \lambda_i)$, $i = 1,2,...$ is calculated, and the λ_i that gives the maximum posterior probability of the features $\{y_i\}$ is decided to represent that state. This decision process is made by Viterbi Algorithm (Rabiner 1989).

The classification rate is defined as:

$$\frac{Correctly\ classified\ samples}{Misclassifieds\ \text{samples} + Correctly\ classified\ samples} \times 100\% \tag{24}$$

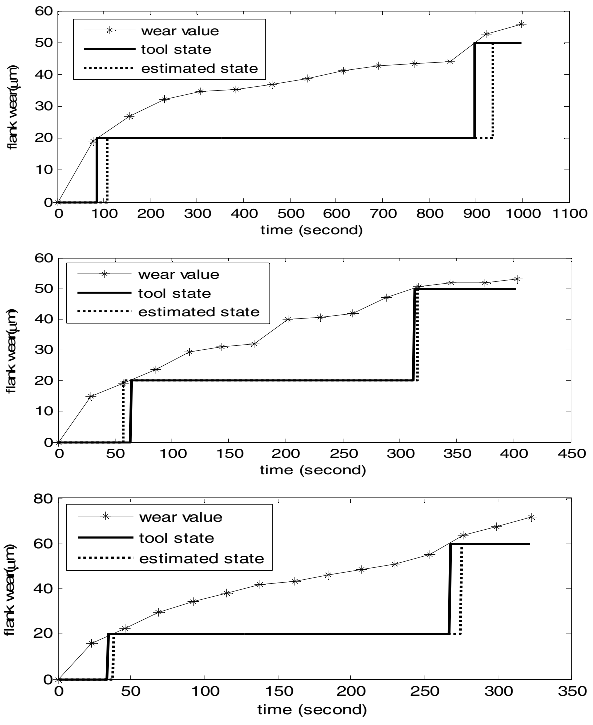
In the multi-rate modeling (Fish et al. 2003), the classification of the tool states is also constrained to the left-right direction, that is, tool states change from initial to progressive and then to accelerated region but can not change back. The tool state may pre-enter into the next state because of the high noise level in the signal in micro machining. We eliminate this situation by introducing a 15-point moving medium filter. In the HMM recognition, we hold the consecutive 15-state estimations but not decide the state immediately, and then find the medium of these 15 estimates. This medium state is our final tool wear state. The corresponding classification rates are listed in Table 7 for copper and Table 8 for steel.

Table 7. Classification rate of copper with 3- state HMMs

Classification rate of the different tests (%)														
Test 1	Test 2	Test 3	Test 5	Test 6	Test 7	Test 8	Test 9	Test 11	Test 12	Test 14	Test 15	Test 16	Test 17	Average
89.5	95.3	96.7	88.5	95.3	89.4	96.8	94.1	97.3	96.5	84.0	93.8	90.1	87.2	92.5

Table 8. Classification rate of steel with 3-state HMMs

Classification rate of the different tests (%)								
Test 18	Test 19	Test 21	Test 22	Test 23	Test 25	Test 26	Test 27	Average
88.3	90.2	89.9	84.6	91.2	95.6	93.7	89.5	90.4



Test 5, Copper
Tool: Φ500 µm
Spindle speed: 20,000 rpm
DOC: 60µm
Radial DOC: 75µm
Feed rate: 150mm/min

Test 16, Copper
Spindle speed: 18,000 rpm
DOC: 150µm
Radial DOC: 225µm
Feed rate: 150mm/min

Test 27, Steel T4
Spindle speed: 18,000 rpm
DOC: 60µm
Radial DOC: 80µm
Feed rate: 100mm/min

Fig.18. HMMs tool states recognition

Case studies are shown in Fig. 18 for the data set from test 5 and test 16 on copper and test 27 on steel.

9.5 Generalization of the HMM-Based Algorithm for TCM

The above discussion assumes that the tool has three states: initial wear, progressive wear and accelerated wear states. As a matter of fact, the HMM decision is based on the similarity measurement of signal patterns of different wear levels. It may be generalized to include more tool states, which emit features that represent corresponding states while also differentiate them from others. It can be easily implemented with the HMMs - just train these states with separate HMMs and add them to the current HMMs. The augmented HMMs can then estimate more states.

Another potential is the application of HMMs for prognostics. Note that there is a by-product of the Viterbi Coding whereby we estimate the possibilities of the three states while only choosing the highest one. We may apply this with the confidence measurement.

$$\text{Confidence of state } C_{s_i} = \frac{p_{s_i}}{\sum_{i=1}^{N} p_{s_i}} \times 100\%, \quad (N: \textit{number of states}) \tag{26}$$

For example, if our confidence of the medium wear state exceeds a certain level (e.g. 80%), we can be more certain in predicting that the tool will wear out and lose effect soon.

10 Conclusion

In this chapter, an approach is proposed based on noise-robust continuous HMM for tool wear multi-category classification in micro-milling. In micro-machining, the signal noise has wide harmonics and non-Gaussian distribution. The traditional denoising methods, based on Gaussian noise assumption, are not effective. The proposed approach assumes that both the Gaussian noise and non-Gaussian noises are signal sources and solve the signal denoising problem as blind source separation (BSS). The results show that FastICA separates both Gaussian and non-Gaussian noise sources. It is superior to signal denoising with wavelet thresholding.

In the HMM, the most discriminant features are extracted according to their FDR scores. To overcome the drawbacks of premature state changing due to noisy signal or signal processing noise in the left-right HMMs, the state sequence evolution is constrained and decision of the states are made with a medium filter after Vertibi estimation. The effectiveness of the approach has been evaluated under different conditions, e.g. different working conditions, different work piece materials, and variations of observation sequence length. The experimental results indicate that an average recognition rate of as high as 92.5% (ranging from 84.0%-97.3%) and 90.5% (ranging from: 84.6%-95.6%) can be achieved for copper and steel respectively. Therefore, the approach based on continuous HMM proposed is highly effective for micro-milling tool wear monitoring. It may also be generalized and implemented to include more states for TCM and can also be used for prognostics of tool life.

References

Bell, A.J., Sejnowski, T.J.: An information-maximisation approach to blind separation and blind deconvolution. Neural computation 7(6), 1004–1034 (1995)

Byrne, G., et al.: Tool condition monitoring (TCM)—the status of research and industrial application. Annals of CIRP 44(2), 541–567 (1995)

Byrne, G., Dornfeld, D., Denkena, B.: Advancing Cutting Technology, Annals of CIRP Keynotes (2003)

Cardoso, J.F.: Blind signal separation: statistical principles. Proceedings of the IEEE 9(10), 2009–2025 (1998)

Donoho, D.L.: De-noising by soft-thresholding. IEEE Trans. Inf. Theory 41(3), 613–627 (1994)

Du, R., Elbestawi, M.A., Wu, S.M.: Computer automated monitoring of manufacturing processes: Part 2, applications, Transactions of ASME. Journal of Engineering for Industry (1995)

Fish, R.K., et al.: Multilevel Classification of Milling Tool Wear with Confidence Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(1) (2003)

Fraley, C., Raftery, A.E.: How many clusters? Which clustering method? -Answers via model-based cluster analysis. The Computer Journal 41, 578–588 (1998)

Gelle, G., Colas, M.: Blind source searation: A tool for rotating machine monitoring by vibration analysis? Journal of Sound and Vibration 248(5), 865–885 (2001)

Goebel, K., Yan, W.: Feature Selection for Tool Wear Diagnosis Using Soft Computing Techniques, GE Technical Information Series, 2001CRD011 (2001)

Heck, L.P., Chou, K.C.: Gaussian Mixture Model Classifiers for Machine Monitoring, ICASSP 1994. In: IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 6, pp. VI/133–136 (1994)

Hong, G.S., Rahman, M., Zhou, Q.: Using neural network for tool condition monitoring based on wavelet decomposition. Int. J. Mach. Tools Manufacture 36(5), 551–566 (1996)

Hyvärinen, A.: Fast and robust fixed-point algorithms for independent component analysis. IEEE Transactions on Neural Networks 10, 626–634 (1999)

Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley & Sons, Chichester (2001)

International Standard Organization (ISO 8688-2: 1989) Tool life testing in milling -Part 2: End milling (1989)

Kassim, A.A., Zhu, M., Mannan, M.A.: Tool condition classification using Hidden Markov Model based on fractal analysis of machined surface textures. Machine Vision and Applications 17, 327–336 (2006)

Kumar, S.A., Ravindra, H.V., Srinivasa, Y.G.: In-process Tool Wear Monitoring through Time Series Modeling and Pattern Recognition. International Journal of Production Research 35(3), 739–751 (1997)

Li, X., Tso, S.K., Wang, J.: Real time tool condition monitoring using wavelet transforms & fuzzy techniques. IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews 31(3), 352–357 (2000)

Li, D., Douglas, O.S.: Speech processing: A Dynamic and Object-oriented Approach, Marcel Dekker, Inc. (2003)

Li, S., Elbestawi, M.A.: Fuzzy Clustering for Automated Tool Conditon Monitoring in Machining. Mechanical Systems and Signal Processing 10(5), 533–550 (1996)

Liang, S.Y., Dornfeld, D.A.: Detection of Cutting Tool-wear Using Adaptive Time Series Modeling of Acoustic Emission Signal, Sensors for Manufacturing, ASME, pp. 27–38 (1987)

Liu, X., et al.: The Mechanics of Machining at the Micro-Scale: Assessment of the Current State of the Science. Journal of Manufacturing Science and Engineering 126(4), 666–677 (2004)

Mallat, S.G.A.: Wavelet Tour of Signal Processing. Academic Press, San Diego (1998)

Niu, Y.M., Wong, Y.S., Hong, G.S.: Multi-category classification of tool conditions using wavelet packets and ART2 network. ASME J. of Manufacturing Science and Technology 120, 807–815 (1998)

Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77(2), 257–286 (1989)

Rahman, R., Kumar, A.S., Prakash, J.R.R.: Micro-milling of pure copper. International Journal of Materials Processing 16, 39–43 (2001)

Roan, M.J., Erling, J.G., Sibul, L.H.: A new, non-linear, adaptive, blind source separation approach to geartooth failure detection and analysis. Mechanical Systems & Signal Processing 16(5), 719–740 (2002)

Saglam, H., Unuvar, A.: Tool Condition Monitoring in Milling on cutting forces by a Neural Network. Int. J. Production Research 41(7), 1519–1532 (2003)

Serviere, C., Fabry, P.: Blind source separation of noisy harmonic signals for rotating machine diagnosis. Journal of Sound and Vibration 272(1-2), 317–339 (2004)

Shaw, M.C.: Metal Cutting Principles, 2nd edn. Oxford University Press, Oxford, UK (2005)

Sun, J., et al.: Identification of feature set for effective tool condition monitoring by acoustic emission sensing. International Journal of Production Research 42(5), 901–918 (2004)

Tansel, I.N., et al.: Micro-end-milling—I. wear and breakage. International Journal of Machine Tools & Manufacture 38, 1419–1436 (1998)

Tansel, I.N., et al.: Tool wear estimation in micro-machining. Part I: tool usage–cutting force relationship. Int. J. Mach. Tools Manufacture 40, 599–608 (2000)

Tansel, I.N., et al.: Micro-end-milling—III. wear estimation and tool breakage detection using acoustic emission signals. International Journal of Machine Tools & Manufacture 38, 1449–1466 (1998)

Theodoridis, S., Koutroumbas, K.: Pattern Recognition, vol. 2. Academic Press, London (2003)

Tian, X., et al.: Gearbox fault diagnosis using independent component analysis in the frequency domain and wavelet filtering. In: IEEE ICASSP 2003, pp. II245–248 (2003)

Wang, L., Mehrabi, M.G., Elijah, K.A.: Hidden Markov Model-based Tool Wear Monitoring in Turning. ASME Journal of Manufacturing Science and Engineering 124, 652–658 (2002)

Wang, W.H., et al.: Sensor fusion for on-line tool condition monitoring in milling. International Journal of Production Research (in press, 2007)

Wu, Y., Escande, P., Du, R.: A New Method for Real-Time Tool Condition Monitoring in Transfer Machining Stations. ASME Journal of Manufacturing Science and Engineering 123, 339–347 (2001)

Zhu, K.P., Hong, G.S., Wong, Y.S.: Cutting Force Denoising in Micro-milling Tool Condition Monitoring. International Journal of Production Research (in press, 2007)

Dynamically Self-generated Fuzzy Neural Networks with Industry Applications

Meng Joo Er and Yi Zhou

School of Electrical and Electronic Engineering, Nanyang Technological University, 50, Nanyang Avenue, Singapore 639798
emjer@ntu.edu.sg
School of Electrical and Electronic Engineering, Singapore Polytechnic, 500, Dover Road, Singapore 139651
zhouyi@sp.edu.sg

1 Introduction

Over the last few decades, fuzzy logic has been shown as a powerful methodology for dealing with imprecision and nonlinearity efficiently. Applications can be found in a wide context ranging from medicine to finance, from human factors to consumer products, from vehicle control to computational linguistics, and so on (Wang 1997; Dubois and Prade 2000; Passino and Yurkovich 1998; Jang et al. 1997; Sugeno 1985; Pedrycz 1993). However, one of the shortcomings of fuzzy logic is the lack of systematic design. To circumvent this problem, fuzzy logic is usually combined with Neural Networks (NNs) by virtue of the learning capability of NNs. NNs are networks of highly interconnected neural computing elements that have the ability of responding to input stimuli and learning to adapt to the environment. Both fuzzy systems and NNs are dynamic and parallel processing systems that estimate input-output functions (Mitra and Hayashi 2000). The merits of both fuzzy and neural systems can be integrated in Fuzzy Neural Networks (FNNs) (Lee and Lee 1974, 1975; Pal and Mitra 1999; Zanchettin and Ludermir 2003). Therefore, the integration of fuzzy and neural systems leads to a symbiotic relationship in which fuzzy systems provide a powerful framework for expert knowledge representation, while NNs provide learning capabilities.

The main issues for designing FNNs are structure identification and parameter estimation. Input-space partitioning and the number of fuzzy neurons are determined for structure identification while both premise and consequent parameters are defined for parameter estimation. A systematic machine learning architecture for generating FNNs is proposed in this chapter.

The objectives of this chapter are to develop FNNs by various machine learning techniques so that these systems can be used for online identification and control process in dealing with nonlinear uncertain systems. The DSGFNN architecture combines Fuzzy Neural Networks (FNN) with other machine learning techniques, e.g. Supervised Learning (SL) and Reinforcement Learning (RL). More specifically, two algorithms, namely Dynamic Fuzzy Neural Networks (DFNN) and Dynamic Self-Generated Fuzzy Q-Learning (DSGFQL) with a wide range of industry applications have been developed in this chapter. The DSGFNN architecture is based on extended Radial Basis Function (RBF) NNs, which are functionally equivalent to Takagi-Sugeno-Kang (TSK) fuzzy systems. These algorithms comprise four parts: 1) Criteria of rules generation; 2) Allocation of premise parameters; 3) Determination of consequent parameters and 4) Pruning technology.

In general, SL implies that the information source used for learning is a direct teacher, which provides, at each time step, the correct control action to be applied by the learner. Typically, these learning methods are based on an input-output set of training data, based upon which we have to minimize errors between the teacher's actions and the learner's responses. The DFNN has been developed for SL-based problems. Firstly, both the input and output data are to be checked by an ε-completeness criterion and the system error criterion for deciding whether to generate a new rule. Premise parameters are allocated and adjusted to fulfill the ε-completeness criterion while consequent parameters are to be determined by Linear Least Squares and Recursive Least Squares. The Error Reduction Ratio method is adopted for rule pruning. In short, automatic generation of FNNs has been demonstrated.

However, the teacher or the training data are not always available especially when a human being has little knowledge about the system or the system is uncertain. In those situations, RL is preferred over SL as RL is a learning process that does not need any supervisor to tell the learner what actions to take. RL dates back to the early days of cybernetics and works in statistics, psychology, neuroscience and computer science (Sutton and Barto 1998). In the last few decades, it has attracted rapidly increasing interest in the machine learning and artificial intelligence communities. RL revolves around solving a problem faced by an agent that must learn through trial-and-error interactions with a dynamic environment. Compared with SL, RL does not need a supervisor to tell the learner what to do. Instead, RL uses only simple evaluative or critic information available for learning. Through RL, those state-action pairs which achieve positive reward will be encouraged in future selections while those which produce negative reward will be discouraged. The DSGFQL has been developed for RL problems, in which fuzzy rules are created, adjusted and deleted according to reward evaluations and consequent parameters are determined via the Fuzzy Q-Learning (FQL) approach in (Jouffe 1998).

Distinguished from other existing methodologies for generating FNNs, the proposed architecture can automatically create, adjust and delete fuzzy rules dynamically without any *priori* knowledge. The two approaches have been applied widely to various industry problems. In this chapter, the DFNN has been applied for real-time control of a Selectively Compliance Assembly Robot Arm (SCARA) which is a SEIKO D-TRAN 3000 series robot. For RL problems, the DSGFQL approaches have been adopted for a wall-following task by a mobile robot which can be potentially applied as an auto-cleaner and an unmanned vehicle. Experiments and simulation studies demonstrate the superiority of the proposed architecture and algorithms over other existing methods and the suitability for industrial applications by the proposed approach.

2 Architecture of the DSGFNN

The new structure based on extended RBF NNs to perform TSK model-based fuzzy system is shown in Fig. 1.

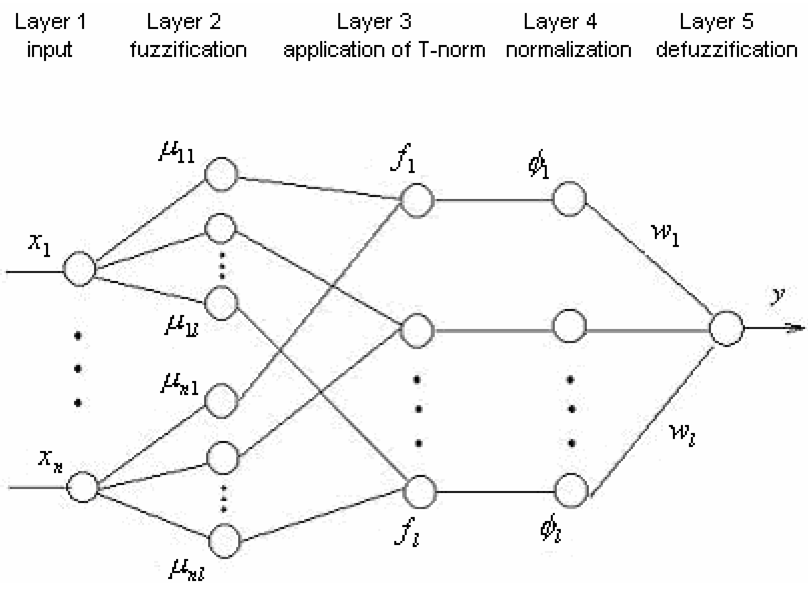


Fig. 1. Structure of the DSGFNN system

Layer one is an input layer and layer two is a fuzzification layer which evaluates the Membership Functions (MFs) of the input variables. The MF is chosen as a Gaussian function and each input variable $x_i (i = 1, 2, ... N)$ has L MFs given by

$$u_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right], i = 1, 2, ..., N \quad j = 1, 2, ... L \tag{1}$$

where u_{ij} is the jth MF of x_i, while c_{ij} and σ_{ij} are the center and width of the jth Gaussian MF of x_i respectively. Layer three is a rule layer. The output of the jth rule $R(j = 1, 2, ... L)$ in layer 3 is given by

$$f_j(x_1, x_2, ..., x_N) = \exp\left[-\sum \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right], \ j = 1, 2, ... L \tag{2}$$

if multiplication is adopted for the T-norm operator. Normalization takes place in layer 4 and we have

$$\phi_j = \frac{f_j}{\sum_{i=1}^{L} f_i}, j = 1, 2, ... L \tag{3}$$

Lastly, nodes of layer five define output variables. If the Center-Of-Gravity (COG) method is performed for defuzzification, the output variable, as a weighted summation of the incoming signals, is given by

$$y = \sum_{j=1}^{L} \phi_j \omega_j \tag{4}$$

where y is the output variable and w_k is the THEN-part (consequent parameters) or connection weight of the kth rule. For the TSK model:

$$\omega_k = \alpha_{k0} + \alpha_{k1} x_1 + \cdots \alpha_{kN} x_N \,,\; k = 1, 2, ..., L \tag{5}$$

When the consequent parameters are real constants i.e.,

$$w_k = \alpha_k , k = 1, 2, \cdots, L \,, \tag{6}$$

we obtain the simplified fuzzy model and we call it the S-model.

3 Dynamic Fuzzy Neural Networks for Supervised Learning

3.1 Criteria of Rule Generation

3.1.1 ε -Completeness

From the point of view of RBF neural networks, an RBF unit is a class of representation over a region defined in the input space. If a new sample falls within the accommodation boundary of one of the currently existing RBF units, which has the same class representation as that of the new sample, the DFNN will not generate a new RBF unit but accommodate the new sample by updating the parameters of existing RBF units. If $d_{\min} > k_d$, a new rule should be considered. Otherwise, the observation (X_i, t_i) can be represented by the existing nearest RBF unit. Here, k_d is the effective radius of accommodation boundary. The ε -completeness criterion in (Jang 1993), which is termed accommodation boundary in (Wu and Er 2000; Wu et al. 2001), is adopted as follows:

For any input in the operating range, there exists at least one fuzzy rule so that the match degree (or firing strength) is no less than ε .

3.1.2 System Errors

It is not essential to only consider the ε -Completeness; therefore, the output error is adopted to recruit new rules in the DFNN which is a SL approach. The error criterion can be described as follows:

For the ith observation (X_i, O_i), where X_i is the input vector and O_i is the desired output, compute the overall DFNN output y_i of the existing structure according to Eq. (4).

Define:

$$\| e_i \| = \| O_i - y_i \| \tag{7}$$

If $\| e_i \| > k_e$, a new rule should be considered. Here, the value of k_e is chosen a priori according to the desired accuracy of the DFNN.

3.2 Rule Generation

Once a new rule is to be created, the new center and width can be allocated as follows (Lu et al. 1997; Wu and Er 2000):

$$\begin{aligned} C_i &= X_i \\ \sigma_i &= k \times d_{\min} \end{aligned} \tag{8}$$

where k is an overlap factor that determines the overlap of responses of the RBF sets. In general, the value of k is set to 1.05~1.2. The new MF can also be assigned as that in (Wu et al. 2001)

$$\begin{aligned} C_i &= X_i \\ \sigma_i &= \frac{\max\{|c_i - c_{i-1}|, |c_i - c_{i+1}|\}}{\sqrt{\ln(1/\varepsilon)}} \end{aligned} \tag{9}$$

3.3 Determination of Consequent Parameters

This problem can be solved by the well-known Linear Least Square method and it has been shown that the learning formulation and the least square problem provide a simple but efficient procedure for determining the consequent parameters and can be used for real-time applications (Chen 1996). As an alternative, Recursive Least Squares, also known as the Kalman Filter algorithm (Lin and Lee 1996) can be employed to determine the weights (Chen 1996; Lin and Lee 1996; Takagi and Sugeno 1985; Jang 1993; Chak et al. 1998; Er et al. 2003).

3.4 Pruning Technology

The trade-off between goodness of fit and simplicity is a fundamental concern for generation of FNNs. If there are too few rules, the system will underperform. However, if there are too many rules, over-fitting will occur. Therefore, redundant rules should be removed from the system. There are several pruning technologies available, e.g. the Sensitivity Calculation Method (Reed 1993), Weight Decay Method (Reed 1993), Competitive Learning (Lin and Lee 1991) and Minimal Output Method (Lu et al. 1997). In this chapter, the Error Reduction Ratio method presented in (Chen et al. 1991) is adopted as a pruning strategy (Wu and Er 2000). The flowchart of the algorithm is illustrated in Fig. 2.

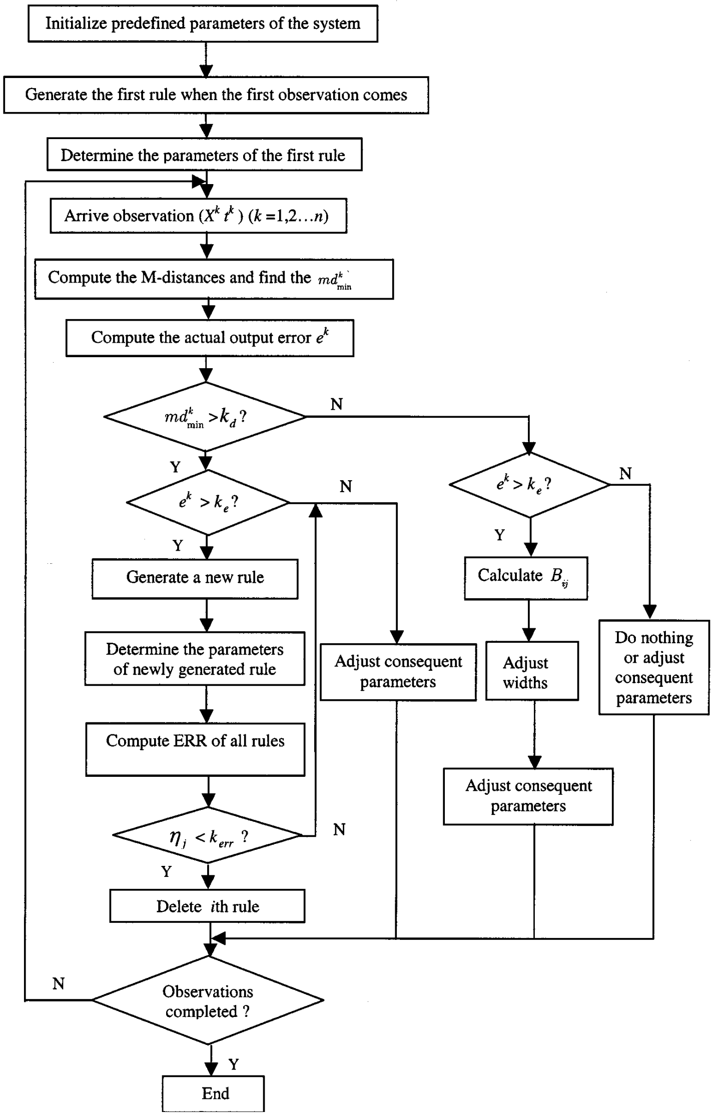

Initialize predefined parameters of the system
Generate the first rule when the first observation comes
Determine the parameters of the first rule
Arrive observation $(X^k\ t^k)$ $(k=1,2\ldots n)$
Compute the M-distances and find the $md^k_{\min}$
Compute the actual output error e^k
$md^k_{\min} > k_d$?
N
Y
$e^k > k_e$?
N
Y
Generate a new rule
Determine the parameters of newly generated rule
Compute ERR of all rules
$\eta_j < k_{err}$?
N
Y
Delete jth rule
Adjust consequent parameters
$e^k > k_e$?
N
Y
Calculate B_{ij}
Adjust widths
Adjust consequent parameters
Do nothing or adjust consequent parameters
N
Observations completed ?
Y
End

Fig. 2. Flowchart of DFNN learning algorithm

In this chapter, the DFNN which can automatically generate the structure and determine the parameters of FNNs via SL is introduced. Details of the determination of consequent parameters and pruning algorithms are not presented in this chapter due to the page limit and readers can refer to (Wu and Er 2000; Wu et al. 2001; Er et al. 2003) for further information.

3.5 Real-Time DFNN Control of a SCARA

3.5.1 Overview of SEIKO TT-3000 SCARA

The SEIKO D-TRAN 3000 Series robot used in this work is a four-axis, closed-loop DC servo Selectively Compliance Assembly Robot Arm (SCARA). Each of the four axes provides a different motion and contributes to one degree of freedom of the robot (see Fig. 3). The main characteristics of the TT-3000 SCARA are its high precision, repeatability and speed. The basic SCARA geometry is realised by arranging two revolute joints (for simplicity, they will be called Joint T1 and Joint T2 herein) and one prismatic joint (for simplicity, it will be called Joint Z herein) in such a way that all axes of motion are in parallel. The acronym SCARA characterises mechanical features of a structure offering high stiffness to vertical loads and compliance to horizontal loads.

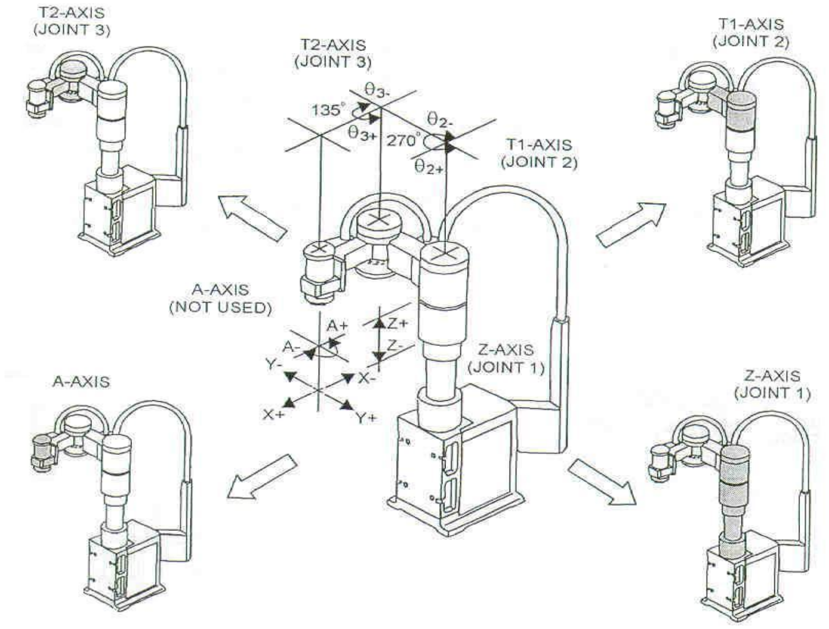


Fig. 3. The SEIKO TT-3000 SCARA

The movement of the end-effector to track a desired trajectory, at a particular velocity, thus requires a complex set of torque functions to be applied to the actuating system of the robot at each joint. The dynamic model of the TT-3000 SCARA has been developed in (Zhang 1996), with most of its parameters determined and verified through experiments. Based on this known mathematical model of the robot, training of the DFNN controller was carried out using MATLAB simulation tools.

3.5.2 Motion Control

A two-joint control structure is used in this work where motion control of Joint T1 and Joint T2 will be executed simultaneously. In this scheme, Joint Z will not be controlled for simplicity. Motion control of the robot arm is shown in Fig. 4. The control

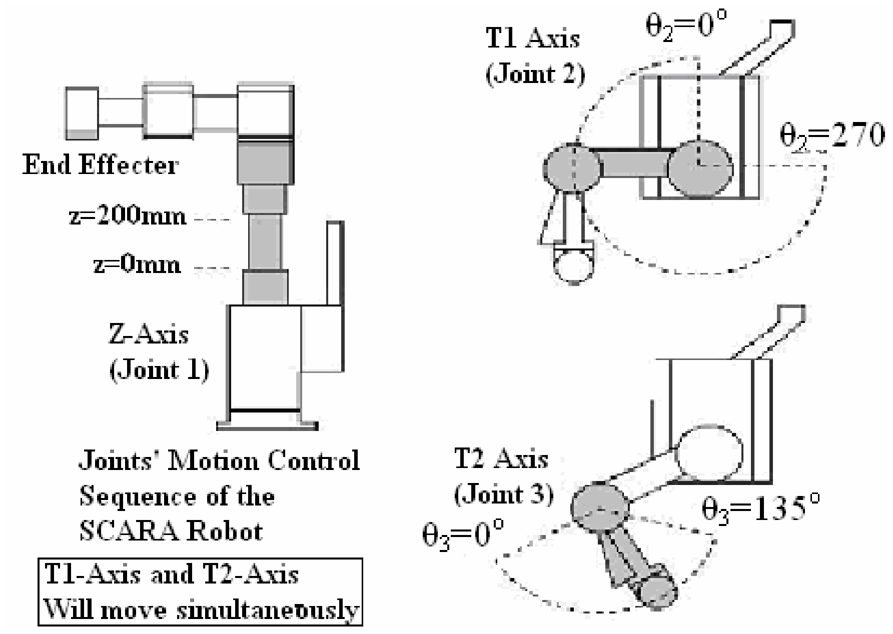


Fig. 4. Joint motion of the SCARA robot

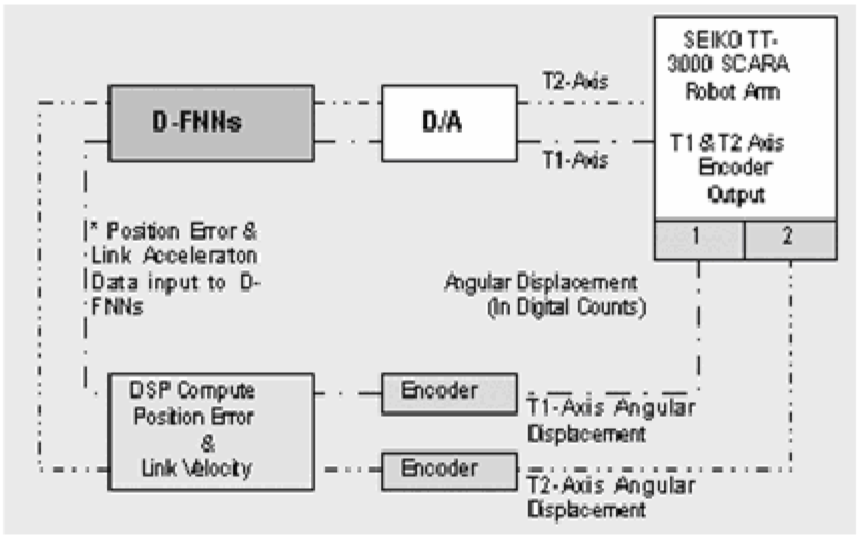

D-FNNs
D/A
T2-Axis
T1-Axis
SEIKO TT-3000 SCARA Robot Arm
T1 & T2 Axis Encoder Output
1
2
* Position Error & Link Acceleraton Data input to D-FNNs
Angular Displacement (In Digital Counts)
DSP Compute Position Error & Link Velocity
Encoder
Encoder
T1-Axis Angular Displacement
T2-Axis Angular Displacement

Fig. 5. Control structure employing D-FNN

system of the two joints is considered a Multiple-Input Multiple-Output (MIMO) system. The degree of difficulty will increase if the number of joints to be controlled simultaneously increases. The implementation of this control architecture on the target system is illustrated in Fig.5.

3.5.3 Design of D-FNN Controller

The proposed adaptive fuzzy neural control scheme is depicted in the following figure.

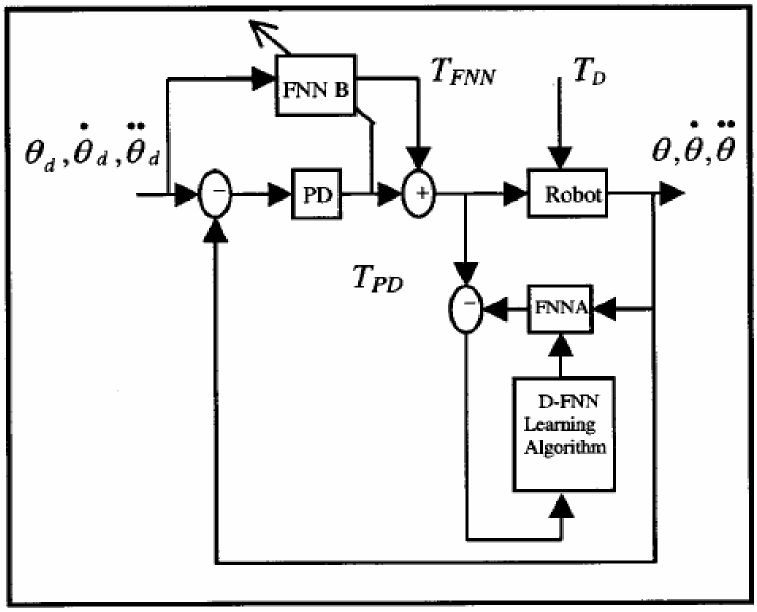

FNN B
T_{FNN}
T_D
$\theta_d, \dot{\theta}_d, \ddot{\theta}_d$
$\theta, \dot{\theta}, \ddot{\theta}$
PD
+
Robot
T_{PD}
FNNA
D-FNN Learning Algorithm

Fig. 6. Adaptive Fuzzy Neural Control System

The basic idea of this approach is to learn the manipulator's inverse characteristics and to use the inverse dynamic model to generate the compensated control signal. There are two FNNs used in this control system. FNN A is employed for weights training of the system whereas FNN B is used to generate the appropriate control signal with the trained weight structure from FNN A. The proposed control algorithm is given by

$$\tau = \tau_{PD} + \tau_{FNNB} \tag{11}$$

where τ is the required control torque, τ_{PD} is the torque generated by the PD controller and τ_{FNNB} is the torque generated by FNN B.

The inverse robot model is obtained by FNN A, employing the DFNN learning algorithm. In online learning, FNN A is trained during real-time control of the manipulator. FNN B is a duplicate copy of FNN A, but its weights will be further adjusted by the error signal τ_{PD} as the controller is in operation. This is to compensate for modelling errors and unmodelled disturbances. The parallel-connected conventional PD controller also caters for faster and more accurate tracking performance.

The gist of weight training is to train FNN B such that the squared error between the desired torque and the actual torque, i.e.

$$E = \frac{1}{2}(\tau - \tau_{FNNB})^2 = \frac{1}{2}(\tau_{PD})^2 \tag{12}$$

is minimised.

To achieve fast weight adjustment, the gradient method is used to minimise E. The weight is adjusted as follows:

$$\Delta W = W(new) - W(old) = -\eta \frac{\partial E}{\partial W} \tag{13}$$

where $\eta > 0$ is the learning rate.

Using the chain rule, we have

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial \tau_{FNNB}} \frac{\partial \tau_{FNNB}}{\partial W} \tag{14}$$

When $y = \tau_{FNNB}$, we have

$$\frac{\partial E}{\partial \tau_{FNNB}} = -\left(\tau - \tau_{FNNB}\right) \tag{15}$$

$$\frac{\partial \tau_{FNNB}}{\partial W} = \Phi \tag{16}$$

Thus, Eq. (13) can be rewritten as

$$\Delta W = \eta\left(\tau - \tau_{FNNB}\right)\Phi = \eta \tau_{PD} \Phi \tag{17}$$

The DFNN was implemented using the dSPACE System DS1102 with the digital signal processor (DSP) TMS320C31 on board (Er and Gao 2003). The sampling time was chosen to be 1 ms. The approach of rapid prototyping as depicted in Fig. 7 was adopted in the implementation of the DFNN. This process allows one to conceptualize solutions using a block-diagram modeling environment and has a preview of the system performance prior to actual implementation.

Once the desired results are achieved, the Real-Time Workshop was used to generate the object code and download it onto the DS1102 DSP controller board. In essence, the process reduces algorithm coding to an automated process, which includes coding, compiling, linking and downloading to target hardware. The ControlDesk-dSPACE software was used to control the SCARA robot. A graphical user interface was also built to monitor output signals from the SCARA robot in real time. In addition, online changes to the control parameters are made possible. This facilitates fine tuning of the controller.

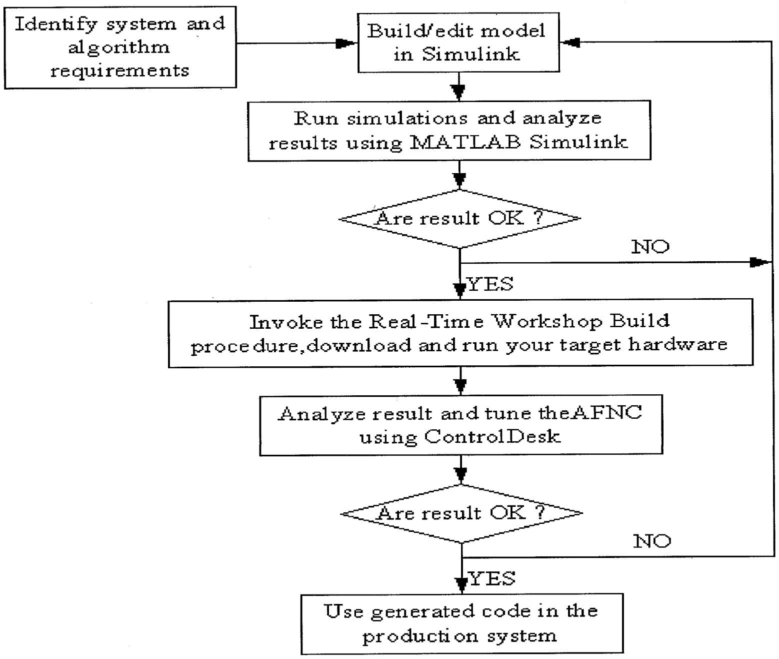

Identify system and algorithm requirements
Build/edit model in Simulink
Run simulations and analyze results using MATLAB Simulink
Are result OK ?
NO
YES
Invoke the Real-Time Workshop Build procedure,download and run your target hardware
Analyze result and tune theAFNC using ControlDesk
Are result OK ?
NO
YES
Use generated code in the production system

Fig. 7. Rapid prototyping process

3.5.4 Comparison Studies

For simplicity, we only control T1 and T2 axes of the SCARA, denoted as θ_1 and θ_2. The desired trajectories for the two axes are given by $\theta_{d1} = 1.5\sin(2\pi t)$ rad and $\theta_{d2} = 0.7\cos(2\pi t)$ rad respectively. Initial conditions are given as $\theta_1 = \theta_2 = 0$ rad, $\dot{\theta}_1 = \dot{\theta}_2 = 0$ rad/s and $\ddot{\theta}_1 = \ddot{\theta}_2 = 0$ rad/s. In a real system, there always exist uncertainties and disturbances. Hence, we deliberately introduce disturbances $T_{d1} = 100\sin(2\pi t)$ Nm and $T_{d2} = 50\sin(2\pi t)$ Nm, which are comparable to the control torques of the robot manipulator. The gains of the conventional PD controller are selected as $K_p = diag[25,25]$ and $K_v = diag[7,7]$.

Fig. 8 shows experimental results of trajectory tracking by the DFNN controller. The results clearly demonstrate that the two joints were able to track the desired trajectories from third trial onwards. It should be highlighted that for θ_2 (T2 axis), whose initial position was set to be 100% error from the desired initial joint position, i.e., $\theta_2(0) = 0$, the joint was still able to follow with the desired trajectory very quickly without any overshoot. The performance of the control system is greatly improved after incorporating the FNN controller with the conventional PD controller.

The proposed adaptive fuzzy neural control scheme has some advantages over normal fuzzy control and NNs control schemes. For example, in normal fuzzy controller

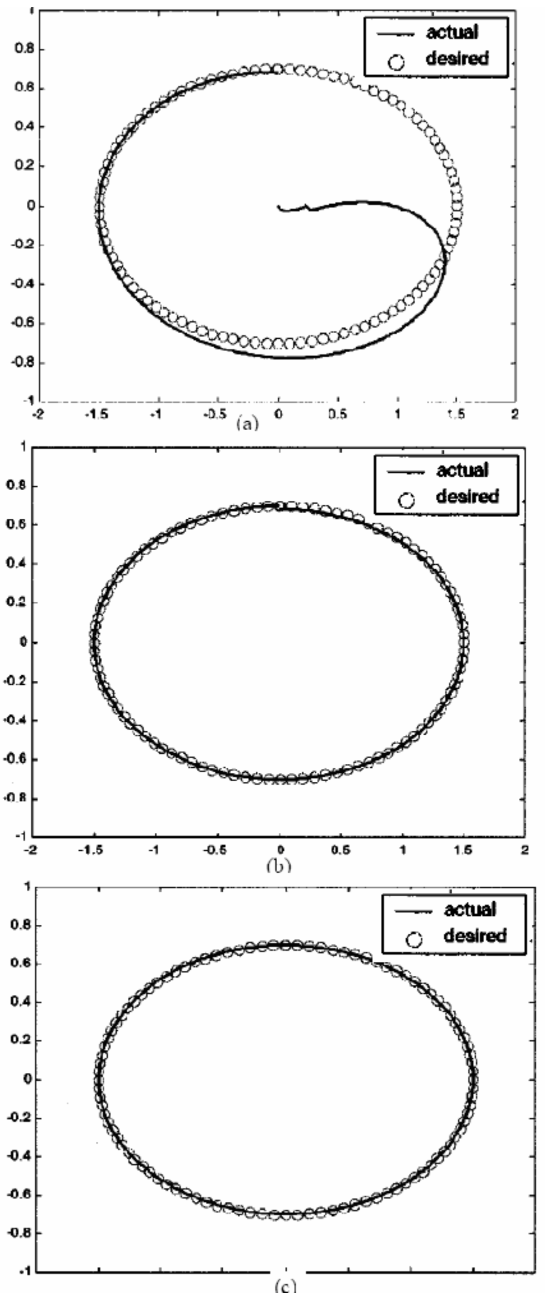


Fig. 8. Tracking Results for AFNC. (a) First trial. (b) Second trial. (c) Third trial onwards.

design, certain controller parameters must be tuned by trials and errors. Such parameters include scalars (or gains) for its input data and output data, the number of MFs, the width of a single MF and the number of control rules. On the contrary, the adaptive fuzzy neural controller based on dynamic model estimation will allow the parameters to be tuned automatically.

In addition, the performance of the DFNN is also considered to be excellent. Table 1 tabulates transient and steady-state control results of the DFNN compared with a conventional PD controller and the AFC of (Er and Chin 2000). The desired trajectory is a straight-line passing through the origin on the plane. Table 1 shows that the DFNN can achieve better transient and steady-state control results compared with the rest and with smaller torque.

Table 1. Performance Comparisons of DFNN with PD and AFC Controllers

		Steady State Error (rad)	Overshoot (%)	Rise Time (Sec)	Max Control Torque (Nm)
PD	θ_1	0.050	9.3	0.287	312.7
$K_p = 6000; K_v = 100$	θ_2	0.018	7.1	0.254	109.9
AFC(Er and Chin 2000)	θ_1	0.022	0	0.237	425.9
$K_p = 25; K_v = 35$	θ_2	0.025	0	0.218	255.6
DFNN	θ_1	0.0001	0	0.001	298.0
$K_p = 25; K_v = 7$	θ_2	0.0003	0	0.001	105.9

4 DSGFQL for Reinforcement Learning

4.1 Recruitment of Fuzzy Neurons

In the proposed algorithm, the ε -Completeness criterion is to be considered for generating new rules which is similar to the DFNN. If $d_{\min} > k_d$, a new fuzzy rule should be created to fulfill the input-space with a certain degree. If the Euclidean distance does not pass the similarity test as mentioned in (Er and Deng 2004) and (Juang 2005), there will be no new MF created. Otherwise, a new Gaussian MF is allocated with

$$\begin{aligned} & c_i(L+1) = x_i \\ & \sigma_{ik} = \frac{\max\left(\left|c_{ik} - c_{i(k-1)}\right|, \left|c_{i(k+1)} - c_{ik}\right|\right)}{\sqrt{\ln(1/k_d)}}, \text{ k=1,2,...,L+1} \end{aligned} \tag{18}$$

4.2 Global and Local Reward Evaluations

As no instructive teaching data are avaiable, the system error criterion in the DFNN is not valid for RL problems. A reward function r(t) is adopted to evaluate the performance of the system. A positive reward is given if the system works well and a negative value is given if the performance is unsatisfactory. As each fuzzy neuron is with a degree of firing strength, the local performance of the fuzzy neuron is evaluated by the local reward as in (Ritthiprava et al. 2004) as follows:

$$r_{local}^{j}(t) = \phi_j r(t), j = 1, 2, ..., L \tag{19}$$

and the average reward of a period of time is given as follows:

$$r_{avg} = \frac{\sum_{t=k}^{k+T} r(t)}{T} \tag{20}$$

where T is the period of time.

4.3 Adjustment of Membership Functions

If the global average reward is less than a threshold, k_g , it means that the overall performance is unsatisfactory. To resolve this problem, the weights of some good rules should be increased which means that the system will be modified by promoting the rule with the best performance, e.g. highest local reward, to a more important position. In such a case, the width of the jth fuzzy rule's MF (the one with the best local reward) will be increased as follows:

$$\sigma_{ij} \leftarrow k_p \sigma_{ij} \quad \text{i=1,2,...,N} \tag{21}$$

where k_p is slightly larger than 1.

If the local reward is larger than, k_{lh} , but smaller than a light threshold k_{ll} , a punishment will be given to the fuzzy rule by decreasing its width of the MF as follows:

$$\sigma_{ij} \leftarrow k_D \sigma_{ik} \quad \text{i=1,2,...,N} \tag{22}$$

where k_D is a positive value less than 1.

Therefore, the overall performance is improved as the rules with good contributions participate more in the system while those rules with poor contributions participate less correspondingly.

4.4 Pruning of Unsatisfactory and Redundant Fuzzy Neurons

If local reward of a fuzzy rule is less than a certain threshold, k_{lh} , the individual contributions are unsatisfactory and that fuzzy rule will be deleted. Besides the reward evaluation, firing strength should also be considered for system evaluation as it is a measurement for participation. If a fuzzy rule has very low firing strength during the entire episode or a long recent period of time, it means that this rule is unnecessary for the system. As more fuzzy rules mean more computation and longer training time, the rule whose mean firing strength over a long period of time is less than a threshold, k_f , should be deleted.

Remark: If a fuzzy rule keeps failing the light local reward check, its firing strength will be reduced by decreasing the width of its MF. When the width is reduced to a certain level, it will fail the firing strength criterion and will be deleted. The light local test gives a certain tolerance, which means the fuzzy rule is not deleted due to one slight fault. However, the fuzzy rule which does not obtain satisfactory result is still punished by being demoted and it will be deleted if it keeps failing the light local test.

Those thresholds are to be set according to the target of the training system. The flowchart of the DSGFQL approach is shown in Fig. 9 and the V(X) is the global Q-function which is updated by the FQL approach. The consequent paramaters are determined via the FQL approach. Details of the FQL can be found in (Jouffe 1998).

4.5 Wall-Following Control with a Khepera Robot

In order to apply the DSGFQL algorithm and compare it with other related methodologies, the DSGFQL approach has been applied to control a Khepera mobile robot of (K-Team 2002) for a task of obstacle avoidance. The robot employed in the experiments described in this chapter is a miniature mobile robot called Khepera (K-Team 2002) shown in Fig. 10. The Khepera mobile robot is cylindrical in shape, with 55 mm in diameter and 30 mm in height weighting only 70g. Its small size allows experiments to be performed in a small work area. The robot is supported by two lateral wheels that can rotate in both directions and two rigid pivots in the front and back.

Khepera can be remotely controlled by a computer through a serial link depicted in Fig. 11. The serial connection provides electrical power and supports fast data communication between the robot and the computer. The control system of the robot can run on the computer that reads in sensory data and gives motor commands in real time while the robot moves on a nearby environment. Alternatively, one can download the code of the control system on the processor of the robot and then disconnect the cable. The robot is equipped with two dc motors coupled with incremental sensors, eight analogue Infra-Red (IR) proximity sensors and an on-board power supply which is the same as that in (Er and Deng 2004) and is shown in Fig. 12. Each IR sensor is composed of an emitter and an independent receiver. The dedicated electronic interface uses multipliers, sample holds and operational amplifiers. This allows absolute ambient light and estimation, by reflection, of the relative position of an object to the robot to be measured. By this estimation, the distance between the robot and the obstacle can be derived.

The aim of the experiment is to design a simple controller for wall following. In order to simplify the problem, we only consider robots moving in clockwise direction at a constant speed. Thus, we only need to deal with four input variables, which are the sensor values S_i ($i = 0,\ldots,3$). All these sensor values can be normalized within the interval [0,1]. The output of the controller is the steering angle of the robot. In order for the robot to follow a wall, it must move in a straight line as much as possible while staying between a maximum distance, d_+, and a minimum distance, d_-, from that wall. The sensor value can be regarded as the distance to the wall being followed.

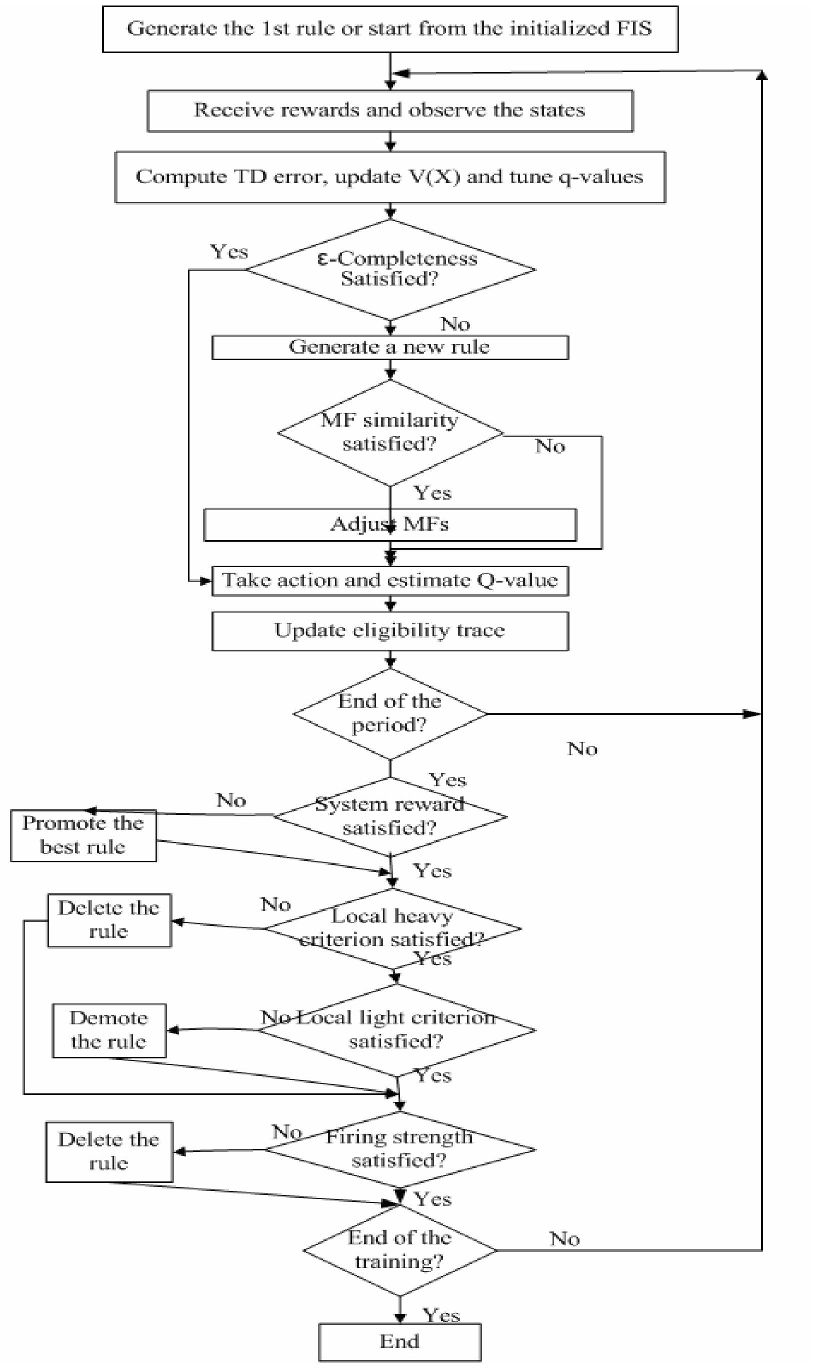

Generate the 1st rule or start from the initialized FIS
Receive rewards and observe the states
Compute TD error, update V(X) and tune q-values
ε-Completeness Satisfied?
Yes
No
Generate a new rule
MF similarity satisfied?
No
Yes
Adjust MFs
Take action and estimate Q-value
Update eligibility trace
End of the period?
No
Yes
System reward satisfied?
No
Promote the best rule
Yes
Local heavy criterion satisfied?
No
Delete the rule
Yes
Local light criterion satisfied?
No
Demote the rule
Yes
Firing strength satisfied?
No
Delete the rule
Yes
End of the training?
No
Yes
End

Fig. 9. Flowchart of the DSGFQL approach

Fig. 10. The miniature mobile robot: Khepera

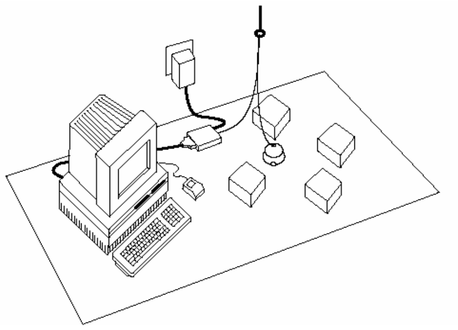


Fig. 11. The Khepera robot and its working environment

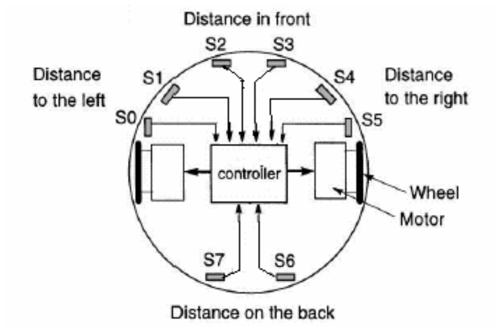


Fig. 12. Position and orientation of sensors on the Khepera

The robot receives a reward after performing every action U. The reward function depends on this action and the next situation:

$$r = \begin{cases} 0.1, & if\ (d_- < d < d_+)\, and\, (U \in [-8^0, +8^0]) \\ -3.0, & if\ (d \le d_-)\, or\, (d_+ \le d) \\ 0.0, & otherwise \end{cases} \tag{23}$$

In these experiments, d_+ =0.85 and d_- =0.15 and the set of discrete actions is A= [-30, -25, -20, -15, -10, -5, 0, 5, 10, 15, 20, 25, 30] where continuous actions will be generated by fuzzy approaches. In order to compare the different approaches systematically and find appropriate parameters, we use the simulated robot and implement these methods on the real robot. The training environment with lighted shaped walls used for a real robot and simulation studies (Nilsson 2003) are shown in Fig. 13 and Fig. 14 respectively. The training starts with a basic fuzzy controller in (Er and Deng 2004) and the wall following performance before and after training is shown in Fig. 15.

Besides the original Fuzzy Q-Learning (FQL) (Jouffe 1998), there are two extended FQL approaches, termed Dynamic Fuzzy Q-Learning (DFQL) (Er and Deng 2004) and online Clustering and Q-value based Genetic Algorithm learning schemes for Fuzzy system design (CQGAF) (Juang 2005), which can dynamically generate a structure of FNNs in recent literature. To demonstrate the superiority of the proposed approach, both DFQL and CQGAF approaches have been applied in the wall-following task as well as the original FQL.

For the FQL approach, the number of fuzzy rules needs to be pre-defined. If we define two MFs for each input, there are $2^4 = 16$ rules for the fuzzy controller. If three

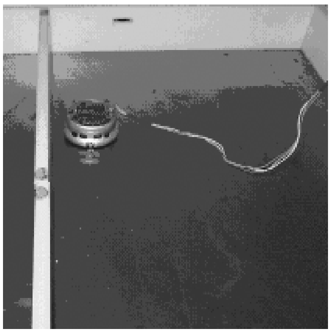


Fig. 13. Actual environment for wall-following experiments

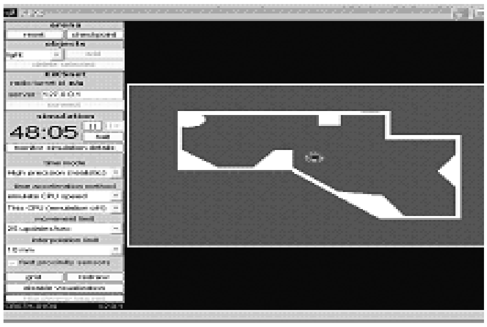


Fig. 14. Simulation environment for wall-following experiments

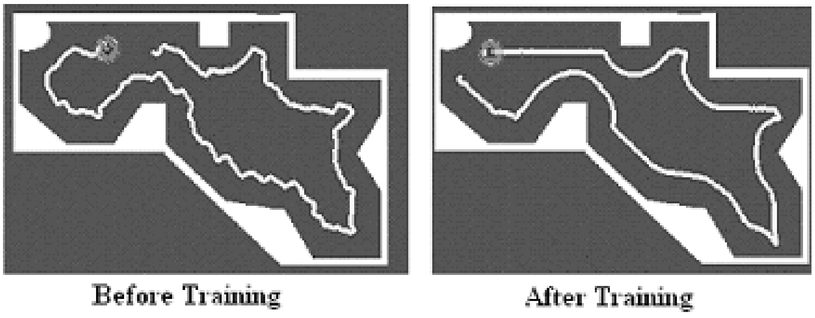


Fig. 15. Wall following performance before and after training

MFs are considered, there would be $3^4 = 81$ rules. In order to cover the input space well, we define MFs cover the region of the input space evenly with the value of ε-completeness set to 0.5. A basic fuzzy controller through trials and errors in (Er and Deng 2004) is adopted as the initial controller. The parameters for the FQL are set as follows: initial Q-value, $\mathrm{k_q} = 3.0$; exploration rate, $\mathrm{S_p} = 0.001$; discounted factor, $\gamma = 0.95$; trace-decay factor, $\lambda = 0.7$ and learning rate, $\alpha = 0.05$.

For the DFQL approach, the same settings as those in the original paper of (Er and Deng 2004) are adopted: the TD error factor for the DFQL is, $\mathrm{K}=50$, and threshold value for the TD error criterion is, $k_e = 1$. The ε-completeness, ε =0.5 and the similarity of MF, $\mathrm{k_{mf}} = 0.3$. Details of setting these parameters are discussed in (Er and Deng 2004). The CQGAF proposed in (Juang 2005) utilizes GA to obtain an optimal solution for the action set and employs the aligned clustering algorithm for structure identification of the FNN. As this chapter focuses on self-generation of the FNN structure, the same action set as that for the DFQL is applied for the CQGAF, which does not apply GA approach to train the action set. The aligned clustering algorithm in (Juang 2005) is adopted and the consequent part of FNN is trained through normal FQL which does not apply GA in the CQGAF. As the aligned clustering algorithm is similar to the ε-completeness criterion in the DFQL but described in another way, all parameters for the aligned clustering algorithm are set the same as those listed above for the DFQL. For the DSGFQL, the training aim is to limit the number of failures to 50. Therefore, the threshold values of the reward criterion should be set around $-3 \times 50/1000 = -0.15$ ($k_g = -0.15$, $k_{ll} = -0.10$ and $k_{hl} = -0.20$). If it requires each rule to be active at least with an average firing strength for 10 times in the 1000 training steps among about 50 rules, the firing strength threshold value is set as $10/(1000 \times 50) = 0.0002$. The performance of the different approaches is evaluated at every episode of 1000 control steps according to three criteria, namely the number of failures which corresponds to the total number of steps the robot has left the "lane" and reward which is accumulated and the number of fuzzy rules. The first two criteria measure the performance of the controller and the last criterion determines the

Table 2. Mean value of failures, rewards and fuzzy rules after 20 episodes for wall-following

	FQL (16 rules)	FQL (81 rules)	DFQL	CQGAF	DSGFQL
Failure	64	62	63	57	53
Rewards	-101	-100	-102	-84	-76
Rules	16	81	53	39	29

computational cost. The final mean values of the number of failures, rewards and rules after 20 episodes are shown in Table 2 for comparisons.

From Table 2, we can see that the DSGFQL method outperforms the FQL method. The DSGFQL approach is capable of online self-organizing learning. Besides the input-space partitioning, both overall and local performances have been evaluated to determine the structure of a fuzzy system. The common approach of conventional input-space partitioning is the so-called grid-type partitioning. The FQL with 16 rules partitions the state space coarsely. On the other hand, the speed of learning 81 rules is slow because a lot more parameters need to be explored. The proposed DSGFQL does not need to partition the input space a priori and is suitable for RL. It partitions the input space online dynamically according to both the input clustering and the performance. The compact fuzzy system considers sufficient rules in the critical state space which requires high resolution and does not include redundant rules in unimportant or unvisited state space so that learning is rapid and efficient.

It can also be concluded that the proposed DSGFQL algorithm is superior to the DFQL and CQGAF without the GA approach as the DSGFQL method achieves similar or even better performances than those approaches but with a much smaller number of rules. The superiority of the DSGFQL algorithm is due to the pruning capability as well as self-modification of the fuzzy MFs. The number of rules has been reduced as redundant rules have been eliminated. Moreover, the MFs can be dynamically adjusted according to the performance evaluation while the MFs can only be adjusted according to the input clustering when a new rule is created in the DFQL and CQGAF approaches.

5 Conclusions

In this chapter, the DSGFNN with a wide range of industry applications is proposed. In particular, the DFNN algorithm has been developed for SL problems and the DSGFQL has been proposed in RL domains. Both the DFNN and DSGFQL algorithms can create, adjust and delete fuzzy rules automatically and dynamically according to the performance. Those methods are suitable for a wide range of industry applications. In this chapter, the DFNN is applied for real-time control of a SCARA robot which is a SEIKO D-TRAN 3000 series robot and the DSGFQL approach is adopted for a wall-following task by a mobile robot which can be potentially applied as an auto-cleaner and an unmanned vehicle. Experiments and simulation studies demonstrate the superiority of the proposed architecture and algorithms over other existing methods and the suitability for industrial applications by the proposed approach.

References

Chak, C.K., Feng, G., Ma, J.: An Adaptive Fuzzy Neural network for MIMO System Model Approximation in High-Dimensional Space. IEEE Trans. Syst, Man, Cybern, Part B: Cybern. 28, 436–446 (1998)

Chen, L.P.: A Rapid Supervised Learning Neural Network for Function Interpolation and Approximation. IEEE Trans. Neural Networks 7, 1220–1229 (1996)

Chen, S., Cowan, C.F.N., Grant, P.M.: Orthogonal Least Squares Learning Algorithm for Radial Basis Function Network. IEEE Trans. Neural Networks 2, 302–309 (1991)

Dubois, D., Prade, H. (eds.): Fundamentals of fuzzy sets. The handbooks of fuzzy sets series. Kluwer Academic Publishers, Boston, London and Dordrecht (2000)

Er, M.J., Chin, S.H.: Hybrid adaptive fuzzy controllers of robot manipulators with bounds estimation. IEEE Trans. Ind. Electron. 47, 1151–1160 (2000)

Er, M.J., Deng, D.: Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning. IEEE Trans on Systems, Man and Cybernetics, Part B 34, 1478–1489 (2004)

Er, M.J., Gao, Y.: Robust Adaptive Control of Robot Manipulators Using Generalized Fuzzy Neural Networks. IEEE Trans. Industrial electronics 50, 620–628 (2003)

Er, M.J., Wu, S., Gao, Y.: Dynamic fuzzy neural networks: architectures, algorithms and applications. McGraw-Hill, NY, USA (2003)

Jang, J.S.R.: Anfis: adaptive-network-based fuzzy inference system. IEEE Trans. System, Man and Cybernetics 23, 665–684 (1993)

Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-fuzzy and soft computing. Prentice-Hall, NJ, USA (1997)

Juang, C.F.: Combination of online clustering and Q-Value based GA for reinforcement fuzzy systems. IEEE Trans. Fuzzy Systems 13, 289–302 (2005)

Jouffe, L.: Fuzzy inference system learning by reinforcement methods. IEEE Trans. Systems, Man, and Cybernetics, Part C 28, 338–355 (1998)

K-Team, S.A.: Khepera 2 User Manual. Lausanne, Switzerland (2002)

Lee, S.C., Lee, E.T.: Fuzzy sets and neural networks. Journal of Cybernetics 4, 83–103 (1974)

Lee, S.C., Lee, E.T.: Fuzzy neural networks. Mathematical Biosciences 23, 151–177 (1975)

Lin, C.T., Lee, C.S.G.: Neural-Network-Based Fuzzy Logic Control and Decision System. IEEE Trans. Computers 40, 1320–1336 (1991)

Lin, C.T., Lee, C.S.G.: Neural Fuzzy System: A Neural-Fuzzy Synergism to Intelligent Systems. Prentice-Hall, Englewood Cliffs, NJ (1996)

Lu, Y., Sundararajan, N., Saratchandran, P.: A Sequential Learning Scheme for Function Approximation by Using Minimal Radial Basis Function Networks. Neural Computing 9, 461–478 (1997)

Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. IEEE Trans. Neural Networks 11, 748–768 (2000)

Nilsson, T.: [online] (2003), available: http://www.kiks.f2s.com

Pal, S.K., Mitra, S.: Neuro-fuzzy pattern recognition: methods in soft computing. John Wiley & Sons, Inc., NY, USA (1999)

Passino, K.M., Yurkovich, S.: Fuzzy control. Addison Wesley Longman Inc., Amsterdam (1998)

Pedrycz, W.: Fuzzy control and fuzzy systems. John Wiley & Sons, Inc., NY, USA (1993)

Reed, R.: Pruning algorithm- a Survey. IEEE Trans. Neural Networks 4, 740–747 (1993)

Ritthiprava, P., et al.: A modified approach to Fuzzy-Q Learning for mobile robots. In: Proc. IEEE International Conference on Systems, Man and Cybernetics, pp. 2350–2356 (2004)

Sugeno, M.: Industrial applications of fuzzy control. Elsevier Science Inc., NY, USA (1985)

Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge, Massachusetts (1998)

Takagi, T., Sugeno, S.: Fuzzy Identification of Systems and its Applications to Modeling and Control. IEEE Trans. Syst. Man, Cybern. 15, 116–132 (1985)

Wang, L.X.: A course in fuzzy systems and control. Prentice-Hall, NJ, USA (1997)

Wu, S., Er, M.J.: Dynamic fuzzy neural networks: a novel approach to function approximation. IEEE Trans. Systems, Man and Cybernetics, Part B 30, 358–364 (2000)

Wu, S., Er, M.J., Gao, Y.: A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. IEEE Trans. Fuzzy Systems 5, 578–594 (2001)

Zanchettin, C., Ludermir, T.B.: A neuro-fuzzy model applied to odor recognition in an artificial nose. Design and application of hybrid intelligent systems, 917–926 (2003)

Zhang, N.: Experimental Research on Robot Control, M. Eng. Thesis, Nanyang Tech. Univ., Singapore (1996)

Kernel Estimators in Industrial Applications

Piotr Kulczycki

Systems Research Institute, Polish Academy of Sciences
ul. Newelska 6, PL-01-447 Warsaw, Poland
kulczycki@ibspan.waw.pl
Cracow University of Technology, Department of Automatic Control
ul. Warszawska 24, PL-31-155 Cracow, Poland
kulczycki@pk.edu.pl

1 Introduction

The specification, based on experimental data, of functions which characterize an object under investigation, constitutes one of the main tasks in modern science and technological problems. A typical example here is the estimation of density function of random variable distribution from any given sample. The classical procedures rely here on arbitrary assumption of the form of this function, and then in specification of its parameters. These are called parametric methods. A valuable advantage is their theoretical and calculational simplicity, as well as their being commonly known and present in subject literature. Nowadays – along with the dynamic development of computer systems – nonparametric methods, whose main feature constitutes a lack of arbitrary assumptions of the form of a density function, are used more and more often. In a probabilistic approach, kernel estimators are becoming the principal method in this subject. Although their concept is relatively simple and interpretation transparent, the applications are impossible without a high class of computer which, even until recently, significantly hindered theoretical, and especially practical research.

In this chapter, first – in Section 2 – the basics of kernel estimators methodology are presented in a form suitable for researchers without thorough knowledge in the area of advanced statistical methods. So, the fundamental definitions of a kernel estimator are described, as are one-dimensional, and also radial and product kernels for the multidimensional case, suboptimal – in a mean-square sense – methods for calculation of functions and parameters occurring there, as well as procedures of smoothing parameter modification and linear transformation. Thanks to today's availability and the possibilities of contemporary computer systems as well as the automation of metrological and data gathering processes, the universal character of kernel estimators allows for their broad application in various problems of modern science and technology, particularly those of an industrial nature. In Section 3 of this chapter, uses of kernel estimators are described for the following subjects:

- data analysis and exploration – recognition of atypical elements (outliers), clustering, and classification – applied to the detection and diagnosis of devices working in real-time, and planning of strategy for mobile phone operators;
- parametric identification illustrated in automatic control applications and by tasks of sharpening of imprecise information;

- definition of spatial distribution of demand based on fuzzy data for the needs of a problem from the telecommunications field.

In addition, the tasks of data compression and dimensionality reduction, based on artificial neural networks and evolutionary algorithms are also commented upon. These among others efficiently reduce calculation time in the above presented tasks.

The detailed description of the basics of kernel estimators methodology can be found in the monographs (Kulczycki 2005, Silverman 1986, Wand and Jones 1994). Many applicational aspects are also found in the papers (Kulczycki 2000-2002; Kulczycki and Charytanowicz 2005, Kulczycki and Daniel 2006, Kulczycki and Mazgaj 2005; Kulczycki and Waglowski 2005, Kulczycki and Wisniewski 2002). The preliminary version of this text was presented as (Kulczycki 2007).

2 Methodology of Kernel Estimators

Let the *n*-dimensional random variable *X*, with a distribution having the density *f*, be given. Its kernel estimator $\hat{f} : \mathbb{R}^n \to [0, \infty)$ is calculated on the basis of the *m*-elements random sample

$$x_1, x_2, \ldots, x_m \quad , \tag{1}$$

experimentally obtained from the variable *X*, and is defined in its basic form by the formula

$$\hat{f}(x) = \frac{1}{mh^n} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right) \quad , \tag{2}$$

where the measurable, symmetrical with respect to zero and having a weak global maximum in this point, function $K : \mathbb{R}^n \to [0, \infty)$ fulfils the condition $\int_{\mathbb{R}^n} K(x) \, \mathrm{d}x = 1$ and is called a kernel, whereas the positive coefficient *h* is referred to as a smoothing parameter.

The interpretation of the above definition is illustrated in Fig. 1, taking a one-dimensional random variable (i.e. when $n = 1$) as an example, for a 9-element sample (i.e. $m = 9$). In the case of the single realization x_i, the function *K* (transposed along the vector x_i and scaled by the coefficient *h*) represents the approximation of distribution of the random variable *X* having obtained the value x_i. For *m* independent realizations x_1, $x_2, \ldots, x_m$, this approximation takes the form of a sum of these single approximations. The constant $1/mh^n$ enables the condition $\int_{\mathbb{R}^n} \hat{f}(x) \, \mathrm{d}x = 1$, required of the density of a probability distribution.

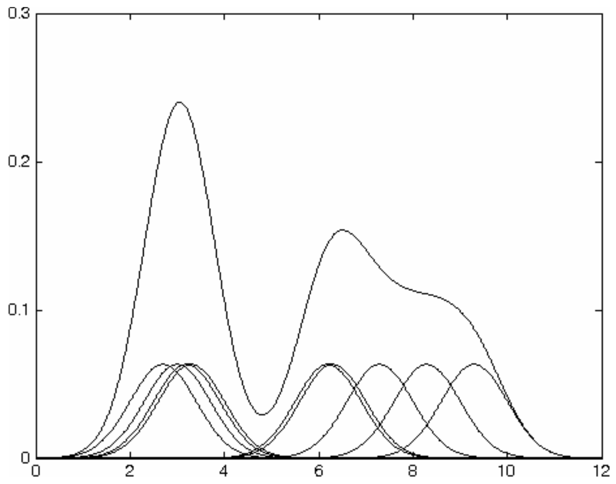


Fig. 1. Kernel density estimator (2) of one-dimensional random variable distribution

It is worth noting that a kernel estimator allows the identification of density for practically every distribution, without any assumptions concerning its membership to a fixed class. Atypical, complex distributions, also multimodal, are regarded here as textbook unimodal. In the multidimensional case, i.e. where $n > 1$, this enables, among others, the discovery of complete dependences between particular coordinates of the random variable under investigation.

Setting the quantities introduced in definition (2), i.e. choice of the form of the kernel *K* as well as calculation of the value for the smoothing parameter *h*, is most often carried out according to the criterion of minimum of an integrated mean-square error. These problems will be discussed in Sections 2.1 and 2.2.

2.1 Choosing the Kernel Form

From the statistical point of view, the form of the kernel appears to have no significance, and so it becomes possible for the choice of the function *K* to be arbitrary, taking into account above all the desired properties of an obtained estimator, for example its class of regularity, assuming (strictly) positive values or other features important in a considered problem, including calculational suitability. This is of particular importance in the case of complicated problems of modern industrial challenges, where the kernel estimator constitutes the most common basis for further comprehensive and complex investigations. Its obtained properties can then not only simplify further procedures, but often actually allow a concrete result, suitable for application, to be reached.

In the one-dimensional case, the function *K* assumes the classic forms of densities of probability distributions, normal, Cauchy, and triangular among others, as well as – in specific tasks – their linear combinations. In the sense of the integrated mean-square error criterion, the so-called Epanechnikov kernel

$$K(x) = \begin{cases} \frac{3}{4}(1-x^2) & \text{for} \quad x \in [-1,1] \\ 0 & \text{for} \quad x \in (-\infty,-1) \cup (1,\infty) \end{cases} \tag{3}$$

is the most effective. In the multidimensional case, two natural generalizations for the above concept are used: the radial kernel

$$K(x) = C\,\mathcal{K}(\sqrt{x^{\mathrm{T}} x}) \tag{4}$$

and the product kernel

$$K(x) = K([\,x_1,\, x_2, \ldots, x_n]^{\mathrm{T}}) = \mathcal{K}\,(x_1) \cdot \mathcal{K}\,(x_2) \cdot \ldots \cdot \mathcal{K}\,(x_n), \tag{5}$$

where $\mathcal{K}$ denotes the aforementioned one-dimensional kernel, while C is a positive constant, calculated so that $\int_{\mathbb{R}^n} K(x)\,\mathrm{d}x = 1$. For any assumed one-dimensional kernel $\mathcal{K}$, radial kernel (4) appears to be more effective than product kernel (5), although the difference – from an applicational point of view – seems to be negligible. Due to this fact, the product kernel is often preferred in practical problems. Apart from specific statistical tasks, it proves to be significantly more suitable in further analysis – for example procedures of integration and differentiation of a product kernel are not much different from the one-dimensional case. Among radial kernels the most effective is the radial Epanechnikov kernel, i.e. defined by dependence (4) when $\mathcal{K}$ is given by formula (3). Likewise in the family of product kernels, the most effective proves to be the product Epanechnikov kernel, described by equalities (5) and (3).

To summarize: the possibility of significant flexibility in choosing the form of the kernel K constitutes a great practical advantage which becomes more obvious as the specific problem under investigation is more complex.

2.2 Calculation of Smoothing Parameter Value

Unlike the kernel's form, the value of the smoothing parameter has a strong influence on the quality of the kernel estimator obtained. Thankfully, suitable algorithms have been developed, allowing the calculation of the value h based on random sample (1), which is close to optimal in a mean-square sense.

So, an approximation can be made – in the primary phase of the analysis – for the value obtained for the normal distribution

$$h = \left(\frac{V(K)}{U(K)^2} \pi^{n/2} \frac{2^{n+2}}{n+2} \frac{1}{m} \right)^{1/(n+4)} \bar{\sigma} \;, \tag{6}$$

where $\bar{\sigma}$ denotes the geometric mean of standard deviations for particular coordinates of the variable X, while $V(K) = \int_{\mathbb{R}^n} K(y)^2\,\mathrm{d}y$ and

$U(K) = \int_{\mathbb{R}^n} y^{\mathrm{T}} y \, K(y) \, \mathrm{d}y$. The cross-validation method is universal, whereby one calculates the value minimizing the real function $g : (0, \infty) \to \mathbb{R}$ defined as

$$g(h) = \frac{1}{m^2 h^n} \sum_{i=1}^{m} \sum_{j=1}^{m} \tilde{K}\left(\frac{x_j - x_i}{h} \right) + \frac{2}{mh^n} K(0) \quad , \tag{7}$$

when $\tilde{K}(x) = K^{*2}(x) - 2K(x)$, where K^{*2} denotes the convolution square of the function *K*. Applying iterative procedures to find the minimum, the initial value can be taken from formula (6).

For particular cases a number of appropriate algorithms have been worked out, such as the simple and effective plug-in method, used in the one-dimensional case.

2.3 Additional Procedures

The basic form of kernel estimator (2) can be generalized for overall improvement of its features as well as possibly extended to include additional aspects adapting the model to reality. As examples of such generalizations, the following procedures will be described:

- modification of the smoothing parameter (Section 2.4),
- linear transformation (Section 2.5),

while sample extensions will be presented in

- support boundary (Section 2.6),
- binary coordinates (Section 2.7).

2.4 Modification of Smoothing Parameter

The value for the smoothing parameter *h* introduced in definition (2) is the same for all kernels related to particular values of the random sample (1). Generally, a small value for this parameter causes a "thinning" of the kernel, whereas a large one, its "flattening". If therefore it is possible to individualize the influence of the smoothing parameter on specific kernels, then for areas "denser" with elements of a random sample, this value should be decreased (thereby better showing specific features of a distribution), as opposed to "sparser" areas, where it should be increased (which results in additional flattening of "tails"). With this aim modification of the smoothing parameter realizes the above according to the following algorithm.

Firstly one calculates the basic form of kernel estimator (2), then its values for particular elements of the random sample, i.e. $\hat{f}(x_1)$, $\hat{f}(x_2), \ldots, \hat{f}(x_m)$, as well as their geometrical mean $\bar{s}$. The modifying parameters $s_i > 0$ $(i = 1, 2, \ldots, m)$ are given by

$$s_i = \left(\frac{\hat{f}(x_i)}{\bar{s}} \right)^{-c} , \tag{8}$$

while $c \geq 0$, and finally, the kernel estimator with modified smoothing parameter is defined as

$$\hat{f}(x) = \frac{1}{mh^n} \sum_{i=1}^{m} \frac{1}{s_i^n} K\left(\frac{x - x_i}{hs_i} \right) . \tag{9}$$

This formula constitutes the generalization of definition (2). If $c = 0$, then $s_i \equiv 1$ and consequently these dependences are equivalent. The parameter *c* refers to intensity of the modification procedure. Corollaries arising from the integrated mean-square error criterion show the value $c = 0{,}5$.

Apart from an obvious improvement in quality of estimation, kernel estimators with a modified smoothing parameter maintain a number of additional advantages in practical applications. Above all they are more robust with respect to imprecise calculation of a smoothing parameter value. What is more, application of this procedure decreases the difference in effectiveness of particular kernel types with respect to optimal Epanechnikov kernel (3), as well as lowering the difference in effectiveness between product kernel (5) with respect to radial kernel (5) in the multidimensional case. These features are especially valuable in practice, since additionally, they increase the possibility of selecting more suitable kernels for further analysis in concrete applicational tasks.

2.5 Linear Transformation

In the multidimensional case, when the radial kernel is used, one can apply a simple procedure fitting a kernel shape to the form of distribution of a random variable under investigation, based on the idea of linear transformation. The definition of radial kernel (4) then generalizes to the equality

$$K(x) = \frac{C}{\sqrt{\det(R)}} \mathcal{K}(\sqrt{x^{\mathrm{T}} R^{-1} x}) , \tag{10}$$

where *R* is a positively defined matrix, while the meaning and value of the constant *C* remain unchanged. If *R* is a unique matrix, then the above formula is equivalent to dependence (4). Particularly useful results can be obtained by taking

$$R = \hat{\mathrm{Cov}} , \tag{11}$$

where $\hat{\mathrm{Cov}}$ denotes an estimator of covariance of the random variable *X*.

2.6 Support Boundary

In practical problems, particular coordinates of the random variable *X* may represent various quantities. Many of these, including those dealing with distance or time, if they are to be correctly and strictly interpreted, should belong to bounded subsets exclusively, for example the set of nonnegative numbers. In order to avoid calculational errors and misinterpretations arising from this, one can apply a simple procedure bounding a kernel estimator's support.

First, the case of a one-dimensional random variable and its left boundary – i.e. when the condition $\hat{f}(x) = 0$ for $x < x_*$, with $x_* \in \mathbb{R}$ fixed, is required – will be presented below. A part of any *i*-th kernel which finds itself beyond the interval $[x_*, \infty)$ becomes symmetrically "reflected" with respect to the boundary x_*, and is treated as a part of the kernel "grounded" in the symmetrical "reflection" of the element x_i with respect to the boundary x_*, so in the point $2x_* - x_i$. Therefore if one defines the function $K_{x_*} : \mathbb{R} \to [0, \infty)$ as

$$K_{x_*}(x) = \begin{cases} K(x) & \text{when} \quad x \geq x_* \\ 0 & \text{when} \quad x < x_* \end{cases} , \tag{12}$$

then the basic form of kernel estimator (2) takes the shape

$$\hat{f}(x) = \frac{1}{mh^n} \sum_{i=1}^{m} \left[K_{x_*}\left(\frac{x - x_i}{h}\right) + K_{x_*}\left(\frac{x + x_i - 2x_*}{h}\right) \right] . \tag{13}$$

Parts of particular kernels "cut" beyond an assumed support are thus "filled in" inside the support directly beside its boundary, therefore within the most commonly accepted – in practice – margin of error.

The case of a right boundary for a kernel estimator's support can be considered similarly. In the multidimensional instance, the above concept can be applied separately for each particular coordinate of the random variable under investigation.

2.7 Binary Coordinates

In many problems of contemporary engineering binary quantities appear, that is taking only two values, denoted symbolically hereinafter by 0 and 1. Besides quantities which are binary in nature, simplifications can often be applied to this form of description of even complicated phenomena, which however are of little influence to the final result. The methodology of kernel estimators enables binary coordinates to be taken into account.

Consider first the *k*-dimensional binary random variable $Y : \Omega \to \{0,1\}^k$. Its distribution is characterized by 2^k probabilities of the appearance of each possible *k*-dimensional vector of binary values. In many practical tasks, one can also infer with respect to values for the probability of the appearance of a given vector with the help of observations of vectors "similar" to it. The greater the "similarity", the likelier the inference becomes. Let therefore the function $p : \{0,1\}^k \to [0,1]$ be given, mapping to every *k*-dimensional vector of binary values the probability of its appearance. The kernel estimator of this function $\hat{p}:\{0,1\}^k \to [0,1]$ is calculated based on values of the *m*-elements random sample

$$y_1, y_2, \ldots, y_m \tag{14}$$

obtained from the variable *Y*, and is given as

$$\hat{p}(y) = \frac{1}{m}\sum_{i=1}^{m} L(y, y_i) \ , \tag{15}$$

while the function $L:\{0,1\}^k \times \{0,1\}^k \to [0,1]$ defines the dependence

$$L(y, y_i) = \delta^{k-d(y,y_i)}(1-\delta)^{d(y,y_i)} \ , \tag{16}$$

where $0{,}5 \le \delta \le 1$ and $0^0 = 1$ can be assumed, whereas the function $d:\{0,1\}^k \times \{0,1\}^k \to \mathbb{N}$ is given by $d(y, y_i) = (y - y_i)^{\mathrm{T}}(y - y_i)$. The value $k - d(y, y_i)$ shows the number of coordinates, to which the vectors *y* and y_i are equal, and represents the aforementioned "similarity" of binary vectors. The function *L* takes the role filled in definition (2) by the kernel *K*, and is called a binary kernel, while the parameter δ is termed a binary smoothing parameter. In practice its value can be calculated by minimizing the function $g:[0{,}5\,;1] \to \mathbb{R}$ given by formula

$$g(\delta) = -\sum_{i=1}^{m} \log[\hat{p}_{-i}(y_i)] \ , \tag{17}$$

where $\hat{p}_{-i}$ denotes estimator (15) obtained using the random sample $y_1, y_2, \ldots, y_{i-1}, y_{i+1}, \ldots, y_m$, therefore, omitting the *i*-th element of sample (14).

And finally, the (*n*+*k*)-dimensional random variable $Z \equiv [X, Y]^{\mathrm{T}}$ will be considered, being a composition of the *n*-dimensional continuous variable *X* investigated so far and the above defined *k*-dimensional binary variable *Y*. If the kernel estimator $\hat{f}:\mathbb{R}^n \times \{0,1\}^k \to [0,\infty)$ for the variable *Z* is calculated using the *m*-elements random sample

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}, \ldots, \begin{bmatrix} x_m \\ y_m \end{bmatrix} \ , \tag{18}$$

then the basic form of kernel estimator (2) becomes

$$\hat{f}(x, y) = \frac{1}{mh^n}\sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right) L(y, y_i) \ . \tag{19}$$

2.8 Sample Size

The last parameter requiring discussion is the crucial sample size, especially its dependence on the dimension of the random variable under investigation. The sample size m_* necessary to ensure a 10% accuracy at point zero for the n-dimensional standard normal distribution can be more or less taken as $m_* = 4^n$. Because of the exceptional regularity of the above distribution, and the significant looseness of the above criterion, such values seem to constitute an absolute minimum (suggested by e.g. $m_* = 4$ for $n = 1$). In the presence of binary coordinates these values should be multiplied by $(3/2)^k$. Next, the obtained result is multiplied by heuristically determined coefficients arising from the necessity to improve the quality of estimation, multimodality and asymmetry of the distribution, and also the correlation of random sample elements – in practice the product of these coefficients is most often equal to 3-10, in extreme cases even 100. For the one-dimensional random variable, the desired sample size in reality amounts to 20-50, increasing accordingly with the growth of a variable dimension. However, thanks to modern computer technology, even where multidimensional and complex tasks with adverse distribution features appear, this does not necessarily prove to be a significant applicational obstacle nowadays. One must always take into account the considerable advantages resulting from the application of kernel estimators, as they enable the identification of practically any distribution appearing in applicational problems, although they require a sample size adequate for immensity and universality of the information contained within.

2.9 Notes and Comments

In Section 2 a compendium of a kernel estimators methodology has been described. The basic form of definition of this type of estimator found in equality (2) was formulated, as were procedures for fixing quantities appearing therein. This form was generalized by the concept of smoothing parameter modification (9) and linear transformation (10), improving the properties of the estimator obtained. Additional procedures enabling the boundary of its support (13) and including binary coordinates (19) were presented, which allows the obtained model to be better fitted to the examined reality many times over. The above generalizations and supplementations were formulated with respect to the basic form of the kernel estimator, which makes it possible to combine them according to the demands of the concrete applicational problem. Because of clear interpretation, the introduction for additional generalizations is also possible. As an example – for the needs of further investigations in Section 3 – basic form (2) can be enhanced by nonnegative coefficients w_i with $i = 1,2,...,m$, and not all equal zero, which can be interpreted as the "meaning" of particular elements of random sample (1). Then

$$\hat{f}(x) = \frac{1}{h^n \sum_{i=1}^{m} w_i} \sum_{i=1}^{m} w_i K\left(\frac{x - x_i}{h}\right) . \qquad (20)$$

If $w_i \equiv 1$, the above formula is equivalent to definition (2).

Kernel estimators allow modeling of the distribution density – a basic functional characteristic of one- and multidimensional random variables – practically independent of its form and features. Consequently this is fundamental to obtain other functional characteristics and parameters. For example, if in a one-dimensional case one chooses the kernel *K*, such that its primitive $I(x) = \int_{-\infty}^{x} K(y)\, \mathrm{d}x$ may be analytically obtained, then the estimator of the distribution function

$$\hat{F}(x) = \frac{1}{m} \sum_{i=1}^{m} I\left(\frac{x - x_i}{h} \right) \tag{21}$$

can be defined. Next, if the kernel *K* has positive values, the solution for the equation

$$\hat{F}(x) = r \tag{22}$$

constitutes the kernel estimator of quantile of order $r \in (0,1)$.

Similar to the estimator of density of random variable distribution, the concept of the kernel estimator of spectral density can be formulated, as can be a particularly interesting notion – from an applicational point of view – the kernel estimator of regression function. According to the general nature of kernel estimators, this function is obtained without arbitrary assumptions fixing its shape, for example as linear or logarithmical.

3 Sample Industrial Applications

The universal character of kernel estimators allows their broad application in a variety of contemporary problems in science and practice. This process is possible thanks to today's modern computer systems, ubiquitous and ever more powerful, as well as the automation of procedures for both measuring and data gathering. Below are shown sample applications of kernel estimators in the following contemporary industrial tasks:

- data analysis and exploration – recognition of atypical elements (outliers), clustering, and classification – applied to the detection and diagnosis of devices working in real time, and planning of strategy for mobile phone operators (Section 3.1);
- parametric identification illustrated in automatic control applications and by tasks of sharpening of imprecise information (Section 3.2);
- definition of spatial distribution of demand based on fuzzy data for the needs of a problem from the telecommunications field (Section 3.3).

3.1 Data Analysis and Exploration

Here, the application of kernel estimators in basic tasks of data analysis and exploration will be considered, as will the recognition of atypical elements (outliers), clustering, and classification, and also the action of these procedures in the fault detection and diagnosis of devices working in real-time, and planning of strategy for mobile phone operators.

First, in many problems of data analysis the task of recognizing atypical elements (outliers) – those which differ greatly from the general population – arises. This enables the elimination of such elements from the available set of data, which increases its homogeneity (uniformity), and facilitates analysis, especially in complex and unusual cases. In practice, the recognition process for outliers is most often carried out using procedures of testing for statistical hypotheses (Barnett and Lewis 1994). The significance test based on the kernel estimators methodology will now be described.

Let therefore the random sample x_1, $x_2, \ldots, x_m$ treated as representative, therefore including a set of elements as typical as possible, be given. Furthermore, let $r \in (0,1)$ denote an assumed significance level. The hypothesis that $\tilde{x} \in \mathbb{R}^n$ is a typical element will be tested against the hypothesis that it is not, i.e. one should denote it as an outlier. The statistic $S : \mathbb{R}^n \to [0,\infty)$, used here, can be defined as

$$S(\tilde{x}) = \hat{f}(\tilde{x}) \quad , \tag{23}$$

where $\hat{f}$ denotes a kernel estimator of density, obtained for the random sample x_1, $x_2, \ldots, x_m$ mentioned above, while the critical set takes the left-sided form

$$A = (-\infty, a] \quad , \tag{24}$$

when a constitutes the kernel estimator of quantile of order r, calculated for the sample $\hat{f}(x_1)$, $\hat{f}(x_2), \ldots, \hat{f}(x_m)$, with the assumption that random variable support is bounded to nonnegative numbers.

Secondly, the aim of clustering is the division of a data set – for example given in the form of the random sample x_1, $x_2, \ldots, x_m$ – into subgroups (clusters), with every one including elements "similar" to each other, but with significant differences between particular subgroups (Hand et al. 2001, Larose 2005). In practice this often allows the decomposition of large data sets with differing characteristics of elements, into subsets containing elements of similar properties, which considerably facilitates further analysis, or even makes it possible at all. The following clustering procedure based on kernel estimators, taking advantage of the gradient methods concept (Fukunaga and Hostetler 1975) will be presented now.

Here the natural assumption is made, that clusters are prescribed to modes (local maximums) of the density kernel estimator $\hat{f}$, calculated for the considered random sample x_1, $x_2, \ldots, x_m$. Within this procedure, particular elements are moved in a direction defined by a gradient, according to the following iterative algorithm:

$$x_j^0 = x_j \quad \text{for} \quad j = 1, 2, \ldots, m \tag{25}$$

$$x_j^{k+1} = x_j^k + b \frac{\nabla \hat{f}(x_j^k)}{\hat{f}(x_j^k)} \quad \text{for} \quad j = 1, 2, \ldots, m \;\; \text{and} \;\; k = 0, 1, \ldots \quad , \tag{26}$$

where $b > 0$ and ∇ denotes a gradient. Thanks to the proper choice of form of the kernel *K*, a suitable analytical formula for the gradient ∇ becomes possible. In practice the value $b = h^2 / (n + 2)$ can be recommended.

As a result of the following iterative steps, the elements of the random sample move successively, focusing more and more clearly on a certain number of clusters. Their final shape can be defined after completing the k^* -th step. To this end one may calculate the kernel estimator for mutual distances of the elements $x_1^{k^*}$, $x_2^{k^*}, \ldots, x_m^{k^*}$ (under the assumption of nonnegative support of the random variable), after which the value is found where this estimator takes on the lowest local minimum, omitting a possible minimum in zero point. Particular clusters are assigned those elements whose distance is not greater than the above value. Thanks to the possibility of change in the smoothing parameter value, it becomes possible to affect the range of a number of obtained clusters, albeit without arbitrary assumptions concerning the strict value of this number, which allows it to be suited to a true data structure. Moreover, possible changes in intensity of the smoothing parameter modification – defined in formula (8) by the constant *c* – allows influence on the proportion of clusters located in dense areas of random sample elements to the number of clusters on the "tails" of the distribution under investigation.

Thirdly, the application of kernel estimators in a classification task is considered. Let the number $J \in \mathbb{N} \setminus \{0,1\}$ be given. Assume also, that the sample x_1, $x_2, \ldots, x_m$ obtained from the *n*-dimensional random variable has been divided into *J* separate subsets

$$x_1,\ x_2, \ldots, x_{m_1} \tag{27}$$

$$x_1,\ x_2, \ldots, x_{m_2} \tag{28}$$

$$\vdots$$

$$x_1,\ x_2, \ldots, x_{m_J} \quad , \tag{29}$$

while $m_1, m_2, \ldots, m_J \in \mathbb{N} \setminus \{0\}$ and $\sum_{j=1}^{J} m_j = m$, representing classes with features as mutually different as possible. The classification task requires deciding into which of them the given element $\tilde{x} \in \mathbb{R}^n$ should be reckoned (Hand et al. 2001, Larose 2005).

The kernel estimators methodology provides a natural mathematical tool for solving the above problem in the optimal Bayes approach. Let thus $\hat{f}_1$, $\hat{f}_2, \ldots, \hat{f}_J$ denote kernel estimators of density calculated for samples (27)-(29), respectively. If sizes m_1, $m_2, \ldots, m_J$ are proportional to the "frequency" of appearance of elements from particular classes, the considered element $\tilde{x}$ should be reckoned into the class for which the value $m_1 \hat{f}_1(\tilde{x})$, $m_2 \hat{f}_2(\tilde{x}), \ldots, m_J \hat{f}_J(\tilde{x})$ is the greatest.

The applicational possibilities of the above-presented procedures will now be illustrated in examples from the areas of fault detection and diagnosis for devices working in real-time, and a mobile phone operator's strategy.

Thus, the fault detection and diagnosis problem has lately become one of the most important challenges in modern control engineering. Early discovery of anomalies appearing in the operation of a controlled system, from an industrial robot to a nuclear reactor (i.e. detection), most often allows serious incidents and even catastrophes to be avoided, which could save material damage, or loss of human life. Secondly, confirmation of kind and location of these anomalies (i.e. diagnosis) is of fundamental meaning, especially when supervising large systems like complex chemical installations, as well as modern ships and airplanes. The importance of the above actions is multiplied by a psychological factor expressed by an increased feeling of safety, as well as – for the producer – prestige and commercial reputation. Finally, economic reasons often translate into a significant decrease in running costs, above all by ensuring the proper technological conditions as well as rationalizing overhauls and reducing repairs. Among the many different procedures used with this aim, the most universal are statistical methods. These very often consist in generating a certain group of variables that characterize the state of technical performance of the device, and then making a statistical inference, on the basis of their current values, as to whether or not the device is working properly, and in the event of a negative response, on the nature of the anomalies.

The procedures presented in this section, of recognition of atypical elements (outliers), clustering, and classification provides an appropriate tool for constructing an effective and suitable algorithm for use. If therefore a random sample represents conditions regarded as typical, signifying the correct operation of a device, and $\tilde{x}$ its current state, then, using the procedure of recognizing outliers it can be confirmed whether this state also should be considered as typical, or rather showing the appearance of anomalies (detection). In this case then, with random samples (27)-(29) characterizing particular kinds of typical faults, applying the classification procedure, one can tell which of them is being dealt with. If the partition of elements describing different kinds of anomalies is ambiguous, the appropriate division into classes can be obtained by clustering.

The next example of a direct application of data analysis and exploration procedures using kernel estimators is the planning of the strategy for mobile phone operators.

So, the continuously high rate of growth on the global mobile phone market forces the development of analytical methods, which serve to precisely specify the needs of an ever increasing group of users of this service. All mobile phone operators now have offers which differ less and less, both in services provided to clients and their obligatory tariffs. The market is consequently dividing with the aim of considering the various customer requirements. This forces effective realization of a company's strategy for satisfying these needs, while at the same time maximizing profits. However, continuation of such a process could lead to excessive segmentation through characterizing subscriber groups, which results in a loss of coherence in strategy with respect to clients. To avoid this one should find applications for new solutions of a systematic form. These allow a given group of mobile phone users to be divided in such a way that specifying a strategy with respect to the newly-formed subgroups does not require the whole procedure to be carried out again, yet uniquely defines the type of action for a given subgroup which maximizes an operator's profits, while on the other hand satisfying the client. In summary, among many factors which characterize every client one should select a set of such quantities which exclusively assign a subscriber to a specific, previously defined group of mobile service users, and specify a coherent strategy for each of them.

The above tasks, to which the methodology presented above will be applied, concern the business market of a particular mobile phone operator. The aim of the research is to define the appropriate strategy for a given client, taking into account such factors as – for example – mean monthly income from each SIM-card, length of subscription, and number of active SIM-cards. Based on the model constructed with the statistical kernel estimators methodology, the selected group of firms undergoes division into subgroups (clusters) characterized by the above factors. Both before and after clustering, those elements which are noted to be atypical (outliers) are excluded from the sample. The obtained division of clusters will result in the possibility of defining a concrete, preferred strategy with respect to each of them. Next, the algorithm enables the classification of any one firm involved in negotiations into the closest cluster with respect to its given characteristics, while also allowing the specification of an appropriate strategy. This will consist of the proper use of discounts which the operator can apply with regard to the client, all the while monitoring the costs incurred. Depending on which solution is applied concerning the subscriber, the risk of their leaving the given operator changes. On the other hand, it is important to find the threshold, where it is still profitable to continue providing a discount with the goal of keeping the client. In the conditions of fierce competition on the mobile phone market, the proper analysis and exploration of information contained in a client database not only allows for it to be effectively maintained, with the appropriate steering of the operator's development, but also becomes a source of information and solutions crucial to the acquisition of new clients.

It is worth mentioning the possibility of applying data compensation and dimensionality reduction procedures for the methodology described in this section. These among others efficiently lower calculation time in the above presented tasks.

In practice some elements of samples representing particular classes may be of little importance – from the point of view of propriety of statistical inference – or even contribute to mistakes. Their removal should therefore result in a reduction in the number of inappropriate decisions and also an increase in calculation speed. In order to realize this task one can apply the sensitivity method, patterned on the artificial neural networks theory.

For similar reasons the suggestion arises to reduce the dimensionality of the considered random variable X by the linear transformation

$$X^* = AX \tag{30}$$

mapping the space $\mathbb{R}^n$ onto the space $\mathbb{R}^{n^*}$ with significantly smaller dimension $n^* < n$. The elements of the matrix A should be selected such that the distances of particular random sample elements in both spaces are as mutually close as possible.

Since the number of these elements amounts to $n \cdot n^*$, which in practice makes using classical optimization methods impossible, then evolutionary algorithms can be proposed for the above task.

The above section contains results presented in the book (Kulczycki 1998), as well as material from research carried out together with Cyprian Prochot and Karina Daniel, published in the common papers (Kulczycki and Prochot 2004) and (Kulczycki and Daniel 2006), and also recently commenced works with Piotr Andrzej Kowalski and Szymon Lukasik.

3.2 Parameter Identification

One of the main problems of systems engineering is parameter identification – the specification of values of parameters existing in an investigated model. In a typical practical task, m independent measurements of the parameter, although suffering from errors of different origin, are available. On this basis one should define the value which, from an overall point of view of the problem to be worked out, would best represent phenomena described by this parameter. Usual estimation procedures, based on minimum integrated mean-square error or maximum likelihood methods, are applied mostly because of their popularity and availability in subject literature, however they do not allow differing causes of estimation errors to be taken into account.

This problem will be illustrated for the example of optimal control. The performance index, fundamental for the above task, may be used for testing not only the quality of a control, but also the procedure of identifying model parameters. As an example let the system, whose dynamic is described by the following differential equation

$$\dot{x}(t) = \begin{bmatrix} v & 1 \\ 0 & v \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ v \end{bmatrix} u(t) \quad , \tag{31}$$

while $v \in \mathbb{R} \setminus \{0\}$, be given. If the optimal feedback controller with quadratic performance index has been constructed with the value $V \in \mathbb{R} \setminus \{0\}$ not necessarily equal to the parameter v existing in object (31), then the obtained graph for this index can be approximated with reasonable precision by a quadratic function where coefficients differ for negative and positive errors (Fig. 2). Treating obtained values of an examined parameter as realizations of a random variable with the distribution density f, one can calculate the value of the optimal – in the sense of minimum expectation value of performance index for the control – estimator using Bayes' decision rule (Berger 1980). If the distribution of the above random variable is obtained with the aid of kernel estimators, the algorithm worked out is suitable for calculational procedures and in consequence for practical use.

First, a basic case will be investigated, considering the single parameter v. As a set of possible decisions $D = \mathbb{R}$ can be assumed, while the loss function is given as the asymmetrical quadratic form:

$$l(\hat{v}, v) = \begin{cases} p(\hat{v} - v)^2 & \text{for} \quad \hat{v} - v \le 0 \\ q(\hat{v} - v)^2 & \text{for} \quad \hat{v} - v \ge 0 \end{cases} \quad , \tag{32}$$

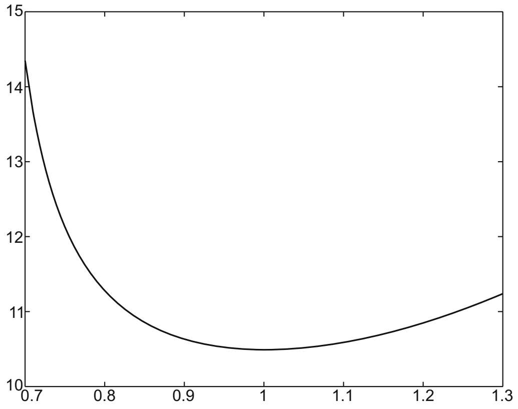


Fig. 2. Performance index value as a function of the parameter V ; ($v = 1$)

where $p, q > 0$ (note that these coefficients can be different), and $\hat{v}$ denotes the desired value of the Bayes decision which here fulfills the role of optimal estimator. This value is then given as the solution of the following equation with the argument $\hat{v}$:

$$(p-q)\int_{-\infty}^{\hat{v}}(\hat{v}-v)f(v)\,\mathrm{d}v = p\int_{-\infty}^{\infty}(\hat{v}-v)f(v)\,\mathrm{d}v \quad . \tag{33}$$

Solving the above criterion is generally no easy task. If, however, the kernel estimators methodology is used in specifying the density *f*, then, thanks to the proper choice of the kernel form, the effective numerical algorithm can be obtained. Let therefore *m* measurements for the examined parameter be given, treated as random sample (1). For the chosen kernel *K* one may define the following real mappings:

$$I(x) = \int_{-\infty}^{x} K(y)\,\mathrm{d}y \tag{34}$$

$$J(x) = \int_{-\infty}^{x} yK(y)\,\mathrm{d}y \quad , \tag{35}$$

and (for the basic form of kernel estimator (2)):

$$\bar{I}(x) = \frac{1}{mh}\sum_{i=1}^{m} I\left(\frac{x-x_i}{h}\right) \tag{36}$$

$$\bar{J}(x) = \frac{1}{mh}\sum_{i=1}^{m} J\left(\frac{x-x_i}{h}\right) \quad . \tag{37}$$

Then, criterion (32) takes the form of the equation

$$(p-q)[\hat{v}\bar{I}(\hat{v})-\bar{J}(\hat{v})]-p\hat{v}=-p\frac{1}{m}\sum_{i=1}^{m}x_i \quad . \tag{38}$$

If the kernel *K* assumes (strictly) positive values, the above solution exists and is unique. Denoting the left and right sides of the above equation as follows

$$L(\hat{v})=(p-q)[\hat{v}\bar{J}(\hat{v})-\bar{I}(\hat{v})]-p\hat{v} \tag{39}$$

$$P=-p\frac{1}{m}\sum_{i=1}^{m}x_i \quad , \tag{40}$$

and calculating the derivative of function (39):

$$L'(\hat{v})=(p-q)\bar{J}(\hat{v})-p \quad , \tag{41}$$

one can then – using Newton's iterative method – effectively obtain a solution for criterion (32) as a limit of the sequence $\{v^{(i)}\}_{i=1}^{\infty}$ given by the formulas

$$v^{(0)}=\frac{1}{m}\sum_{i=1}^{m}x_i \tag{42}$$

$$v^{(k+1)}=v^{(k)}+\frac{P-L(v^{(k)})}{L'(v^{(k)})} \qquad \text{for } k=0,1,\dots \quad . \tag{43}$$

The properties of kernel estimators allow generalizations of the above concept to be made for the multidimensional (parameters' vector), conditional (dependence on conditional factors), and polynomial (loss function to a power greater than two) cases.

Thus, in the multidimensional case, i.e. the task of a parameters' vector estimation, it is possible to carry out the above-proposed procedure with respect to a multidimensional random variable. As an example, if for the two-dimensional parameters' vector $v=[v_1,v_2]^{\mathrm{T}}$ one assumes the loss function in the following asymmetrical quadratic form:

$$l\left(\begin{bmatrix}\hat{v}_1\\ \hat{v}_2\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix}\right)=\begin{cases} c_l(\hat{v}_1-v_1)^2+c_{ld}(\hat{v}_1-v_1)(\hat{v}_2-v_2)+c_d(\hat{v}_2-v_2)^2 \\ \qquad \text{when } \hat{v}_1-v_1\le 0 \text{ i } \hat{v}_2-v_2\le 0 \\ c_p(\hat{v}_1-v_1)^2+c_{pd}(\hat{v}_1-v_1)(\hat{v}_2-v_2)+c_d(\hat{v}_2-v_2)^2 \\ \qquad \text{when } \hat{v}_1-v_1\ge 0 \text{ i } \hat{v}_2-v_2\le 0 \\ c_l(\hat{v}_1-v_1)^2+c_{lg}(\hat{v}_1-v_1)(\hat{v}_2-v_2)+c_g(\hat{v}_2-v_2)^2 \\ \qquad \text{when } \hat{v}_1-v_1\le 0 \text{ i } \hat{v}_2-v_2\ge 0 \\ c_p(\hat{v}_1-v_1)^2+c_{pg}(\hat{v}_1-v_1)(\hat{v}_2-v_2)+c_g(\hat{v}_2-v_2)^2 \\ \qquad \text{when } \hat{v}_1-v_1\ge 0 \text{ i } \hat{v}_2-v_2\ge 0 \end{cases} \quad , \tag{44}$$

where $c_l, c_p, c_g, c_d > 0$, $c_{ld}, c_{pg} \geq 0$ and $c_{pd}, c_{lg} \leq 0$, then here criterion (32) takes on the form of two equations, defining Bayes' decisions $\hat{v}_1$ and $\hat{v}_2$ of similar, albeit slightly more complex form.

In the case where the examined parameter depends heavily on a conditioning factor, the proposed procedures can be generalized with the purpose of including its influence. Let therefore the conditioning factor be given in the form of the random variable Z, taking the form of the composition of an *n*-dimensional continuous variable, whose distribution has a density, and a *k*-dimensional binary variable. Statistical kernel estimators can be used to estimate the density of total distribution. During application of the procedure, after fixing the concrete value of the conditional factor $Z(\omega)$, criterion (33) can be applied to a "cross-section" defined by this value. When significant conditioning for factors represented by variable Z occurs to an object under consideration, such an approach can considerably improve the quality of received results.

The form of the asymmetrical quadratic loss function (32) may be generalized to a polynomial case:

$$l(\hat{v}, v) = \begin{cases} (-1)^k \, p(\hat{v} - v)^k & \text{for} \quad \hat{v} - v \leq 0 \\ q(\hat{v} - v)^k & \text{for} \quad \hat{v} - v \geq 0 \end{cases}, \tag{45}$$

while $p, q > 0$ and $k = 2, 3, \ldots$. For instance when $k = 3$, a criterion analogical to equation (33) becomes

$$(p + q) \int_{-\infty}^{\hat{v}} (\hat{v} - v)^2 f(v) \, dv = q \int_{-\infty}^{\infty} (\hat{v} - v)^2 f(v) \, dv \quad . \tag{46}$$

The above text contains material from research carried out together with Aleksander Mazgaj, published in the common paper (Kulczycki and Mazgaj 2005).

Similar investigations concerning an asymmetrical linear loss function, i.e. in the case when formula (32) is replaced with the following dependence

$$l(\hat{v}, v) = \begin{cases} - p(\hat{v} - v) & \text{for} \quad \hat{v} - v \leq 0 \\ q(\hat{v} - v) & \text{for} \quad \hat{v} - v \geq 0 \end{cases}, \tag{47}$$

where $p, q > 0$ (while these coefficients can differ) were used to sharpen imprecise information in research carried out together with Malgorzata Charytanowicz, and published in the common paper (Kulczycki and Charytanowicz 2005).

3.3 Definition of Spatial Distribution of Demand

The Local Multipoint Distribution System (LMDS) is applied by telecommunications operators for wireless broadband data transmission purposes, which to a large degree results from a radical rise in the demand for Internet access. This system allows to connect the operator's network node to the buildings in which customers are located, without the necessity of constructing an expensive cable infrastructure. Thus, data is transmitted between base-stations, distributed across a metropolitan area, and providing regular connections with subscriber-stations located within the effective coverage of transceivers belonging to base-stations. Subscriber-stations installed on building roofs or facades then transmit data to customers through local (e.g. cable) networks. An essential factor which often decides about the economic justification of the LMDS system implementation is to determine base-station locations so that the highest profit can be achieved within the available investment funds. In this section an algorithm for designing an optimal system of LMDS base-stations will be presented.

The distribution of the spatial demand for data transmission services in the area under consideration has strict point structure regarding particular potential customers. Such a model has, however, limited applicational significance, since it is practically unidentifiable in a metropolitan area. Here, statistical kernel estimators will be applied for this purpose. The variable X existing in their definition is therefore two-dimensional, with its particular coordinates representing latitude and longitude. Thus, once in possession of a data base composed of m potential locations of subscriber buildings, with each of them characterized by its geographical position $x_i = [x_{i1}, x_{i2}]^{\mathrm{T}}$ and the coefficient w_i representing potential demand for data transmission services corresponding to this location (see formula (20)), one can calculate the kernel estimator describing the density spatial distribution of demand in a given area. Such identified distribution becomes continuous due to the properties of statistical kernel estimators. Moreover, thanks to the averaging aspects of such estimators, it is possible to use a simplified data base, including only the locations of main subscriber buildings, and also taking smaller objects in their neighborhood into account in the corresponding coefficients w_i. This considerably facilities the most difficult and expensive phase of the procedure of planning optimal locations of LMDS base-stations.

In practice, it is not difficult to point out a limited number of k sites for installing base-stations, including e.g. tall buildings and telecommunications towers. When one possesses the kernel estimator $\hat{f}$ characterizing the spatial distribution of demand in the area under investigation, for any such location x_j ($j = 1,2,...,k$) the performance index can be defined as

$$E_j = \int_{C_j} \hat{f}(x)\,\mathrm{d}x \quad , \tag{48}$$

where C_j denotes a circle with the center at x_j and the positive radius r_j representing maximum range of the transceiver mapped to this location. Thanks to the proper form of the kernel it is possible to describe this value with an analytical formula. It should be underlined that, from the point of view of the optimization problem investigated here, the positive constant $1 \big/ (h^2 \sum_{i=1}^{m} w_i)$ present in definition (20) may be omitted.

Based on the above dependence one can calculate the performance index for any system of base-stations with specific types of transceiver (or lack thereof, where installation of such equipment is not foreseen) mapped to particular locations. The proposed procedure also allows "shaded" areas (where transmission is not possible because of uneven terrain or obstacles, e.g. tall buildings) to be taken into account, as well as limited bitrate of equipment. Namely, in very attractive districts, demand for coverage by particular transceivers can not be satisfied by one base-station and so some of the unsatisfied demand should be met by another base-station within range – the optimal division is calculated based on the classic linear optimization methods, in particular the simplex algorithm.

With possession of the performance index for a fixed base-stations system created above, one can calculate – using operational research procedures, in particular the branch and bound method – their optimal system within the available investment budget. It is also possible to expand the problem to the task of planning over a few years, with changing conditions.

The coefficients w_i for $i = 1, 2, \ldots, m$ represent the demand for teletransmission services assigned to particular subscriber-station locations. Their identification is in practice conducted on the basis of an expert opinion expressed verbally, often based on intuitional premise. Consequently, the description of the predicted demand for teletransmission services by a subscriber-station will require fuzzy logic elements. The task with fuzzy character of the coefficients w_i introduced in definition (20) is commented below in detail, as such a problem can arise in various applications of kernel estimators in engineering challenges. What should be taken into account here is the specific nature of the task under consideration: a lot of fuzzy numbers (equal to the sample size *m*) necessary to identify and to use in subsequent analysis, as well as the fact that incidentally, the coefficients w_i may be deterministic owing to previously executed agreements.

In this situation, especially suitable are fuzzy numbers of the type *L-R*, whose membership function is denoted here in the following form:

$$\mu_{(w_i, \alpha_i, \beta_i)}(x) = \begin{cases} L\left(\dfrac{w_i - x}{\alpha_i}\right) & \text{for} \quad x \le w_i \\ R\left(\dfrac{x - w_i}{\beta_i}\right) & \text{for} \quad x \ge w_i \end{cases}, \tag{49}$$

where $w_i \in \mathbb{R}$ and $\alpha_i, \beta_i > 0$, while the real functions *L* and *R* are symmetrical with respect to zero, assume here the value 1 and are nondecreasing within the interval $(-\infty, 0]$. The parameter w_i may therefore be interpreted as a modal value, while α_i and β_i describe left- and right-hand concentration around that value, respectively. The fuzzy number $\mathcal{A}$ of the type *L-R* may, therefore, be identified by three parameters, which will be denoted as $\mathcal{A} = (w, \alpha, \beta)$, and, consequently, the process of identification requires only to determine the values which are close to intuitional interpretation. Algebraic operations on fuzzy numbers of the type *L-R* are defined as follows:

$$\mathcal{A} + \mathcal{B} = (w_{\mathcal{A}}, \alpha_{\mathcal{A}}, \beta_{\mathcal{A}}) + (w_{\mathcal{B}}, \alpha_{\mathcal{B}}, \beta_{\mathcal{B}}) = (w_{\mathcal{A}} + w_{\mathcal{B}}, \alpha_{\mathcal{A}} + \alpha_{\mathcal{B}}, \beta_{\mathcal{A}} + \beta_{\mathcal{B}}) \tag{50}$$

$$\mathcal{A} - \mathcal{B} = (w_{\mathcal{A}}, \alpha_{\mathcal{A}}, \beta_{\mathcal{A}}) - (w_{\mathcal{B}}, \alpha_{\mathcal{B}}, \beta_{\mathcal{B}}) = (w_{\mathcal{A}} - w_{\mathcal{B}}, \alpha_{\mathcal{A}} + \alpha_{\mathcal{B}}, \beta_{\mathcal{A}} + \beta_{\mathcal{B}}) \tag{51}$$

$$c \cdot \mathcal{A} = (cw_{\mathcal{A}}, c\alpha_{\mathcal{A}}, c\beta_{\mathcal{A}}) \ , \tag{52}$$

where $\mathcal{A}$ and $\mathcal{B}$ denote fuzzy numbers, while c is a positive real number. If one adopts the notation in which the real number a is described in the form of three parameters $a = (a, 0, 0)$, those operations may be generalized to addition and subtraction of the fuzzy and real numbers. Moreover, formulas (50)-(52) also correctly express the operations on two real numbers. Finally, the result is that the fuzzy number of the type *L-R* in the above range is a generalization of the real number. In the end, each coefficient w_i introduced in formula (20), was generalized to the three-parameter fuzzy number suitable for identification and calculation in practice, denoted below as $\mathcal{W}_i = (w_i, \alpha_i, \beta_i)$. In a special case, $\mathcal{W}_i = (w_i, 0, 0)$ may represent the real (nonfuzzy) number w_i.

If the coefficients w_i are fuzzy, then the performance index of the base-station system under consideration has a form of linear combination of three-parameter fuzzy numbers $\mathcal{W}_i$, and, therefore, due to formulas (50)-(52), it also becomes a three-parameter fuzzy number, denoted below as $\mathcal{E}$. To allow for the comparison of qualities of particular base-station systems, the methodology of fuzzy preference theory (Fodor and Rubens 1994) will be applied. The preference function $\mathcal{P}$ of the fuzzy number $\mathcal{E}$, with the bounded support of the membership function, will be adopted in the form resulting from the decision-making practice (Berger 1980):

$$\mathcal{P}(\mathcal{E}) = \delta \frac{\int_{\min \operatorname{supp} \mu_{\mathcal{E}}}^{\max \operatorname{supp} \mu_{\mathcal{E}}} x\mu_{\mathcal{E}}(x)\,\mathrm{d}x}{\int_{\min \operatorname{supp} \mu_{\mathcal{E}}}^{\max \operatorname{supp} \mu_{\mathcal{E}}} \mu_{\mathcal{E}}(x)\,\mathrm{d}x} + (1-\delta) \min \operatorname{supp} \mu_{\mathcal{E}} \ , \tag{53}$$

where $\delta \in [0,1]$, $\mu_{\mathcal{E}}$ means the membership function of the fuzzy number $\mathcal{E}$, while $\operatorname{supp} \mu_{\mathcal{E}}$ denotes its support. The value of the membership function is therefore a linear combination with weights δ and $1-\delta$ of the average value of the fuzzy number and the minimum value of its support. The average number corresponds to the Bayes decision rule and expresses a "realistic" operation, while the minimum value of the membership function support results from the minimax rule and represents the "pessimistic" point of view. The parameter δ determines therefore the company's strategy in the range from realistic (assuming average predicted demand) for $\delta = 1$, to pessimistic (assuming the lowest level of predicted demand) for $\delta = 0$. When clear preferences are missing, the value $\delta = 0{,}5$ can be proposed.

Finally, when two base-station systems characterized by fuzzy performance indexes are considered, the one for which the preference function (53) is larger should be treated as the "better".

The above section contains material from research carried out together with Jacek Waglowski, published in the common paper (Kulczycki and Waglowski 2005).

References

Barnett, V., Lewis, T.: Outliers in Statistical Data. Wiley, Chichester (1994)

Berger, J.O.: Statistical Decision Theory. Springer, New York (1980)

Fodor, J., Roubens, M.: Fuzzy Preference Modelling and Multicriteria Decision Support. Kluwer, Dordrecht (1994)

Fukunaga, K., Hostetler, L.D.: The estimation of the gradient of a density function, with applications in Pattern Recognition. IEEE Transactions on Information Theory 21, 32–40 (1975)

Hand, J.H., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge (2001)

Kulczycki, P.: Wykrywanie uszkodzen w systemach zautomatyzowanych metodami statystycznymi. Alfa, Warsaw (1998)

Kulczycki, P.: Fuzzy Controller for Mechanical Systems. IEEE Transactions on Fuzzy Systems 8, 645–652 (2000a)

Kulczycki, P.: A Statistical Fault Detection System. In: 16th IMACS World Congress on Scientific Computation. Applied Mathematics and Simulation, Lausanne, August 21-25, 2000, pp. CD: 216–214 (2000b)

Kulczycki, P.: An Algorithm for Bayes Parameter Identification. Journal of Dynamic Systems, Measurement, and Control 123, 611–614 (2001a)

Kulczycki, P.: A Fuzzy Approach for Mechanical System with Uncertain Load. In: European Control Conference, Porto, September 4-7, 2001, pp. 2658–2663, CD: ecc4561 (2001b)

Kulczycki, P.: Statistical Inference for Fault Detection: A Complete Algorithm Based on Kernel Estimators. Kybernetika 38, 141–168 (2002a)

Kulczycki, P.: A test for comparing distribution functions with strongly unbalanced samples. Statistica 62, 39–49 (2002b)

Kulczycki, P.: Bayes Parameter Identification with Reference to Nonlinear Optimal Control. In: 15th World Congress IFAC, Barcelona, July 21-26, 2002, CD: 1329 (2002c)

Kulczycki, P.: Estymatory jadrowe w analizie systemowej. WNT, Warsaw (2005)

Kulczycki, P.: Estymatory jadrowe w badaniach systemowych. In: Kulczycki, P., Hryniewicz, O., Kacprzyk, J. (eds.) Techniki informacyjne w badaniach systemowych, pp. 79–105. WNT, Warsaw (2007)

Kulczycki, P., Charytanowicz, M.: Bayes sharpening of imprecise information. Applied Mathematics and Computer Science 15, 393–404 (2005)

Kulczycki, P., Daniel, K.: Algorytm wspomagania strategii marketingowej operatora telefonii komorkowej. In: Kacprzyk, J., Budzinski, R. (eds.) Badania operacyjne i systemowe; metody i techniki, EXIT, Warsaw, pp. 245–256 (2006)

Kulczycki, P., Mazgaj, A.: An algorithm for Bayes parameter identification with quadratic asymmetrical loss function. Control and Cybenetics 34, 1127–1148 (2005)

Kulczycki, P., Prochot, C.: Wykrywanie elementow odosobnionych za pomoca metod estymacji nieparametrycznej. In: Kulikowski, R., Kacprzyk, J., Slowinski, R. (eds.) Badania operacyjne i systemowe; podejmowanie decyzji – podstawy teoretyczne i zastosowania, EXIT, Warsaw, pp. 313–328 (2004)

Kulczycki, P., Waglowski, J.: On the application of statistical kernel estimators for the demand-based design of a wireless data transmission system. Control and Cybernetics 34, 1149–1167 (2005)

Kulczycki, P., Wisniewski, R.: Fuzzy Controller for a System with Uncertain Load. Fuzzy Sets and Systems 131, 185–195 (2002)

Larose, D.T.: Discovering Knowledge in Data. In: An Introduction to Data Mining, Wiley, New York (2005)

Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)

Wand, M.P., Jones, M.C.: Kernel Smoothing. Chapman and Hall, London (1994)

Negative Selection Algorithm with Applications in Motor Fault Detection

X.Z. Gao, S.J. Ovaska, and X. Wang

Institute of Intelligent Power Electronics, Helsinki University of Technology, Otakaari 5 A, FI-02150 Espoo, Finland
Tel.: +358 9 451 2434, Fax: +358 9 451 2432
gao@cc.hut.fi, seppo.ovaska@tkk.fi, xiaolei@cc.hut.fi
http://powerelectronics.tkk.fi/

1 Introduction

Natural immune systems are complex and enormous self-defense systems with the remarkable capabilities of learning, memory, and adaptation (Goldsby et al. 2003). Artificial Immune Systems (AIS), inspired by the natural immune systems, are an emerging kind of soft computing methods (de Castro and Timmis 2002). With the distinguishing features of pattern recognition, data analysis, and machine learning, the AIS have recently gained considerable research interest from different communities (Dasgupta 2006; Dasgupta and Attoh-Okine 1997; Garrett 2005). Being an important constituent of the AIS, Negative Selection Algorithm (NSA) is based on the principles of maturation of T cells and self/nonself discrimination in the biological immune systems. It was developed by Forrest *et al.* in 1994 for the real-time detection of computer viruses (Forrest et al. 1994). During the past decade, the NSA has been widely applied in such promising engineering areas as anomaly detection (Stibor et al. 2005), networks security (Dasgupta and González 2002), aircraft fault diagnosis (Dasgupta 2004), and milling tool breakage detection (Dasgupta and Forrest 1995). In this paper, we first introduce the basic principle of the NSA. Two modified NSA, clonal selection algorithm-optimized and neural networks-based NSA, are next introduced. Their applications in the motor fault detection are also discussed.

2 Principles of Negative Selection Algorithm

It is well known that the immune system is an efficient self-defense system that can protect the human body from being affected by foreign antigens or pathogens (Goldsby et al. 2003). One of its most important functions is pattern recognition and classification. In other words, the immune system is capable of distinguishing the self, i.e., normal cells, from the nonself, such as bacteria, viruses, and cancer cells. This capability is achieved by two types of lymphocytes: B cells and T cells. More precisely, both the B cells and T cells are produced in the bone marrow. However, for the T cells, they must pass a *negative* selection procedure in the thymus thereafter. Only those that do not match the self proteins of the body will be released out from the bone marrow, while the remaining others are eventually destroyed there. The censoring of T cells actually can prevent the immune system from attacking the body's own proteins.

Forrest *et al.* proposed the NSA to mimic the aforementioned mechanism of biological immune system, as shown in Fig. 1. Their approach can be conceptually described as follows. Defining the self, we first collect a data set containing all the representative self samples. Next, the candidate detectors are randomly generated, and compared with the self set. Note, like the above selection of T cells, only those detectors that do not match any element of this set are retained. To put it into more details, let $[x_1, x_2, \cdots, x_N]$ and $[w_1, w_2, \cdots, w_N]$ denote a self sample and a candidate detector, respectively, where N is their common order. The matching degree d between $[x_1, x_2, \cdots, x_N]$ and $[w_1, w_2, \cdots, w_N]$ can be calculated based on the Euclidean distance:

$$d = \sqrt{\sum_{i=1}^{N} (x_i - w_i)^2} \, . \tag{1}$$

d is compared with a preset threshold λ, and detector matching error E is obtained:

$$E = d - \lambda \, . \tag{2}$$

If $E > 0$, we conclude that detector $[w_1, w_2, \cdots, w_N]$ fails to match self sample $[x_1, x_2, \cdots, x_N]$, and if $E \le 0$, $[w_1, w_2, \cdots, w_N]$ matches $[x_1, x_2, \cdots, x_N]$. After a certain number of qualified detectors are obtained by this negative selection procedure, they are used to detect the nonself or anomaly in the incoming samples. That is to say, when a new sample $[y_1, y_2, \cdots, y_N]$ matches $[w_1, w_2, \cdots, w_N]$, the existing anomaly

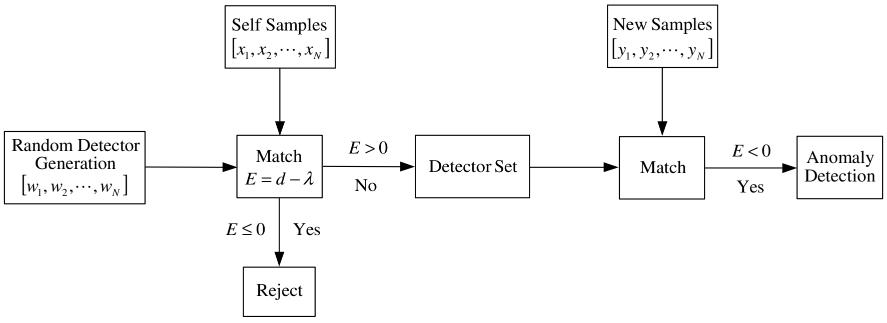

Self Samples $[x_1, x_2, \cdots, x_N]$
New Samples $[y_1, y_2, \cdots, y_N]$
Random Detector Generation $[w_1, w_2, \cdots, w_N]$
Match $E = d - \lambda$
$E > 0$
No
Detector Set
Match
$E < 0$
Yes
Anomaly Detection
$E \le 0$
Yes
Reject

Fig. 1. Negative selection algorithm

can be detected. In the next section, we are going to discuss how the NSA can be applied for detecting incipient motor faults.

3 Motor Fault Detection Using Negative Selection Algorithm

Fault detection/diagnosis methods are crucial in modern industry to assure the normal working conditions of plants, such as electrical machines and motors (Frank 1990). AC and DC motors are applied in various industrial applications. Changing working environments and dynamical loading always strain and wear motors, and may further cause some emerging faults, e.g., shorted turns, broken bearings, and damaged rotor bars. These faults can result in serious performance degradation and even eventual system failures, if they are not properly detected as well as handled. Therefore, motor drive monitoring and fault detection are very important but challenging topics in the electrical engineering field (Chow 1997).

Generally, the anomalies in the feature signals acquired from operating motors are considered to be caused only by incipient faults. Hence, fault detection is converted to a problem of anomaly detection, i.e., self/nonself discrimination, in the characteristic time series, which can be solved by utilizing the above NSA. The NSA-based motor fault detection scheme consists of three main stages. Firstly, the feature signals of healthy motors are sampled and preprocessed. Secondly, with the negative selection principle, a certain number of eligible detectors, S, are generated. Thirdly, the feature signals of the motors under inspection are sampled, preprocessed, and matched with these detectors. Fault detection results can be obtained based on the statistics of *activated* detectors.

Figure 2 shows the structure of the NSA in motor fault detection, in which there are two principal phases involved: detector generation phase and fault detection phase. In the detector generation phase, the feature signals of healthy motors are first collected, and they are next split into *non-overlapping* windows in the signal preprocessing unit, denoted by $[x_1, x_2, \cdots, x_N]$, as the input patterns of detectors. Applying the negative selection approach together with healthy feature signals, we can generate a pool of qualified detector candidates. In the fault detection phase, these detectors are employed to detect any anomaly in the feature signals from operating motors.

As aforementioned, the motor fault detection results are acquired based on the analysis of activated detectors. The fault detection rate is calculated using the feature signals of the healthy and faulty motors. More precisely, in case of n-fault detection, for fault i ($i = 1, 2, \cdots, n$), let L_i and M_i denote the numbers of sample patterns a detector can match ($E < 0$) in the healthy and faulty time series, respectively. The detection rate ρ_i is given:

$$\rho_i = \frac{M_i}{M_i + L_i} \times 100\% . \tag{3}$$

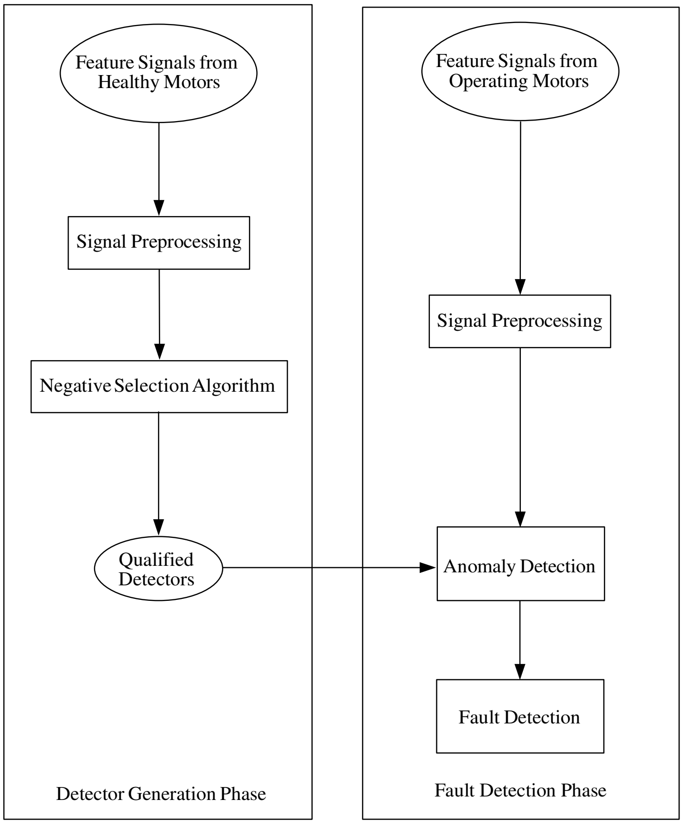

Feature Signals from Healthy Motors
Signal Preprocessing
Negative Selection Algorithm
Qualified Detectors
Detector Generation Phase
Feature Signals from Operating Motors
Signal Preprocessing
Anomaly Detection
Fault Detection
Fault Detection Phase

Fig. 2. Negative selection algorithm-based motor fault detection

Note that this NSA-based fault detection approach is not limited to only electric motors. It is actually independent of plants and fault types, and can, thus, be regarded as a general-purpose solution to a large variety of fault detection problems. In the next sections, two modified NSA, clonal selection principle-optimized NSA (Gao et al. 2006) and neural networks-based NSA (Gao et al. 2004), are proposed to improve the anomaly detection performance of regular NSA. We will also discuss their applications in motor fault detection.

4 Clonal Selection Principle-Optimized NSA in Motor Fault Detection

As we know, detectors of the original NSA are generated in a random manner. Unfortunately, conventional detector generation algorithms usually lead to an exhaustive and time-consuming procedure (Ayara et al. 2002; Ji and Dasgupta 2006). In addition, these detectors cannot be always guaranteed to achieve the optimal anomaly detection performance. To overcome this drawback, we employ the clonal optimization method to optimize the NSA detectors for the best anomaly detection rate, and further explore their applications in motor fault detection.

A. Clonal Optimization Method

Inspired by natural immune mechanisms, Artificial Immune Optimization (AIO) algorithms have been successfully applied to deal with numerous challenging optimization problems with superior performances over classical optimization techniques (Wang et al. 2004). Clonal optimization method is one of the most widely employed AIO approaches. It is based on the Clonal Selection Principle (CSP), which explains how an immune response is mounted, when a non-self antigenic pattern is recognized by the B cells (de Castro and von Zuben 2002). In the natural immune systems, only the antibodies that can recognize intruding antigens are selected to proliferate by cloning (Ada and Nossal 1987). Therefore, the fundamental of the clonal optimization method is that those cells (antibodies) capable of recognizing non-self cells (antigens) will proliferate. More precisely, the underlying principles borrowed from the CSP are:

- Maintenance of memory cells functionally disconnected from the repertoire;
- Selection and cloning of most stimulated antibodies;
- Suppression of non-simulated cells;
- Affinity maturation and re-selection of clones with higher affinity;
- Mutation rate proportional to cell affinity.

The diagram of the clonal optimization method is shown in Fig. 3, in which the corresponding iteration steps are explained as follows.

1. Initialize the antibody pool (P_{init}) including the subset of memory cells (M).
2. Evaluate the fitness of all the individuals in population P. The fitness here refers to the affinity measure.
3. Select the best candidates (P_r) from population P, according to their fitness (affinity with the antigen).
4. Clone these best antibodies into a temporary pool (C).
5. Generate a mutated antibody pool (C_1). The mutation rate of each individual is inversely proportional to its fitness.
6. Evaluate all the antibodies in C_1.
7. Eliminate those antibodies similar to the ones in C, and update C_1.
8. Re-select the individuals with better fitness from C_1 to build memory set M. Other improved individuals of C_1 can replace certain members with poor fitness in P to maintain the antibody diversity.

We should point out that the clone size in Step (4) is generally defined as either a monotonic function of the affinity measure or a constant value (de Castro and von Zuben 2002). A unique mutation operator is used in Step (5), through which the mutated values of individuals are inversely proportional to their fitness by means of choosing different mutation variations. In other words, the better fitness the antibody has, the less it may change. The similarity among candidates can also affect the

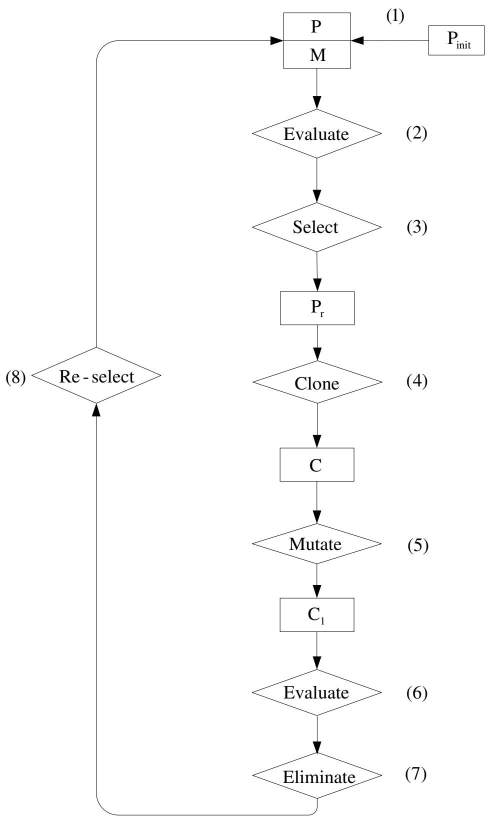


Fig. 3. Diagram of clonal optimization method

convergence speed of the clonal optimization method. The idea of antibody suppression inspired by the immune network theory is introduced to eliminate the newly generated antibodies, who are similar to those already in the candidate pool (Step 7). With such a diverse antibody pool, the clonal optimization method can avoid being trapped into local minima. In contrast to the popular Genetic Algorithms (GA), which usually tend to bias the whole population of chromosomes towards the best candidate solution (Poli and Langdon 2002), it can effectively handle the multi-model optimization tasks (Wang et al. 2005; Yoo and Hajela 1999). Thus, we utilize the clonal optimization method to optimize the NSA detectors.

B. Motor Fault Detection Using Clonal Selection Principle-Optimized NSA

Similarly with Fig. 2, Figure 4 shows the structure of our clonal optimization-based NSA in motor fault detection, where there are two main phases: detector optimization phase and fault detection phase. In the detector optimization phase, the feature signals

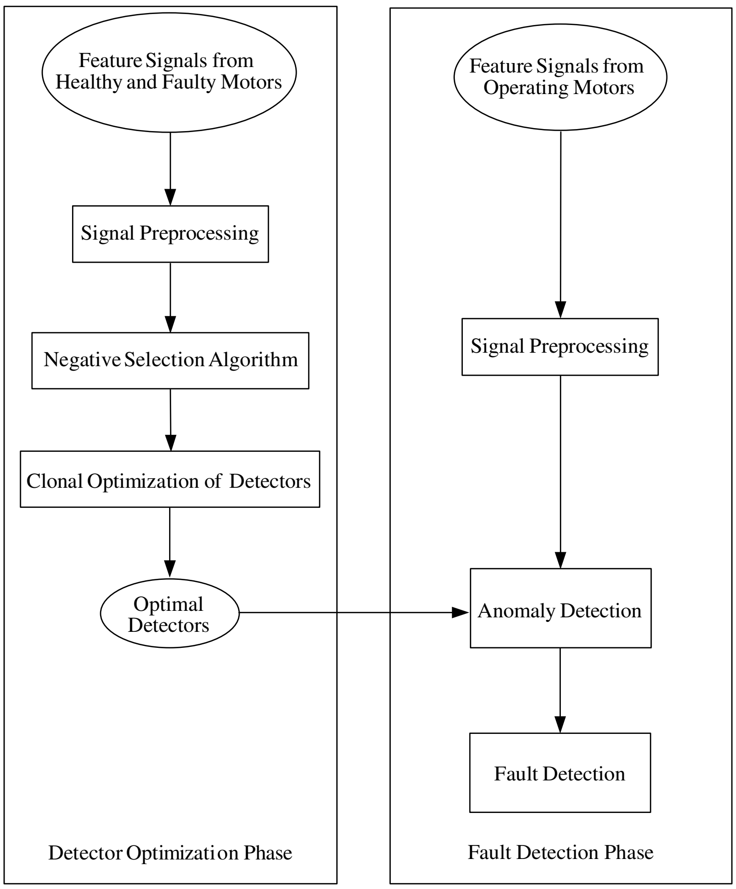

Feature Signals from Healthy and Faulty Motors
Signal Preprocessing
Negative Selection Algorithm
Clonal Optimization of Detectors
Optimal Detectors
Detector Optimization Phase
Feature Signals from Operating Motors
Signal Preprocessing
Anomaly Detection
Fault Detection
Fault Detection Phase

Fig. 4. Clonal optimization of NSA in motor fault detection

of both healthy and faulty motors are first collected. Applying the negative selection approach for the healthy feature signals, we can generate S qualified detector candidates. The clonal optimization method is further deployed to acquire only the optimal detectors with the best motor fault detection rates, where a detector is considered as an antibody, and the fault detection rate of each detector is its optimization fitness (affinity measure). In case of n-fault detection, for fault i ($i = 1,2,\cdots,n$), let L_i and M_i denote the numbers of sample patterns a detector can match ($E < 0$) in the healthy and faulty time series, respectively. The detection rate ρ_i is given:

$$\rho_i = \frac{M_i}{M_i + L_i} \times 100\% \,. \tag{4}$$

Therefore, the fitness of this detector is defined as $\sum_{i=1}^{n} \rho_i$. With the clonal optimization method, all the NSA detectors are optimized to maximize $\sum_{i=1}^{n} \rho_i$, and a group of optimal detectors can be obtained for motor fault detection. In the fault detection phase, these detectors are employed to detect any anomaly in the feature signals from operating motors.

The proposed motor fault detection scheme has several distinguishing features. Firstly, it provides us with the optimized detectors to achieve the best fault detection performance. Secondly, our approach is flexible in manipulating with multiple fault detection criteria by selecting proper fitness for the clonal optimization method. Lastly, utilizing the clonal optimization results in not only optimal but also *diverse* detectors, which give this fault detection system the unique robustness against isolated detector failures. Next, we are going to demonstrate the effectiveness of our clonal optimization-based NSA in motor bearings fault detection using computer simulations.

C. Simulations

Bearings are indispensable components in rotating machinery. Therefore, appropriate monitoring of their conditions is crucial to ensure the normal operation of motors (Chow 1997). Since the defects on the inner raceway, outer raceway, as well as balls are typical faults of bearings, we will examine only these three faults in our simulations. However, we must emphasize that the bearings fault detection problem is employed here only as a simplified testbed, and most of its practical details are not considered. The bearings feature signals are collected at the sampling frequency of 20 kHz from a vibration sensor mounted on top of the NYLA-K eight-ball bearings. The model of the vibration sensor used is IMI Sensors 601A01. The motor is a three-phase industrial motor of 0.5 horsepower manufactured by the Baldor Electric Company, which has the rotation speed at 1,782 rpm (Li et al. 2000). Figures 5 (a), (b), (c), and (d) show the vibration signal samples from the healthy bearings and those faulty bearings with the inner raceway, outer raceway, and ball faults, respectively.

Some simulation parameters in the generation of NSA detectors are first given as follows:

number of feature signal samples used: 5000,

number of detectors: $S = 10$,

window width: $N = 10$,

matching threshold: $\lambda = 0.25$.

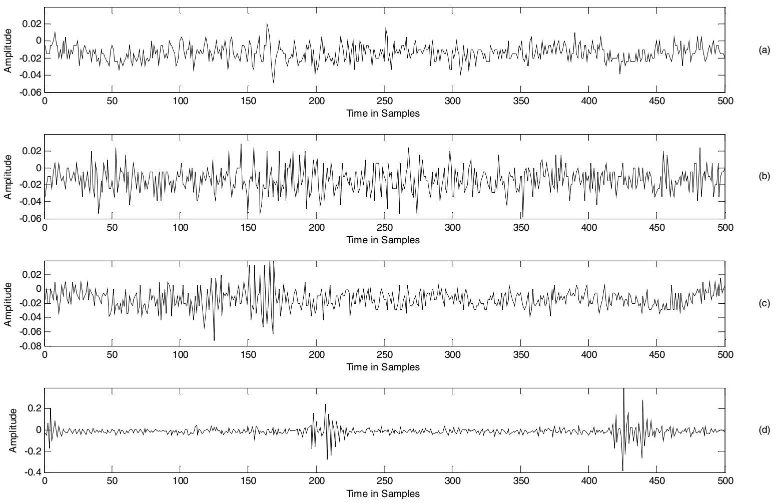


Fig. 5. Features signals of healthy and faulty bearings (a) Healthy bearings (b) Faulty bearings with inner raceway fault (c) Faulty bearings with outer raceway fault (d) Faulty bearings with ball fault

Note, *fresh* feature signals are next deployed to examine the fault detection rates of different bearings faults of the generated NSA detectors. There are totally 10,000 new samples in each of these three verification time series. The beginning 5,000 samples are taken from healthy bearings, and the following 5,000 samples are from three faulty bearings with the corresponding inner race, outer raceway, and ball faults, as demonstrated in Figs. 6 (a), (b), and (c), respectively. Ten detectors are generated based on the regular NSA:

$$\begin{bmatrix}
-0.0789 & 0.0766 & -0.0919 & 0.0984 & 0.0762 & 0.0692 & 0.0784 & -0.0852 & 0.0963 & 0.0953 \\
0.0945 & 0.0887 & 0.0965 & -0.0989 & -0.0813 & 0.0679 & 0.0817 & -0.0264 & -0.0902 & 0.0961 \\
-0.0654 & 0.0976 & -0.0746 & 0.0816 & 0.0998 & 0.0960 & 0.0342 & -0.0901 & 0.0906 & 0.0732 \\
0.0871 & 0.0878 & -0.0960 & 0.0653 & 0.0919 & -0.0774 & 0.0804 & 0.0896 & 0.0940 & 0.0585 \\
0.0795 & 0.0987 & 0.0911 & 0.0887 & 0.0667 & 0.0624 & 0.0998 & -0.0282 & 0.0780 & 0.0649 \\
0.0997 & 0.0483 & 0.0918 & 0.0707 & 0.0980 & -0.0939 & 0.0572 & 0.0679 & 0.0612 & 0.0984 \\
0.0956 & 0.0976 & -0.0589 & 0.0516 & 0.0718 & 0.0959 & 0.0920 & 0.0619 & -0.0522 & 0.0943 \\
-0.0873 & 0.0715 & -0.0758 & -0.0948 & 0.0925 & -0.0926 & 0.0993 & 0.0653 & 0.0719 & 0.0968 \\
-0.0250 & 0.0920 & 0.0898 & 0.0992 & -0.0165 & 0.0619 & 0.0788 & -0.0991 & 0.0843 & 0.0985 \\
0.0506 & 0.0585 & 0.0249 & 0.0996 & 0.0999 & 0.0959 & 0.0990 & -0.0097 & 0.0900 & 0.0973
\end{bmatrix}.$$

With the fresh verification feature signals in Fig. 6, the detection performances of these detectors for the inner raceway, outer raceway, and ball faults are verified and illustrated in Figs. 7 (a), (b), and (c), respectively. It should be pointed out that the 'Time in Samples' in the figures here actually refers to 'Time in Every Ten Samples', since after the data preprocessing, all the time series have been split into non-overlapping windows with a width of ten. The detection rates of the original NSA detectors for the three different bearings faults are summarized in Table 1.

Table 1. Fault detection rates of original NSA detectors

	M	L	ρ
Inner Raceway Fault	18	6	75.0%
Outer Raceway Fault	34	6	85.0%
Ball Fault	62	6	91.2%

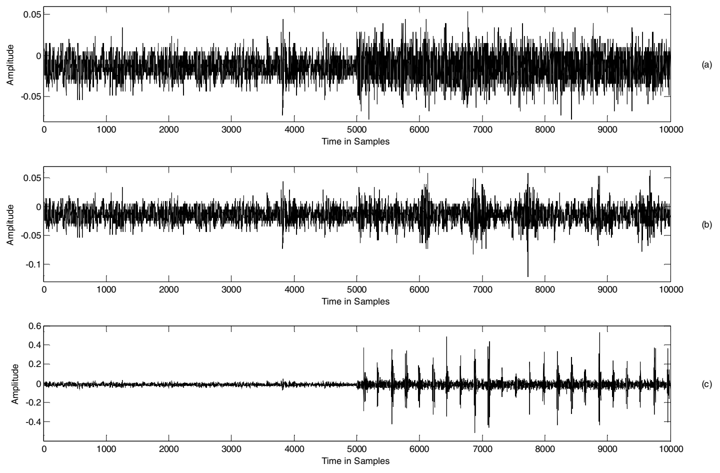


Fig. 6. Fresh verification feature signals for fault detection of NSA detectors (a) Inner raceway fault detection (b) Outer raceway fault detection (c) Ball fault detection

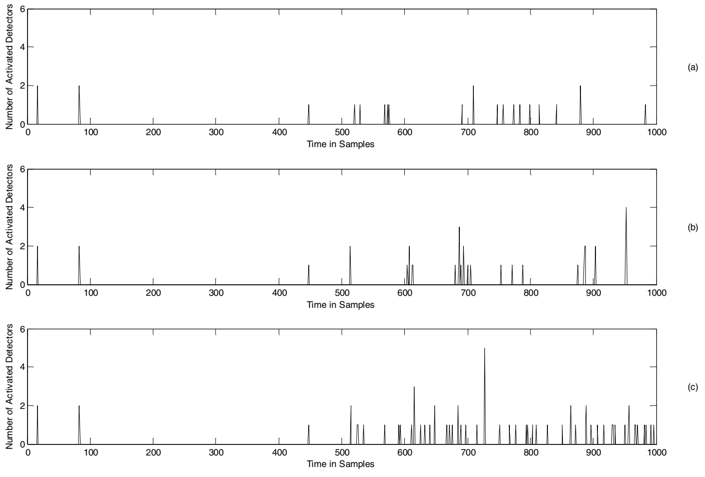


Fig. 7. Bearings fault detection with original NSA detectors (a) Inner raceway fault detection (b) Outer raceway fault detection (c) Ball fault detection

We first investigate the single fault, inner raceway fault, detection case. That is, the original NSA detectors are optimized using the aforementioned clonal optimization method to achieve the best detection rate for only this fault. The resulting optimal detectors are given as follows:

$$\begin{bmatrix} -0.0651 & 0.0979 & -0.0747 & 0.0807 & 0.0981 & 0.0955 & 0.0341 & -0.0909 & 0.0897 & 0.0723 \\ 0.0959 & 0.0883 & -0.1005 & 0.0936 & 0.0730 & -0.0811 & 0.1009 & -0.0806 & 0.0368 & -0.0737 \\ -0.0702 & 0.0951 & -0.0941 & 0.0914 & -0.0651 & 0.0895 & 0.0774 & -0.0961 & 0.0867 & 0.0557 \\ 0.0966 & -0.0977 & 0.0809 & 0.0980 & -0.0374 & -0.0288 & 0.0971 & 0.0806 & -0.0837 & 0.0963 \\ 0.0801 & -0.0638 & -0.0921 & 0.0905 & -0.0860 & 0.0874 & -0.0793 & 0.0970 & 0.0583 & 0.0861 \\ 0.0813 & 0.0974 & 0.0978 & -0.0906 & 0.0973 & 0.0040 & -0.0996 & -0.0634 & 0.0898 & 0.0685 \\ 0.0902 & -0.0896 & 0.0649 & 0.0856 & -0.0939 & 0.0965 & -0.0990 & 0.0173 & -0.0985 & 0.0857 \\ -0.0914 & 0.1002 & -0.0438 & -0.0214 & -0.0838 & 0.0915 & 0.0909 & -0.0895 & 0.0951 & 0.0989 \\ 0.0933 & 0.0751 & -0.0948 & 0.0953 & 0.0721 & -0.0895 & 0.0931 & -0.0413 & -0.0900 & 0.0836 \\ 0.0545 & 0.0936 & -0.0787 & 0.0783 & 0.0598 & 0.0839 & 0.0938 & -0.0934 & 0.0785 & -0.0912 \end{bmatrix}.$$

We emphasize that these clonal optimized detectors are considerably distributed, due to the solution diversity characteristics of the clonal optimization method. Their detection performance is illustrated in Fig. 8, and the detection rate is given in Table 2. Obviously, ρ of the original detectors is only 75.0% ($M = 18$ and $L = 6$), and it is increased to 90.3% ($M = 102$ and $L = 11$) after they have been optimized. This indeed demonstrates that our clonal optimization-based NSA can significantly improve the detection rate of single fault.

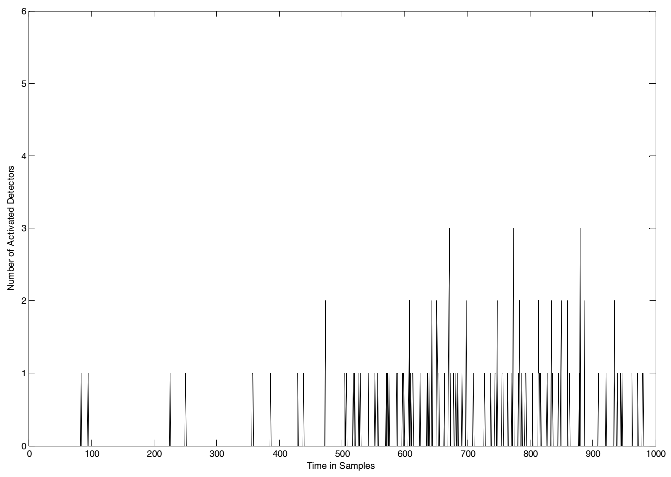


Fig. 8. Single fault detection with clonal optimized NSA detectors

Table 2. Detection rate of single fault with clonal optimized NSA detectors

	M	L	ρ
Inner Raceway Fault	102	11	90.3%

Next, the same original detectors are optimized to deal with a more complex problem: dual fault, both inner raceway and outer raceway faults, detection. In other words, the fitness for the clonal optimization procedure is the sum of the detection rates of these two faults. The optimal detectors acquired this time are:

$$\begin{bmatrix} -0.0654 & 0.0973 & -0.0746 & 0.0816 & 0.0998 & 0.0961 & 0.0343 & -0.0902 & 0.0905 & 0.0729 \\ 0.0945 & 0.0884 & -0.0984 & 0.0923 & 0.0744 & -0.0815 & 0.0999 & -0.0821 & 0.0365 & -0.0721 \\ -0.0702 & 0.0951 & -0.0942 & 0.0921 & -0.0636 & 0.0892 & 0.0768 & -0.0956 & 0.0872 & 0.0567 \\ 0.0967 & -0.0967 & 0.0815 & 0.0968 & -0.0359 & -0.0295 & 0.0986 & 0.0820 & -0.0846 & 0.0954 \\ 0.0795 & -0.0654 & -0.0930 & 0.0933 & -0.0855 & 0.0864 & -0.0803 & 0.0991 & 0.0584 & 0.0882 \\ 0.0795 & 0.0994 & 0.0992 & -0.0896 & 0.0981 & 0.0037 & -0.0984 & -0.0632 & 0.0895 & 0.0690 \\ 0.0922 & -0.0861 & 0.0626 & 0.0845 & -0.0979 & 0.0969 & -0.0969 & 0.0151 & -0.0988 & 0.0845 \\ -0.0920 & 0.0998 & -0.0429 & -0.0214 & -0.0844 & 0.0901 & 0.0924 & -0.0883 & 0.0946 & 0.0996 \\ 0.0914 & 0.0766 & -0.0954 & 0.0967 & 0.0756 & -0.0906 & 0.0942 & -0.0422 & -0.0902 & 0.0842 \\ 0.0590 & 0.0968 & -0.0761 & 0.0812 & 0.0619 & 0.0865 & 0.0966 & -0.0949 & 0.0819 & -0.0952 \end{bmatrix}.$$

The dual fault detection results using these detectors are given in Fig. 9 and Table 3. As we can see, the detection rates of the inner raceway and outer raceway faults are increased from 75.0% and 85.0% to 89.7% and 89.1%, respectively. It is clearly visible that our clonal optimization-based NSA is capable of even handling the dual fault detection with a satisfactory performance.

Table 3. Detection rates of dual fault with clonal optimized NSA detectors

	M	L	ρ
Inner Raceway Fault	87	10	89.7%
Outer Raceway Fault	82	10	89.1%

Finally, the triple fault detection problem including the detection of all the three bearings faults, i.e., inner raceway, outer raceway, and ball faults is examined. Those clonal optimized detectors are as follows:

$$\begin{bmatrix} -0.0652 & 0.0973 & -0.0747 & 0.0815 & 0.0998 & 0.0959 & 0.0342 & -0.0901 & 0.0903 & 0.0731 \\ 0.0945 & 0.0885 & -0.0984 & 0.0922 & 0.0743 & -0.0815 & 0.0998 & -0.0821 & 0.0366 & -0.0723 \\ -0.0696 & 0.0954 & -0.0937 & 0.0920 & -0.0646 & 0.0891 & 0.0771 & -0.0961 & 0.0866 & 0.0567 \\ 0.0975 & -0.0974 & 0.0823 & 0.0979 & -0.0367 & -0.0280 & 0.0990 & 0.0831 & -0.0849 & 0.0961 \\ 0.0795 & -0.0655 & -0.0929 & 0.0932 & -0.0855 & 0.0865 & -0.0803 & 0.0991 & 0.0584 & 0.0882 \\ 0.0966 & -0.0693 & 0.0964 & -0.0955 & 0.0803 & 0.0311 & 0.0816 & 0.0599 & 0.0914 & 0.0981 \\ 0.0928 & -0.0932 & 0.0862 & -0.0676 & -0.0926 & 0.0971 & 0.0973 & -0.0341 & 0.0999 & 0.0812 \\ -0.0917 & 0.0990 & -0.0418 & -0.0215 & -0.0841 & 0.0892 & 0.0915 & -0.0881 & 0.0941 & 0.0984 \\ 0.0924 & 0.0765 & -0.0966 & 0.0972 & 0.0757 & -0.0908 & 0.0955 & -0.0420 & -0.0904 & 0.0844 \\ 0.0591 & 0.0970 & -0.0762 & 0.0813 & 0.0618 & 0.0865 & 0.0965 & -0.0951 & 0.0816 & -0.0953 \end{bmatrix}.$$

Figure 10 shows this triple fault detection performance, and Table 4 gives the detection rates of the three faults. The new detection rates of the inner raceway, outer raceway, as well as ball faults are 86.3%, 87.2%, and 94.4%, respectively. Compared with Table 1, all these three detection criteria have been improved, among which the improvement of the inner raceway fault detection rate is the largest. The detection rates of the other two faults: outer raceway and ball faults are only marginally increased, because their

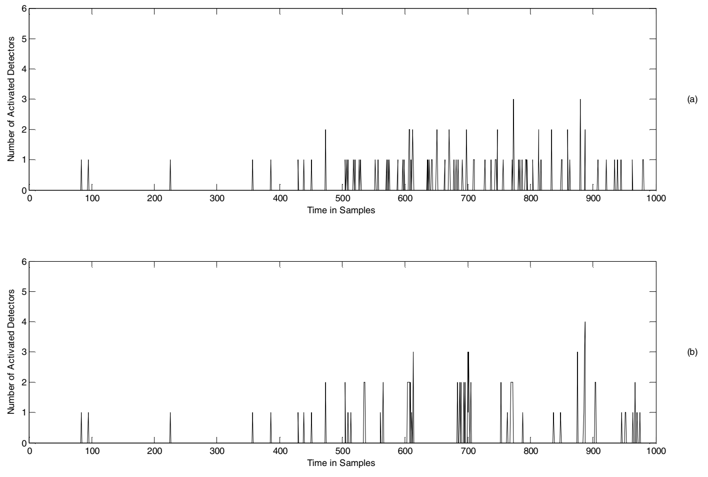


Fig. 9. Dual fault detection with clonal optimized NSA detectors (a) Inner raceway fault detection (b) Outer raceway fault detection

Table 4. Detection rates of triple fault with clonal optimized NSA detectors

	M	L	ρ
Inner Raceway Fault	69	11	86.3%
Outer Raceway Fault	75	11	87.2%
Ball Fault	187	11	94.4%

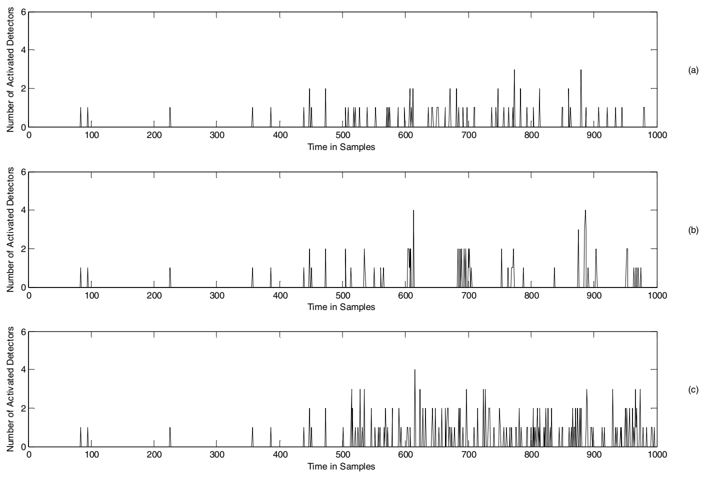


Fig. 10. Triple fault detection with clonal optimized NSA detectors (a) Inner raceway fault detection (b) Outer raceway fault detection (c) Ball fault detection

original values are high enough. Therefore, we conclude that our clonal optimization-based NSA is indeed effective and flexible in coping with multiple fault detection problems, in which an appropriate optimization trade-off among the detection rates of individual faults can be achieved.

D. Remarks

We propose a clonal optimization-based NSA, in which the detectors are optimized by the clonal selection principle to achieve the best anomaly detection performance. The applications of our new NSA in motor fault detection are also discussed. A bearings fault detection problem is employed here to verify the proposed scheme. Enhanced performances of single, dual, and triple bearings fault detection are obtained in computer simulations.

5 Neural Networks-Based NSA in Motor Fault Detection

It is obvious that effective generation of detectors is pivotal in the NSA, which depends on a few important factors, e.g., the size of the self set, matching rule between the detectors and self, and detector generation strategies (Ji and Dasgupta 2006). Particularly, the form of the detectors plays a crucial role here. González compared the performances of three typical detector representations including hyper-rectangles, fuzzy rules, and hyper-spheres in (González 2003). Unfortunately, conventional NSA detectors represented by either binary strings or real values usually are not adaptive (González et al. 2003; Stibor 2005). With no capability of adaptation, these detectors are not well suited for dealing with real-world problems under the time-varying circumstances. To handle this shortcoming as well as improve the performance of conventional NSA detectors, an adaptive neural networks-based NSA is proposed, and the corresponding training algorithm is derived in this section. A general fault diagnosis framework using the new NSA is also constructed. Simulations of two numerical examples, anomaly detection in Mackey-Glass time series and bearings fault detection, are made to verify our scheme.

A. Neural Networks-Based Negative Selection Algorithm

In the new neural networks-based NSA, detectors are built on the structures of three-layer feedforward neural networks, as shown in Fig. 11. $[x_1, x_2, \cdots, x_N]$ is the input vector, N is the number of inputs, and $[w_1, w_2, \cdots, w_N]$ and $[v_1, v_2, \cdots, v_N]$ are the connection weights between Layer 1 & Layer 2 and Layer 2 & Layer 3, respectively. More precisely, in Layer 1, the matching degree d_i between x_i of $[x_1, x_2, \cdots, x_N]$ and w_i of $[w_1, w_2, \cdots, w_N]$ is calculated in every input node i :

$$d_i = (x_i - w_i)^2, \tag{5}$$

where $i = 1, 2, \cdots, N$. The hidden node outputs are weighted in Layer 2 by $[v_1, v_2, \cdots, v_N]$. Thus, the final single output of Layer 3, y, can be given:

$$y = \sum_{i=1}^{N} v_i \,\mathrm{f}(d_i) = \sum_{i=1}^{N} v_i \,\mathrm{f}\left[(x_i - w_i)^2\right]. \tag{6}$$

$\mathrm{f}(\cdot)$ is the node function of Layer 2. Similarly with the original NSA, y is compared with a preset threshold λ, and the detector matching error E is obtained:

$$E = y - \lambda \tag{7}$$

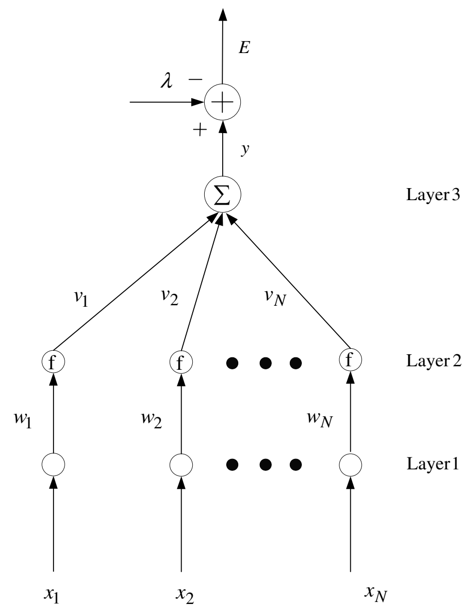


Fig. 11. Neural networks-based NSA detectors

If for any $[x_1, x_2, \cdots, x_N]$ in all the training samples, $E > 0$, this detector does not match the self. It can, therefore, be used for detecting the nonself. It should be emphasized in the neural networks-based NSA, both $[w_1, w_2, \cdots, w_N]$ and $[v_1, v_2, \cdots, v_N]$

are trainable. However, different from the normal gradient descent oriented Back-Propagation (BP) training method (Haykin 1998) that always tries to minimize E, a 'positive' learning algorithm is derived here for these two sets of weights. The goal of such training is actually to raise the matching error between the NSA detectors and self samples so that the qualified detectors can be generated. Figure 12 illustrates the principle of this training approach. Let $\mathbf{w}$ and $\mathbf{v}$ denote $[w_1, w_2, \cdots, w_N]$ and $[v_1, v_2, \cdots, v_N]$, respectively. Starting from a given point ($\mathbf{w}^*$, $\mathbf{v}^*$) in the weight-matching error ($(\mathbf{w}, \mathbf{v})$ - $E(\mathbf{w}, \mathbf{v})$) space, in order to decrease $E(\mathbf{w}, \mathbf{v})$ by means of the regular BP learning, weights $\mathbf{w}^*$ and $\mathbf{v}^*$ must be changed by $\Delta\mathbf{w}^*$ and $\Delta\mathbf{v}^*$, which are proportional to the *negative* gradient value of $E(\mathbf{w}^*, \mathbf{v}^*)$:

$$\Delta\mathbf{w}^* = -\eta\nabla_{\mathbf{w}^*}E(\mathbf{w}^*, \mathbf{v}^*) = -\mu\frac{\partial E(\mathbf{w}^*, \mathbf{v}^*)}{\partial\mathbf{w}^*}, \tag{8}$$

$$\Delta\mathbf{v}^* = -\eta\nabla_{\mathbf{v}^*}E(\mathbf{w}^*, \mathbf{v}^*) = -\mu\frac{\partial E(\mathbf{w}^*, \mathbf{v}^*)}{\partial\mathbf{v}^*}, \tag{9}$$

where μ is the learning rate. On the other hand, in the new training algorithm of the proposed neural networks-based NSA, $\Delta\mathbf{w}^*$ and $\Delta\mathbf{v}^*$ are chosen to indeed increase $E(\mathbf{w}, \mathbf{v})$ for appropriate detectors generation and tuning. This is accomplished by employing the *positive* gradient information, as the solid line in Fig. 12 shows, in the $\Delta\mathbf{w}^*$ and $\Delta\mathbf{v}^*$ calculation. Therefore, there are:

$$\Delta\mathbf{w}^* = \eta\nabla_{\mathbf{w}^*}E(\mathbf{w}^*, \mathbf{v}^*) = \mu\frac{\partial E(\mathbf{w}^*, \mathbf{v}^*)}{\partial\mathbf{w}^*}, \tag{10}$$

$$\Delta\mathbf{v}^* = \eta\nabla_{\mathbf{v}^*}E(\mathbf{w}^*, \mathbf{v}^*) = \mu\frac{\partial E(\mathbf{w}^*, \mathbf{v}^*)}{\partial\mathbf{v}^*}, \tag{11}$$

$w_i^{(k)}$ and $v_i^{(k)}$ are denoted as the instant values of w_i and v_i at iteration k, respectively. They can be updated:

$$w_i^{(k+1)} = w_i^{(k)} + \mu\frac{\partial E}{\partial w_i^{(k)}} = w_i^{(k)} + \mu\frac{\partial y}{\partial w_i^{(k)}} = w_i^{(k)} - 2\mu v_i^{(k)}\,\mathrm{f}'\left\{\left[x_i - w_i^{(k)}\right]^2\right\}\left[x_i - w_i^{(k)}\right], \tag{12}$$

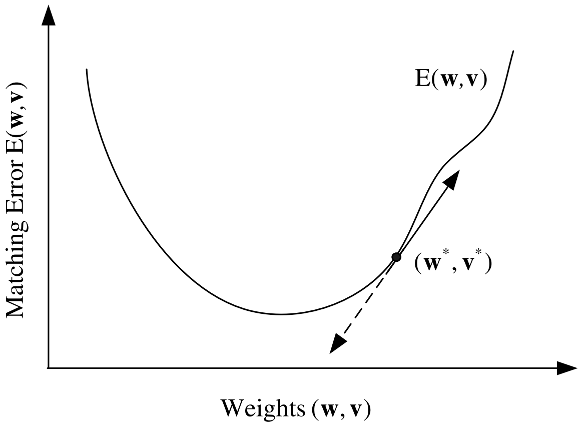


Fig. 12. Principle of training algorithm for neural networks-based NSA detectors (Dotted line: negative gradient, Solid line: positive gradient)

$$v_i^{(k+1)} = v_i^{(k)} + \mu \frac{\partial E}{\partial v_i^{(k)}} = v_i^{(k)} + \mu \frac{\partial y}{\partial v_i^{(k)}} = v_i^{(k)} + \mu \mathrm{f}\left\{\left[x_i - w_i^{(k)}\right]^2\right\}. \tag{13}$$

The above training algorithm has several distinguishing advantages. Firstly, it provides a more effective detector generation scheme than the conventional solutions, in which the binary or real-valued detectors are initially generated in a stochastic manner. Secondly, if applied as a postprocessing phase, it can automatically fine-tune those rough detectors. Lastly, embedded with the advantageous characteristics of adaptation, the neural networks-based NSA is capable of coping with the time-varying anomaly detection problems. The application of this new NSA in fault diagnosis will be next demonstrated.

B. Fault Diagnosis Using Neural Networks-Based Negative Selection Algorithm

The neural networks-based fault diagnosis scheme consists of four stages. Firstly, the feature signals from healthy as well as faulty plants are sampled and preprocessed. Secondly, a certain number of eligible detectors, S, are generated based on the negative selection principle. Only the feature signals of healthy plants are used at this step. Thirdly, these detectors are trained with the feature signals of both healthy and faulty plants. In addition to the learning algorithm presented above, a 'margin' training strategy is also developed for the neural networks-based detectors. The detector weights are separately adapted in the following two cases (regions), refer to Fig. 13.

Case 1 (for the faulty plant feature signals only): if $0 < E < \gamma$ (Training Region I in Fig. 13), the detectors are trained using the normal BP learning algorithm to decrease E; otherwise, no training is employed.

Case 2 (for the healthy plant feature signals only): if $-\gamma < E < 0$ (Training Region II in Fig. 13), the detectors are trained using the above 'positive' BP learning algorithm to increase E ; otherwise, no training is employed.

γ, the margin parameter, is a minor percentage of λ, e.g., 0.1λ. Finally, after this weight refinement stage, the detectors can be deployed to detect possible anomalies, i.e., existing faults. The effectiveness of our new neural networks-based NSA in motor fault diagnosis is verified using computer simulations.

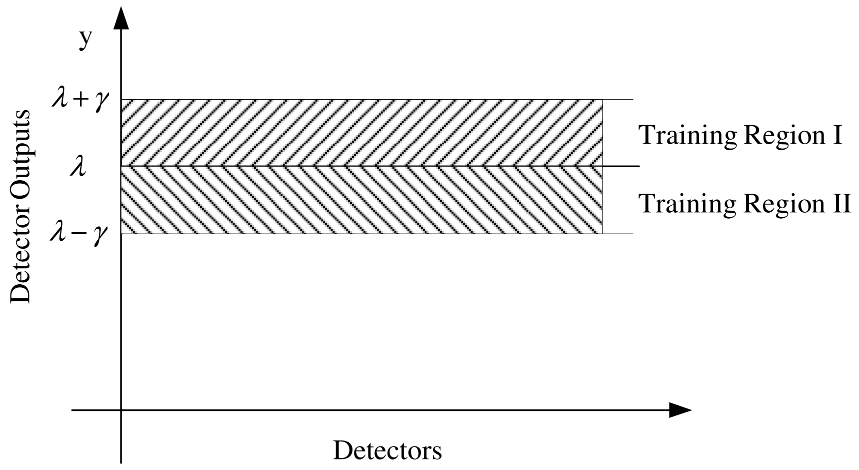


Fig. 13. Margin training regions of neural networks-based NSA detectors in fault diagnosis

C. Simulations

Two representative testbeds, anomaly detection in chaotic time series and inner raceway fault detection of motor bearings, are employed in the simulations.

1. Anomaly detection in chaotic time series

As a popular benchmark for intelligent data analysis methods, the Mackey-Glass chaotic time series, $x(t)$, can be generated by the following nonlinear differential equation (Mackey and Glass 1977):

$$\dot{x}(t) = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t), \tag{14}$$

where $a = 0.1$, $b = 0.2$, and $c = 10$. Figures 14 (a) and (b) illustrate two different Mackey-Glass time series with $\tau = 30$ and $\tau = 17$, respectively. Suppose the normal Mackey-Glass time series result from $\tau = 30$. The goal of utilizing the neural networks-based NSA here is to detect the anomaly caused by $\tau = 17$. Hence, as in the above detector training steps, two 500-sample data sets are collected from the cases of $\tau = 30$ and $\tau = 17$, respectively, and cascaded together as one compact training set of 1,000 samples. The related NSA parameters are given as follows:

number of detectors: $S = 100$,
window width: $N = 10$,
matching threshold: $\lambda = 5.25$,
margin coefficient: $\gamma = 0.05\lambda = 0.26$.

To validate the neural networks-based NSA, the *fresh* anomalous Mackey-Glass time series shown in Fig. 14 (c) are used, where those samples from 1 to 500 correspond to $\tau = 30$, and the succeeding 500 samples $\tau = 17$. The anomaly detection results acquired from the untrained (before training) and trained (after training) NSA detectors are demonstrated in Figs. 15 (a) and (b), respectively. There are only 100 detector instants illustrated in Fig. 16, since the window width N is 10. Note that the untrained

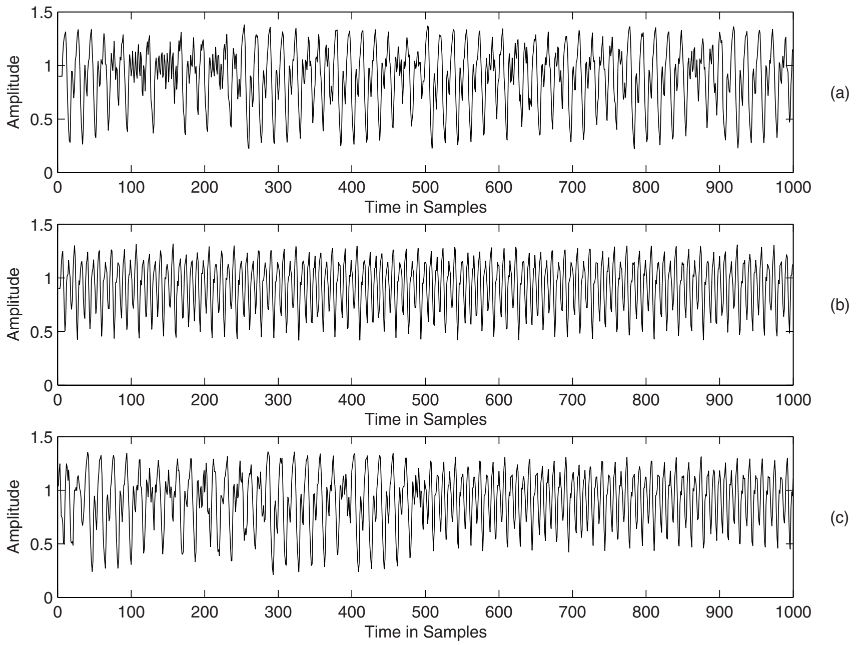


Fig. 14. Mackey-Glass time series ((a) $\tau = 30$, (b) $\tau = 17$, (c) fresh anomalous data for verification)

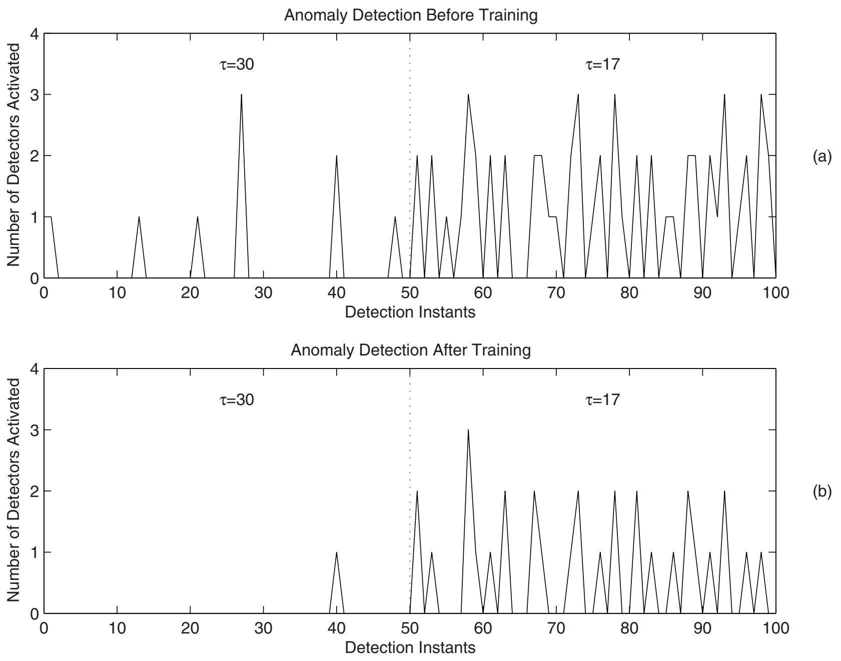


Fig. 15. Anomaly detection in Mackey-Glass time series ((a) before detector training, (b) after detector training)

detectors are randomly initialized and generated from the normal Mackey-Glass time series, i.e., the first half of the 1,000-sample training data set. Apparently, before training, there are:

$$M = 57,\ L = 9,\ \text{and}\ \eta = \frac{M}{M + L} \times 100\% = 86\%,$$

and after training there are:

$$M = 31,\ L = 1,\ \text{and}\ \eta = \frac{M}{M + L} \times 100\% = 97\%.$$

Although the proposed detector adaptation algorithms decrease the numbers of both incorrectly and correctly activated detectors, η actually has grown from 86% to 97%. In other words, an improved anomaly detection performance can be achieved by using this neural networks-based NSA.

2. Inner raceway fault detection of motor bearings

The proposed neural networks-based NSA is also examined using the above inner raceway fault detection problem. In the preprocessing unit, for the convenient manipulation with detector matching error E in Eq. (7), a small constant 0.08 is added to all the signal samples so that both the training and testing data sets contain only positive values. The parameters of the neural networks-based NSA are:

number of detectors: $S = 100$,
window width: $N = 5$,
matching threshold: $\lambda = 4.6$,
margin coefficient: $\gamma = 0.05\lambda = 0.23$.

The relationship between the fault detection rate η and training epochs is depicted in Fig. 16. Fresh feature signals from faulty bearings are deployed as the verification data.

It is clearly visible that η increases with moderate oscillations, when the number of training epochs grows. These oscillations are due to the nature of the gradient descent principle. In the simulations, η is only 79% ($M = 27$ and $L = 7$) at the beginning of training, but finally reaches 100% ($M = 15$ and $L = 0$) after about 800 iteration steps. This indeed demonstrates the neural networks-based NSA can significantly improve the fault detection rate.

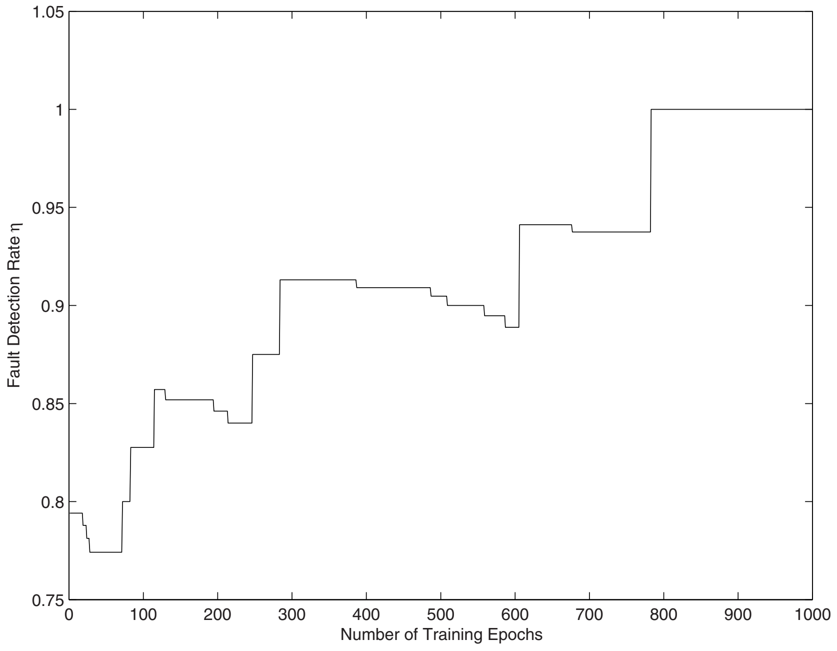


Fig. 16. Fault detection rate η of neural networks-based NSA

6 Conclusions

In this chapter, the basic principle of the NSA is first introduced. Its applications in motor fault detection are next presented. The main feature of applying the NSA in detecting motor faults is that only negative data samples are needed for the detectors generation, which makes this approach moderately easy to implement. We also discuss two modified NSA in details, clonal optimization-based NSA and neural networks-based NSA. The testbeds of anomaly detection in chaotic time series and bearings fault detection are employed to examine the proposed schemes. Both simulated and real-world data is used for validation. Enhanced performances of anomaly detection and bearings fault detection have been obtained in computer simulations. Study of time-varying fault diagnosis problems using the NSA will be an interesting research topic for the future work.

Acknowledgments

This research work was funded by the Academy of Finland under Grant 214144.

References

Ada, G.L., Nossal, G.J.V.: The clonal selection theory. Scientific American 257(2), 50–57 (1987)

Ayara, M., et al.: Negative selection: how to generate detectors. In: Proceedings of the 1st international conference on artificial immune systems, Canterbury, UK, pp. 89–98 (2002)

Chow, M.Y.: Methodologies of using neural network and fuzzy logic technologies for motor incipient fault detection. World Scientific Publishing, Singapore (1997)

Dasgupta, D.: Advances in artificial immune systems. IEEE Computational Intelligence Magazine 1(4), 40–49 (2006)

Dasgupta, D., Attoh-Okine, N.: Immunity-based systems: a survey. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, Orlando, FL, pp. 369–374 (1997)

Dasgupta, D., Forrest, S.: Tool breakage detection in milling operations using a negative selection algorithm. Technical Report CS95-5, Department of Computer Science, University of New Mexico, NM (1995)

Dasgupta, D., González, F.: An immunity-based technique to characterize intrusions in computer networks. IEEE Transactions on Evolutionary Computation 6(3), 281–291 (2002)

Dasgupta, D., et al.: Negative selection algorithm for aircraft fault detection. In: Proceedings of the 3rd international conference on artificial immune systems, Catania, Sicily, Italy, pp. 1–13 (2004)

de Castro, L.N., Timmis, J.: Artificial immune systems: a new computational intelligence approach. Springer, London (2002)

de Castro, L.N., von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6(3), 239–251 (2002)

Forrest, S., et al.: Self-nonself discrimination in a computer. In: Proceedings of the IEEE symposium on research in security and privacy, Los Alamos, CA, pp. 202–212 (1994)

Frank, P.M.: Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results. Automatica 26(3), 459–474 (1990)

Gao, X.Z., et al.: A neural networks-based adaptive negative selection algorithm with application in motor fault diagnosis. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, The Hague, The Netherlands, pp. 3408–3414 (2004)

Gao, X.Z., et al.: Clonal optimization of negative selection algorithm with applications in motor fault detection. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, Taipei, Taiwan, pp. 5118–5123 (2006)

Garrett, S.M.: How do we evaluate artificial immune systems? Evolutionary Computation 13(2), 145–178 (2005)

Goldsby, G.A., et al.: Immunology, 5th edn. W.H. Freeman and Company, New York (2003)

González, F.: A study of artificial immune systems applied to anomaly detection. Ph.D. Dissertation, Division of Computer Science, University of Memphis, TN (2003)

González, F., Dasgupta, D., Nino, L.F.: A randomized real-value negative selection algorithm. In: Proceedings of the 2nd international conference on artificial immune systems, Edinburgh, UK, pp. 261–272 (2003)

Haykin, S.: Neural networks, a comprehensive foundation, 2nd edn. Prentice-Hall, Upper Saddle River (1998)

Ji, Z., Dasgupta, D.: Applicability issues of the real-valued negative selection algorithms. In: Proceedings of the genetic and evolutionary computation conference, Seattle, WA, pp. 111–118 (2006)

Li, B., et al.: Neural-network-based motor rolling bearing fault diagnosis. IEEE Transactions on Industrial Electronics 47(5), 1060–1069 (2000)

Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. Science 197, 287–289 (1977)

Poli, R., Langdon, W.B.: Foundations of genetic programming. Springer, Heidelberg (2002)

Stibor, T., et al.: Is negative selection appropriate for anomaly detection? In: Proceedings of the genetic and evolutionary computation conference, Washington D.C., pp. 321–328 (2005)

Stibor, T., Timmis, J., Eckert, C.: A comparative study of real-valued negative selection to statistical anomaly detection techniques. In: Proceedings of the fourth international conference on artificial immune systems, Banff, Alberta, Canada, pp. 262–275 (2005)

Wang, X., Gao, X.Z., Ovaska, S.J.: Artificial immune optimization methods and applications – a survey. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, The Hague, The Netherlands, pp. 3415–3420 (2004)

Wang, X., Gao, X.Z., Ovaska, S.J.: A hybrid optimization algorithm in power filter design. In: Proceedings of the 31st annual conference of the IEEE industrial electronics society, Raleigh, NC, pp. 1335–1340 (2005)

Yoo, J., Hajela, P.: Immune network simulations in multicriterion design. Structural Optimization 18(2–3), 85–94 (1999)

Harmony Search Applications in Industry

Zong Woo Geem

Environmental Planning and Management, Johns Hopkins University, USA

1 Introduction

In this chapter, the recently-developed music-inspired harmony search (HS) algorithm is introduced and its various industrial applications are reviewed.

The HS algorithm (Geem et al 2001) mimics the behaviors of musicians improvising to find a fantastic harmony in terms of aesthetics. Similarly, the optimization process seeks a superior vector in terms of objective function. This is the core analogy between improvisation and optimization in the HS algorithm.

This soft-computing algorithm could overcome the drawbacks of conventional calculus-based optimization techniques because it has a novel derivative based on solution density and selection probability (Geem et al 2001). This new derivative provides a search direction for discrete decision variables that are undifferentiable. For example, the design variables (= pipe diameters) in water distribution networks are discrete because they are commercially manufactured in factories (Mott 2005). Also, HS does not require initial values for decision variables, which gives HS an increasing flexibility in finding global optimum (Geem, 2006a).

The HS algorithm has been successfully applied to various benchmark and recreational examples, such as Rosenbrock's banana function (Lee and Geem 2005), multiple local optima functions (Lee and Geem 2005), the traveling salesperson problem (Geem et al 2001), artificial neural networks (Geem et al 2002), various continuous functions (Tian et al. 2005; Mahdavi et al. 2007), tour route planning (Geem et al. 2005b), Sudoku puzzle solving (Geem 2007c), and music composition (Geem and Choi 2007).

Also, HS has been extensively applied to various real-world industrial problems as follows:

- **Civil Engineering**
  - Water network design (Geem 2006b & c)
  - Multiple Dam Scheduling (Geem 2007a)
- **Structural Engineering**
  - Dome truss design (Lee and Geem 2004)
  - Grillage structure design (Erdal and Saka 2006)
  - Mix proportioning of steel and concrete (Lee 2004)
- **Traffic Engineering**
  - School bus routing (Geem et al. 2005a)

- **Aerospace Engineering**
  - Satellite heat pipe design (Geem and Hwangbo 2006)
- **Petroleum Engineering**
  - Petroleum structure mooring (Ryu et al. 2007)
- **Industrial Engineering**
  - Fluid-transport minimal spanning tree (Geem and Park 2006)
- **Environmental Engineering**
  - Parameter calibration of flood routing model (Kim et al. 2001)
  - Parameter calibration of rainfall-runoff model (Paik et al. 2005)
- **Energy Engineering**
  - Energy-saving pump operation (Geem 2005)
- **Mechanical Engineering**
  - Pipeline leakage detection (Kim et al. 2006)
- **Chemical Engineering**
  - Prediction of oil well heat wash (Liu and Feng 2004)
- **Geological Engineering**
  - Soil slope stability (Li et al. 2005)
- **Agricultural Engineering**
  - Large-scale irrigation network design (Geem 2007b)
- **Information Technology**
  - Web-based optimization (Geem and Geem 2007)

As the above list demonstrates, HS applications in industry are described in various journals, conference proceedings and degree theses. Since Geem and Tseng (2002) first summarized the industrial applications, many articles have been published recently. Thus, the objective of this chapter is to provide a common understanding of the "big picture" of HS algorithm applications for researchers and students in the above mentioned fields.

2 Harmony Search Algorithm

The music-based HS algorithm has the following six steps:

2.1 Problem Formulation

An optimization problem solved by the HS algorithm can be formulated in a general framework as follows (Mays and Tung 1992):

$$\begin{aligned} \text{Optimize} \quad & f(\boldsymbol{x}) \\ \text{Subject to} \quad & \mathbf{g}(\boldsymbol{x}) \geq 0 \\ & \mathbf{h}(\boldsymbol{x}) = 0 \\ & \boldsymbol{x} \in \boldsymbol{S} \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a solution vector of n decision variables ($x_1, x_2, \ldots, x_n$), $f(\boldsymbol{x})$ is an objective function, $\mathbf{g}(\boldsymbol{x})$ is a vector of inequality constraints, $\mathbf{h}(\boldsymbol{x})$ is a vector of equality constraints, and $\boldsymbol{S}$ is the solution space. If the decision variable x_i is discrete, $\boldsymbol{S} = \{x_i(1), x_i(2), \ldots, x_i(k), \ldots, x_i(K)\}$; if x_i is continuous, $\underline{x_i} \le \boldsymbol{S} \le \overline{x_i}$.

2.2 Harmony Memory Initialization

Before a new harmony is improvised (= generated) by HS, a matrix, named harmony memory (HM), is filled with a group of randomly-generated vectors.

$$\left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_n^1 & f(\boldsymbol{x}^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & f(\boldsymbol{x}^2) \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_n^{HMS} & f(\boldsymbol{x}^{HMS}) \end{array}\right] \tag{2}$$

In Eq. 2, each row represents each solution vector, and the number of total vectors is HMS (harmony memory size).

In order to start HS computation with better HM, random vectors may be generated more than HMS, then only better vectors are collected in the HM.

2.3 New Harmony Improvisation

Once the HM is prepared, a new vector (= harmony) $\boldsymbol{x}^{New} = (x_1^{New}, x_2^{New}, \ldots, x_n^{New})$ is improvised based on the following three operations.

2.3.1 Memory Consideration

A value of decision variable x_i^{New} can be chosen from any values stored in the HM with a probability of HMCR (harmony memory considering rate).

$$x_i^{New} \leftarrow x_i^{New} \in \{x_i^1, x_i^2, \ldots, x_i^{HMS}\} \quad \text{w.p.} \quad HMCR \tag{3}$$

2.3.2 Random Selection

Instead of memory consideration, a value of decision variable x_i^{New} can be chosen from any values in solution space $\boldsymbol{S}_i$ rather than in the HM.

$$x_i^{New} \leftarrow x_i^{New} \in \boldsymbol{S}_i \quad \text{w.p.} \quad (1 - HMCR) \tag{4}$$

2.3.3 Pitch Adjustment

Once a value of decision variable x_i^{New} is obtained from the memory consideration operation rather than the random selection operation, the value can be further modified into its neighboring values with a probability of PAR (pitch adjusting rate).

$$x_i^{New} \leftarrow \begin{cases} x_i(k+m) & \text{w.p.} \quad PAR \\ x_i(k) & \text{w.p.} \quad (1 - PAR) \end{cases} \tag{5}$$

where $x_i(k)$ is identical to the value of the decision variable x_i^{New} obtained in the memory consideration operation; and neighboring index m can have -1 or 1.

For continuous-valued variables, the following scheme is used for pitch adjustment (Lee and Geem 2005).

$$x_i^{New} \leftarrow \begin{cases} x_i^{New} + \Delta & \text{w.p.} \quad PAR \\ x_i^{New} & \text{w.p.} \quad (1 - PAR) \end{cases} \tag{6}$$

where Δ is non-uniform amount for pitch adjustment.

2.4 Other Consideration

2.4.1 Ensemble Consideration

A value of decision variable x_i^{New} can be chosen from the relationship among decision variables (Geem, 2006d).

$$x_i^{New} \leftarrow f(x_j^{New}) \quad \text{where} \quad \max_{i \neq j} \left\{ | Corr(\boldsymbol{x}_i, \boldsymbol{x}_j) | \right\} \tag{7}$$

2.4.2 Rule Violation

Once the new harmony vector $\boldsymbol{x}^{New}$ is generated, it is then checked to determine whether it violates any constraint. As accomplished composers like Bach and Beethoven used rule-violated harmonies (for example, parallel fifths), HS uses rule-violated vectors by adding a penalty (Geem and Choi 2007) to the objective function value.

2.5 Harmony Memory Update

If the new harmony vector $\boldsymbol{x}^{New}$ is better than the worst harmony vector in the HM in terms of the objective function value, including the penalty, the new vector is included in the HM and the worst vector is excluded from the HM. There may exist a maximum number of identical vectors in order to prevent premature HM.

2.6 Termination of Computation

If the number of harmony improvisations reaches MaxImp (maximum improvisations), the computation stops. Otherwise, the process described in sections 2.3 - 2.6 is repeated.

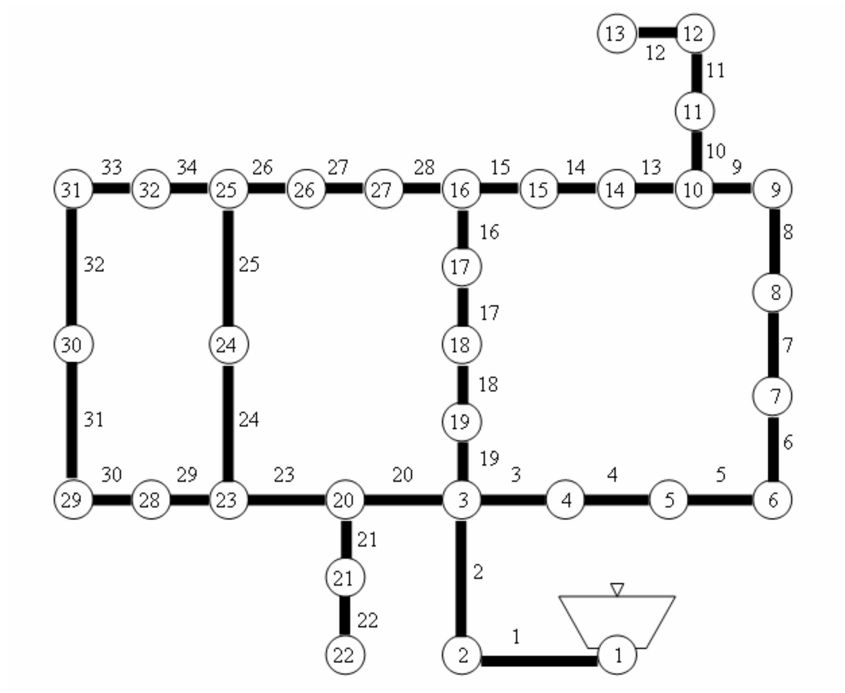


Fig. 1. Hanoi water distribution network

3 Harmony Search Applications in Industry

3.1 Water Network Design

The harmony search algorithm was applied to the design of water distribution networks. Figure 1 shows one of examples (Geem 2006b; Geem and Kim 2007). The objective of the problem is to minimize design cost by choosing minimal pipe diameters while satisfying pressure and quantity demand at each node.

HS was successfully applied to the water network design, and its result was compared with those of other soft computing algorithms, such as genetic algorithm (GA) (Wu et al. 2001), simulated annealing (SA) (Cunha and Sousa 2001), tabu search (TS) (Cunha and Ribeiro 2004), ant colony optimization (ACO) algorithm (Zecchin et al. 2006), and shuffled frog leaping (SFL) algorithm (Eusuff and Lansey 2003). The feasible solution obtained by HS ($6,081,000) was the best one when compared with those of GA ($6,183,000), SA ($6,093,000), and ACO ($6,134,000). Although TS ($6,056,000) and SFL ($6,073,000) claimed that they found better solutions, their solutions violated the nodal pressure constraint when examined with a standard hydraulic analyzer, EPANET 2.0 (Rossman 2000).

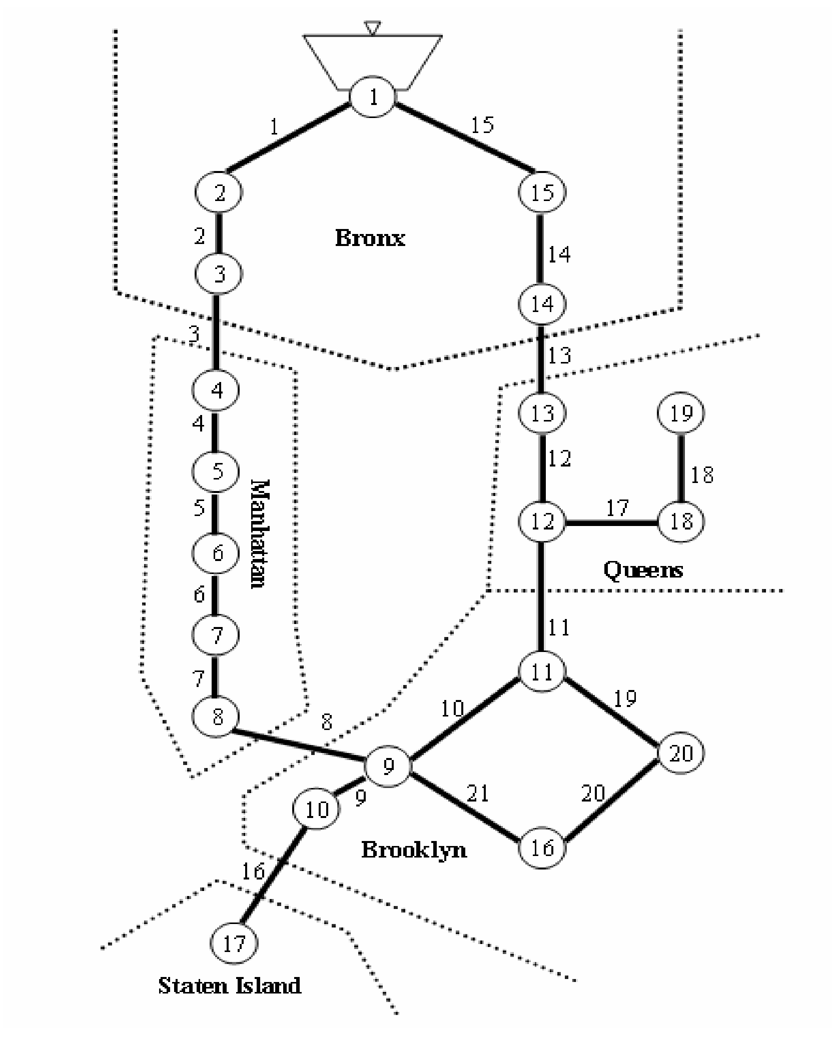


Fig. 2. New York City water distribution network

HS also was applied to the expansion problem for a water network shown in Figure 2. The objective of the problem is to find minimal diameters of additional pipes while satisfying pressure requirements at remote nodes.

HS (Geem 2006c), GA (Broad et al. 2005), ACO (Maier et al. 2003), and CE (Perelman and Ostfeld 2005) found the identical lowest expansion cost ($38.64 million) while SA (Cunha and Sousa 2001) and SFL (Eusuff and Lansey 2003) found the second best cost ($38.80 million). Although Cunha and Ribeiro (2004) claimed that the TS found $37.13 million, their solution violated nodal pressure constraint at three nodes (nodes 16, 17, and 19).

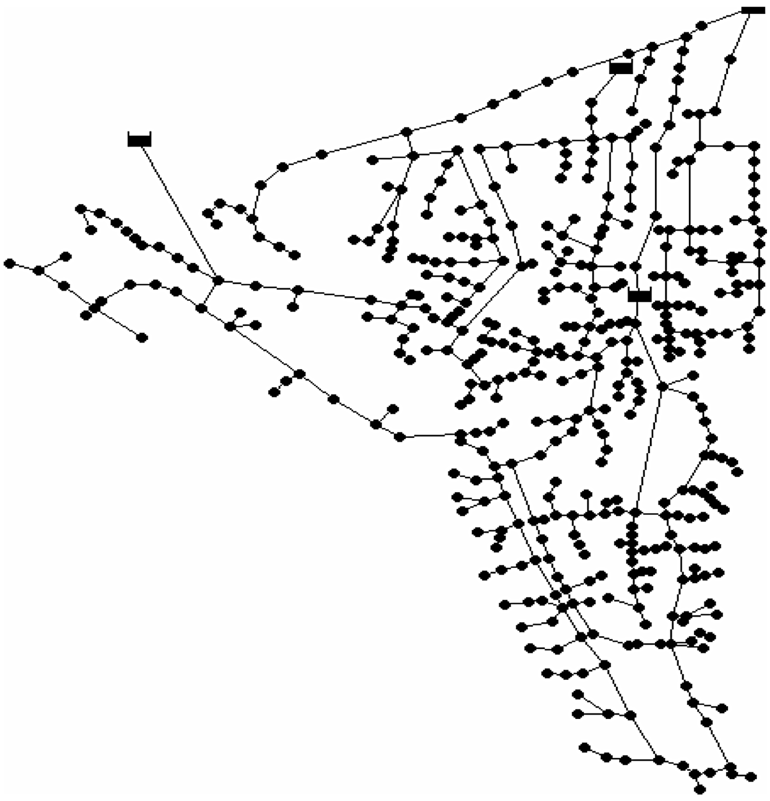


Fig. 3. Balerma water distribution network

Among the soft-computing algorithms that reached the identical least cost ($38.64 million), HS reached the cost with the least number of objective function evaluations: HS made a determination after 3,373 evaluations (2.5 seconds on Intel Celeron 1.8GHz) while ACO after 7,014 evaluations, CE after 70,000 evaluations, and GA after 800,000 evaluations.

HS also challenged a large-scale water network (4 reservoirs, 8 loops, 443 nodes, and 454 pipes) shown in Figure 3.

Reca and Martinez (2006) originally proposed the network, and obtained 2.302 million € using improved GA (an integer coding scheme, use of penalty multiplier, rank-based roulette wheel, three crossover strategies, uniform one-point mutation, and steady-state-delete-worst reproduction plan). Geem (2007b) applied HS to the same problem, and obtained 2.018 million € (12.4% less than that of GA).

3.2 Structural Design

The HS algorithm has been applied to the design of various structures. The objective of the problem is to minimize total weight by choosing minimal member size

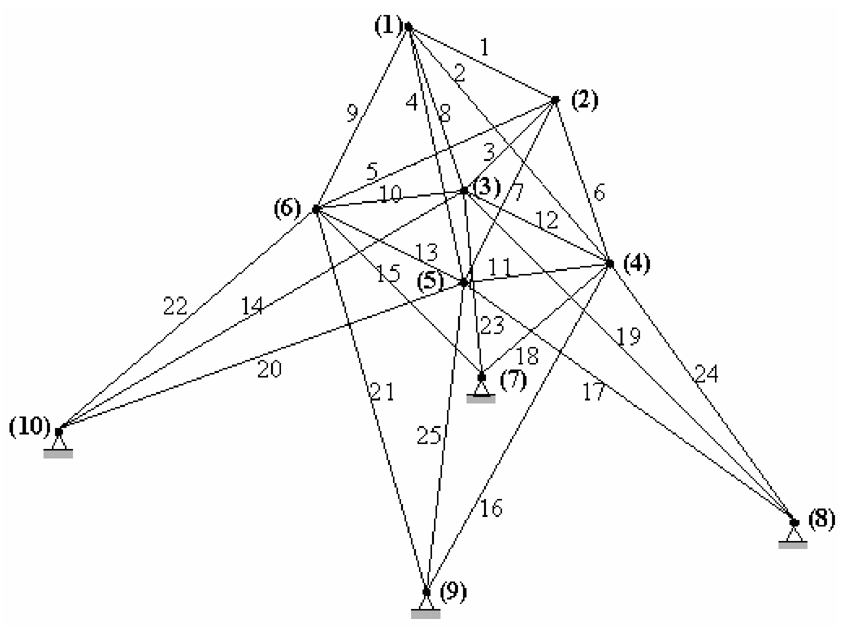


Fig. 4. 25-member transmission tower

while satisfying stress, displacement, and buckling limit constraints that can be checked by the finite element method (FEM) routine. Figure 4 shows one of structural examples.

For the 25-member tower problem, HS (Geem et al. 2005c) found the least weight (484.85 lb) while GA 485.05 lb (Camp et al. 1998), SA 537.23 lb (Park and Sung 2002), and artificial neural network (ANN) 543.95 lb (Adeli and Park 1996).

HS was also applied to the design of a dome structure shown in Figure 5, and found a reasonable solution.

HS was sometimes combined with other algorithms to solve optimization problems. Li et al. (2007) developed an algorithm (HPSO) based on HS and particle swarm, and applied it to various structural designs. While the HPSO obtained better solutions than those of most algorithms, it could not outperform the original HS (Lee and Geem 2004) in most cases.

3.3 Dam Scheduling

A dam is a barrier structure built across a river or stream to store water. HS was applied to the scheduling of multiple dam system as shown in Figure 6.

The objective of the problem is to maximize total benefits from both hydropower generation and irrigation by choosing water release amount at each dam while satisfying release and storage limit constraints.

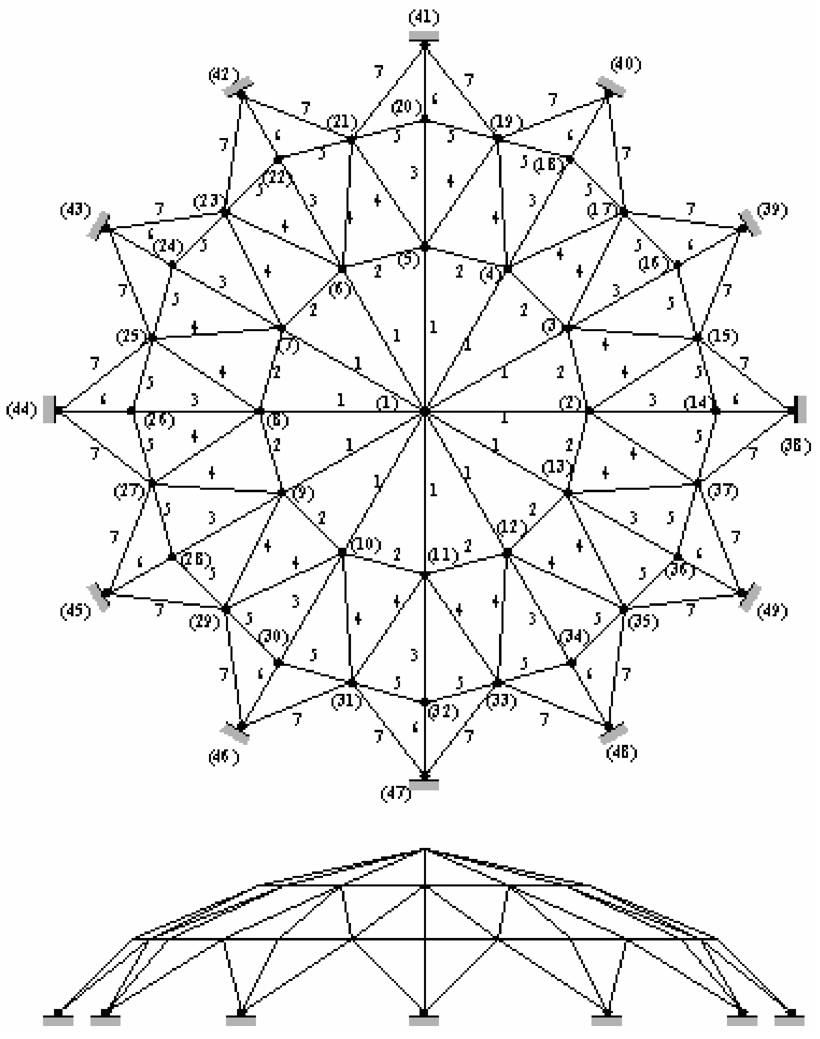


Fig. 5. 120-member dome structure

Wardlaw and Sharif (1999) obtained near-optimum benefits (400.5 units total) using improved GA (binary, gray & real-value representations, tournament selection, three crossover strategies, and uniform & modified uniform mutation), whereas Geem (2007a) obtained five different global optima (401.3 units) using HS (HMS = 30, HMCR = 0.95, PAR = 0.05, and MaxImp = 35,000).

Figure 7 shows one example of the water release trajectories from five global optima.

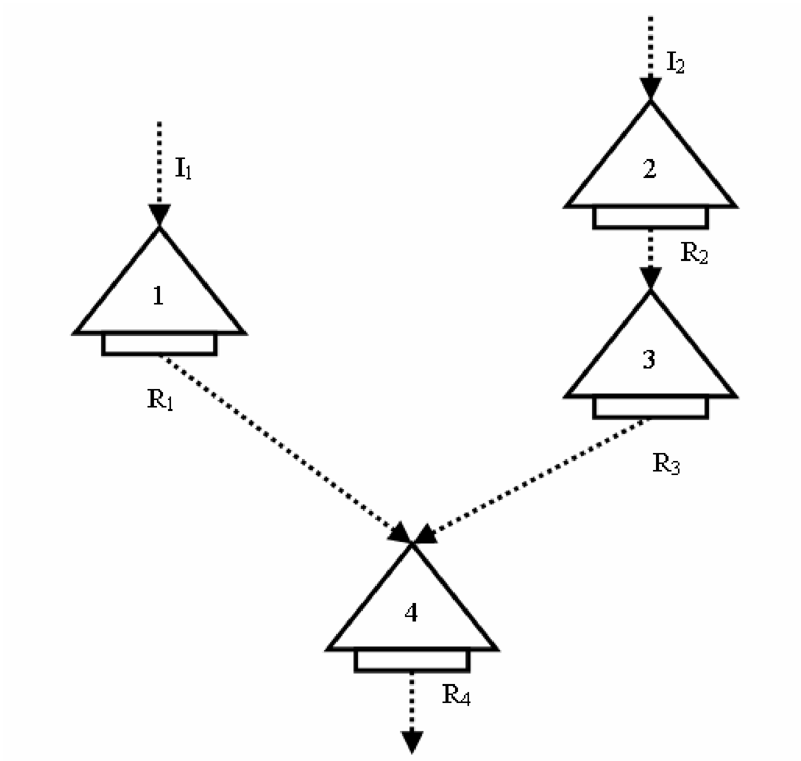


Fig. 6. Four-dam system

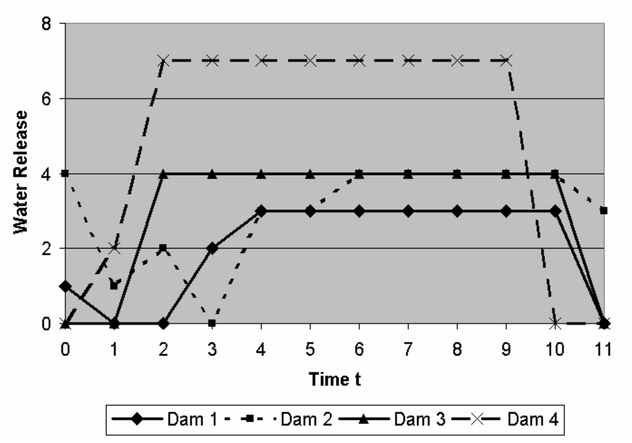


Fig. 7. Water release trajectory at each dam

3.4 Vehicle Routing

The vehicle routing problem (VRP) requires one to design a set of routes for multiple vehicles from a single depot in order to serve a set of customers who are geographically dispersed. HS was applied to a school bus routing problem, one of VRP's, as shown in Figure 8.

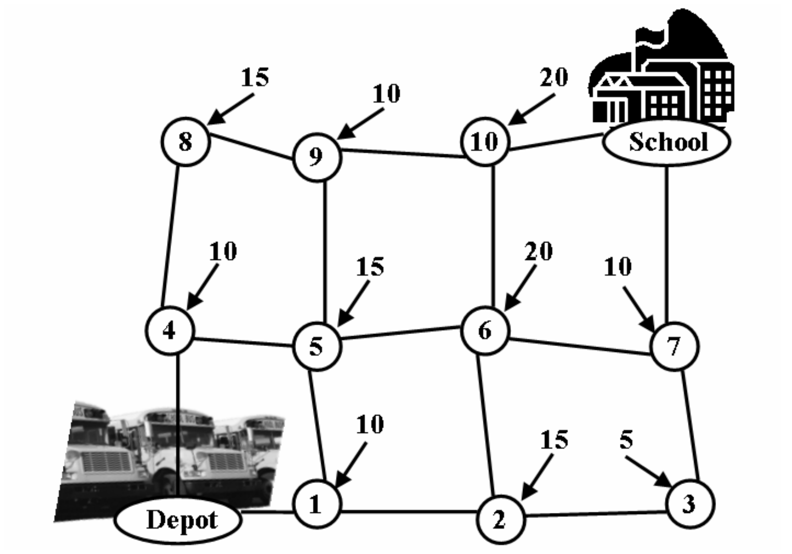


Fig. 8. School bus routing network

The objective of the problem is to minimize both the number of buses and their travel times, while satisfying bus capacity and time-window constraints.

HS found an average solution of \$399,870 over 20 different runs, while the GA found that of \$409,597 over the same number of runs (Geem et al. 2005a).

3.5 Satellite Heat Pipe Design

A heat pipe is heat transfer device having effective thermal conductivity and the capability of transporting heat over a considerable distance. In space, a satellite needs this device to balance temperature over its entire body. HS was applied to the design of a satellite heat pipe, as shown in Figure 9.

The multiple objectives of the problem are to maximize the heat transfer conductance, as well as to minimize total mass (Because the heat pipe is used in space, mass instead of weight is considered).

HS (Geem and Hwangbo 2006) found a Pareto solution (thermal conductance = 0.381 W/K and total mass = 26.7 kg) when compared with that of Broyden-Fletcher-Goldfarb-Shanno (BFGS) technique (thermal conductance = 0.375 W/K and total mass = 26.9 kg).

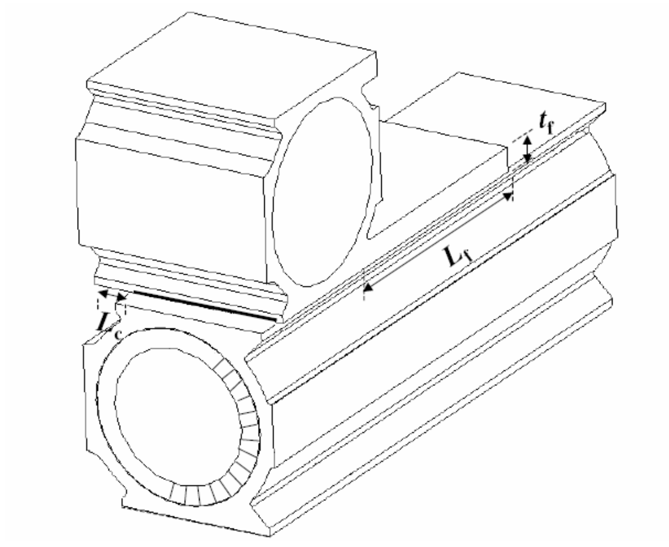


Fig. 9. Satellite heat pipe

3.6 Petroleum Vessel Mooring

Mooring (or anchoring) means to hold an offshore vessel in a stable fashion. HS was applied to the mooring of a Floating Production, Storage & Offloading (FPSO) vessel, as shown in Figure 10.

The objective of the problem is to minimize the cost of offshore vessel mooring by finding the appropriate size of each component while satisfying the constraints of platform offset, mooring line tension, and bottom chain length.

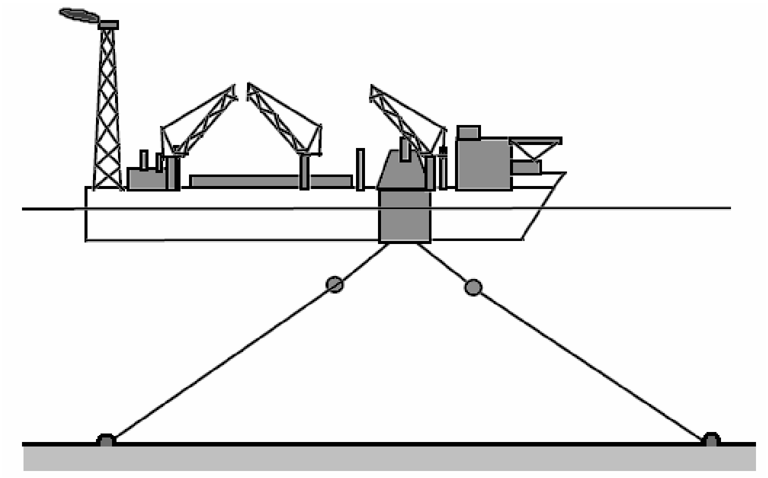


Fig. 10. Floating production, storage & offloading vessel

HS (Ryu et al. 2007) found a reasonable mooring cost (\$4.8 million) after having started with a feasible cost of \$30.0 million.

3.7 Branched Network Layout

HS was applied to the layout design of a branched fluid-transport network as shown in Figure 11. The objective of the problem is to find a minimal-cost branched layout from numerous candidate layouts. For the 64-node rectilinear network in Figure 11, HS (Geem and Park 2006) found the global optimum (layout cost = 5062.8). In contrast, the evolutionary algorithm (EA) (Walters and Smith 1995) found 5095.2 and GA (Walters and Lohbeck 1993) found 5218.0. In addition, HS reached the optimum after 1,500 function evaluations, while the number of total enumerations is 1.26×10^{26}.

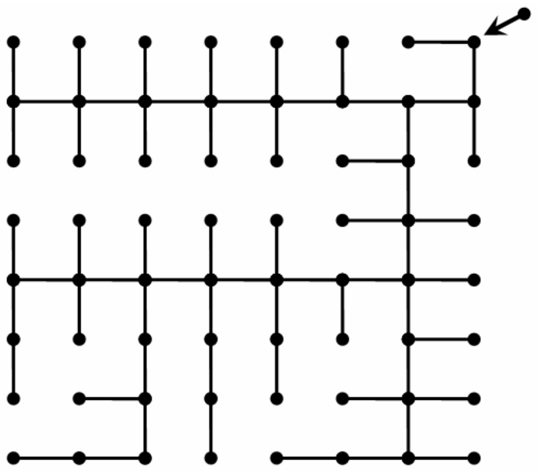


Fig. 11. Schematic of Branched Network

3.8 Model Parameter Calibration

HS was applied to the parameter calibration for a flood routing model, as shown in Figure 12.

The objective of the problem is to minimize the difference between observed and routed (computed) flows by varying model parameter values. HS (Kim et al. 2001) obtained the least error solution (36.78), while the least-squares method (Gill 1978) obtained an error of 145.69, the Lagrange multiplier method (Das 2004) obtained 130.49, the Hooke-Jeeves pattern search (Tung 1985) obtained 45.61, and GA (Mohan 1997) obtained 38.24. The improved HS (Lee and Geem 2005) for considering continuous-valued decision variables found an even better solution (36.77), which is very close to global optimum (36.7679) (Geem 2006a).

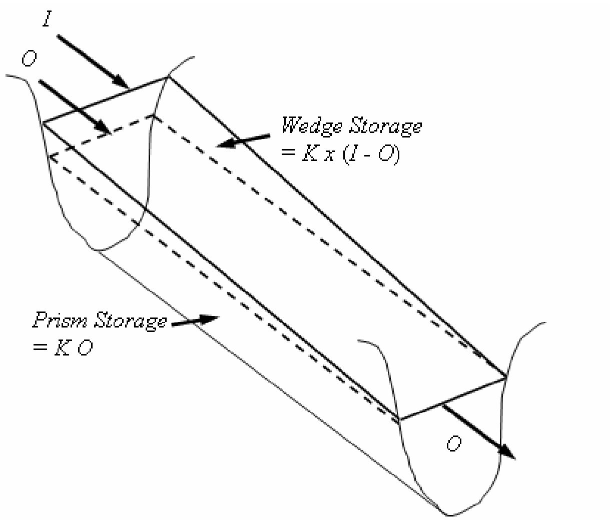
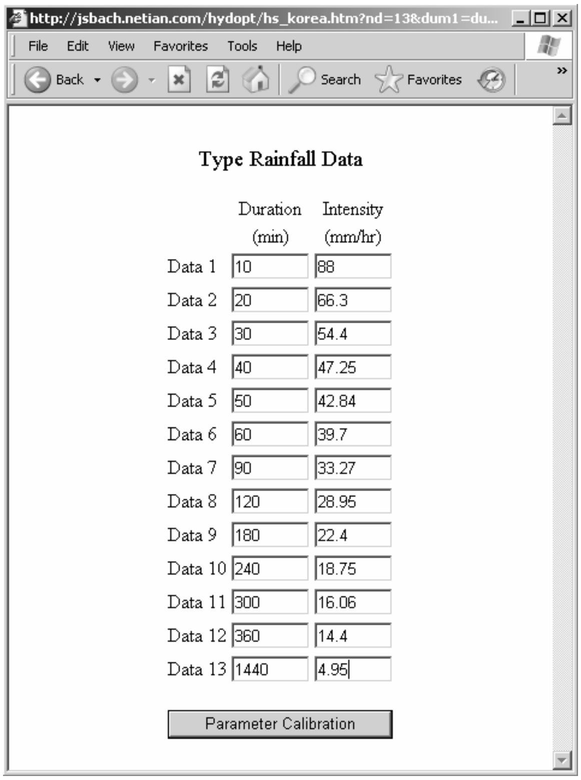


Fig. 12. Schematic of flood routing model


http://jsbach.netian.com/hydopt/hs_korea.htm?nd=13&dum1=du...
File Edit View Favorites Tools Help
Back Search Favorites

Type Rainfall Data

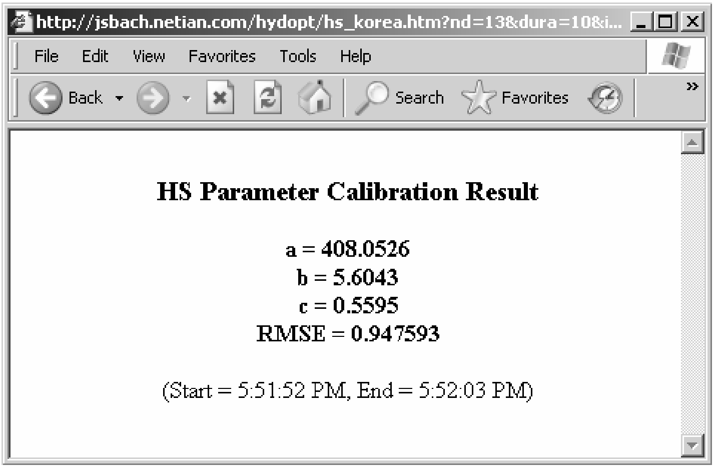
	Duration (min)	Intensity (mm/hr)
Data 1	10	88
Data 2	20	66.3
Data 3	30	54.4
Data 4	40	47.25
Data 5	50	42.84
Data 6	60	39.7
Data 7	90	33.27
Data 8	120	28.95
Data 9	180	22.4
Data 10	240	18.75
Data 11	300	16.06
Data 12	360	14.4
Data 13	1440	4.95

Fig. 13. Screen of hydrologic data entry

3.9 Web-Based Optimization

HS was performed on a web-based platform for the parameter calibration of a rainfall intensity model, as shown in Figure 13.

HS calibrated the hydrologic parameters on a web-browser using a client-side scripting-language called VBScript. HS obtained better solutions than those of Powell and GA (Geem and Geem 2007). Figure 14 shows a result screen of the HS computation.


http://jsbach.netian.com/hydopt/hs_korea.htm?nd=13&dura=10&i...
File Edit View Favorites Tools Help
Back
Search
Favorites

HS Parameter Calibration Result

a = 408.0526
b = 5.6043
c = 0.5595
RMSE = 0.947593

(Start = 5:51:52 PM, End = 5:52:03 PM)

Fig. 14. Screen of HS computation result

4 Conclusions

This chapter showed various industrial applications of the recently-developed music-inspired HS algorithm. The applications include water network design, structural design, dam scheduling, vehicle routing, satellite heat pipe design, petroleum vessel mooring, branched network layout, model parameter calibration, and web-based optimization.

The HS algorithm has generally outperformed other soft-computing algorithms in the above-mentioned problems. When compared with GA, HS explicitly considers the relationship among variables using ensemble operation, while the GA implicitly considers the relationship using building block theory which does not work very well when the variable positions in a chromosome are not carefully considered (Geem 2006d).

When compared with mathematical methods, HS does not require complex calculus or starting values, and it easily handles discrete variables as well as continuous variables. HS also finds better solutions for combinatorial optimization problems when compared with the branch and bound method (Geem 2005).

For future study, a parameter-free HS algorithm should be developed because HS users are currently required to set proper values for algorithm parameters such as HMS, HMCR, and PAR. Also, the development of HS software will facilitate the use of this soft-computing theory by engineers in industry.

References

Adeli, H., Park, H.S.: Hybrid CPN-neural dynamics model for discrete optimization of steel structures. Microcomputer Civil Engineering 11, 355–366 (1996)

Broad, D.R., Dandy, G.C., Maier, H.R.: Water distribution system optimization using metamodels. Journal of Water Resources Planning and Management, ASCE 131, 172–180 (2005)

Camp, C., Pezeshk, S., Cao, G.: Optimized design of two-dimensional structures using a genetic algorithm. Journal of Structural Engineering, ASCE 124, 551–559 (1998)

Cunha, M.C., Ribeiro, L.: Tabu search algorithms for water network optimization. European Journal of Operational Research 157, 746–758 (2004)

Cunha, M.C., Sousa, J.: Hydraulic infrastructures design using simulated annealing. Journal of Infrastructure Systems, ASCE 7, 32–39 (2001)

Das, A.: Parameter estimation for Muskungum models. Journal of Irrigation and Drainage Engineering 130, 140–147 (2004)

Erdal, F., Saka, M.P.: Optimum design of grillage systems using harmony search algorithm. In: Proceedings of 8th International Conference on Computational Structures Technology (CST 2006), Las Palmas de Gran Canaria, Spain (2006) (CD-ROM)

Eusuff, M., Lansey, K.E.: Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management, ASCE 129, 210–225 (2003)

Geem, Z.W.: Harmony search in water pump switching problem. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 751–760. Springer, Heidelberg (2005)

Geem, Z.W.: Parameter estimation for the nonlinear Muskingum model using BFGS technique. Journal of Irrigation and Drainage Engineering, ASCE 132, 474–478 (2006a)

Geem, Z.W.: Optimal cost design of water distribution networks using harmony search. Engineering Optimization 38, 259–280 (2006b)

Geem, Z.W.: Comparison harmony search with other meta-heuristics in water distribution network design. In: Proceedings of 8th Annual International Symposium on Water Distribution Systems Analysis (WDSA 2006), ASCE, Cincinnati, OH, USA (2006c) (CD-ROM)

Geem, Z.W.: Improved harmony search from ensemble of music players. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4251, pp. 86–93. Springer, Heidelberg (2006d)

Geem, Z.W.: Optimal scheduling of multiple dam system using harmony search algorithm. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 316–323. Springer, Heidelberg (2007a)

Geem, Z.W.: Harmony search algorithm for the optimal design of large-scale water distribution network. In: Proceedings of the 7th International IWA Symposium on Systems Analysis and Integrated Assessment in Water Management (Watermatex 2007), IWA, Washington DC, USA (2007b) (CD-ROM)

Geem, Z.W.: Harmony search algorithm for solving Sudoku (submitted. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, pp. 371–378. Springer, Heidelberg (2007c)

Geem, Z.W., Choi, J.Y.: Music composition using harmony search algorithm. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 593–600. Springer, Heidelberg (2007)

Geem, Z.W., Geem, W.B.: Cutting-edge optimization technique and its applications to the civil engineering. Magazine of the Korean Society of Civil Engineers, KSCE 55, 155–171 (2007)

Geem, Z.W., Hwangbo, H.: Application of harmony search to multi-objective optimization for satellite heat pipe design. In: Proceedings of US-Korea Conference on Science, Technology, & Entrepreneurship (UKC 2006), Teaneck, NJ, USA (2006) (CD-ROM)

Geem, Z.W., Kim, J.H.: Efficient design of urban water supply network using improved harmony search. In: Proceedings of the 4th IWA Specialist Conference on Efficient Use and Management of Urban Water Supply (Efficient 2007), IWA, Jeju Island, South Korea (2007) (CD-ROM)

Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: Harmony search. Simulation 76, 60–68 (2001)

Geem, Z.W., Kim, J.H., Loganathan, G.V.: Application of harmony search algorithm to water resources problems. In: Proceedings of 2002 Conference of the Environmental and Water Resources Institute, ASCE, Roanoke, VA, USA (2002) (CD-ROM)

Geem, Z.W., Lee, K.S., Park, Y.: Application of harmony search to vehicle routing. American Journal of Applied Sciences 2, 1552–1557 (2005a)

Geem, Z.W., Lee, K.S., Tseng, C.L.: Harmony search for structural design. In: Proceedings of 2005 Genetic and Evolutionary Computation Conference (GECCO 2005), Washington, DC, USA, pp. 651–652 (2005c)

Geem, Z.W., Park, Y.: Harmony search for Layout of Rectilinear Branched Networks. WSEAS Transactions on Systems 6, 1349–1354 (2006)

Geem, Z.W., Tseng, C.L.: Engineering applications of harmony search. In: Late-Breaking Papers of 2002 Genetic and Evolutionary Computation Conference (GECCO 2002), New York City, USA, pp. 169–173.

Geem, Z.W., Tseng, C.L., Park, Y.: Harmony search for generalized orienteering problem: Best touring in China. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 741–750. Springer, Heidelberg (2005b)

Gill, M.A.: Flood routing by the Muskingum method. Journal of Hydrology 36, 353–363 (1978)

Kim, J.H., Geem, Z.W., Kim, E.S.: Parameter estimation of the nonlinear Muskingum model using harmony search. Journal of the American Water Resources Association 37, 1131–1138 (2001)

Kim, S.H., et al.: Transient analysis and leakage detection algorithm using GA and HS algorithm for a pipeline system. Journal of Mechanical Science and Technology, KSME 20, 426–434 (2006)

Lee, C.H.: Optimized mix proportioning of steel and hybrid fiber reinforced concrete using harmony search algorithm. Master Thesis, Department of Civil and Environmental Engineering, Korea University, South Korea (2004)

Lee, K.S., Geem, Z.W.: A new structural optimization method based on the harmony search algorithm. Computers & Structures 82, 781–798 (2004)

Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering 194, 3902–3933 (2005)

Li, L., Chi, S.C., Lin, G.: Genetic algorithm incorporated with harmony procedure and its application to searching of non-circular critical slip surface in soil slopes (In Chinese). Shuili Xuebao 36, 1–8 (2005)

Li, L.J., et al.: A Heuristic particle swarm optimizer for optimization of pin connected structures. Computers & Structures 85, 340–349 (2007)

Liu, T.N., Feng, Z.B.: Adaptive identification and filtering based on harmony search (In Chinese). Journal of Jilin University 22, 306–309 (2004)

Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Applied Mathematics and Computation 188(2), 1567–1579 (2007)

Maier, H.R., et al.: Ant colony optimization for design of water distribution systems. Journal of Water Resources Planning and Management, ASCE 129, 200–209 (2003)

Mays, L.W., Tung, Y.K.: Hydrosystems engineering and management. McGraw-Hill, New York (1992)

Mohan, S.: Parameter estimation of nonlinear Muskingum models using genetic algorithm. Journal of Hydraulic Engineering 123, 137–142 (1997)

Mott, R.L.: Applied Fluid Mechanics. Prentice-Hall, Englewood Cliffs (2005)

Paik, K., et al.: A conceptual rainfall-runoff model considering seasonal variation. Hydrological Processes 19, 3837–3850 (2005)

Park, H.S., Sung, C.W.: Optimization of steel structures using distributed simulated annealing algorithm on a cluster of personal computers. Computers and Structures 80, 1305–1316 (2002)

Perelman, L., Ostfeld, A.: Water distribution systems optimal design using cross entropy. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005), Washington, DC, USA, pp. 647–648 (2005)

Reca, J., Martinez, J.: Genetic algorithms for the design of looped irrigation water distribution networks. Water Resources Research 42 (2006)

Rossman, L.A.: EPANET2 Users Manual. US Environmental Protection Agency. Cincinnati, OH, USA (2000)

Ryu, S., et al.: Offshore mooring cost optimization via harmony search. In: Proceedings of 26th International Conference on Offshore Mechanics and Arctic Engineering, ASME, San Diego, CA, USA (2007)

Tian, Y.H., Bo, Y.M., Gao, M.F.: Parameters choice criteria in harmony annealing for function optimization (In Chinese). Computer Simulation 22, 70–74 (2005)

Tung, Y.K.: River flood routing by nonlinear Muskingum method. Journal of Hydraulic Engineering 111, 1447–1460 (1985)

Wardlaw, R., Sharif, M.: Evaluation of genetic algorithms for optimal reservoir system operation. Journal of Water Resources Planning and Management, ASCE 125, 25–33 (1999)

Walters, G., Lohbeck, T.: Optimal layout of tree networks using genetic algorithms. Engineering Optimization 22, 27–48 (1993)

Walters, G., Smith, D.: Evolutionary design algorithm for optimal layout of tree networks. Engineering Optimization 24, 261–281 (1995)

Wu, Z.Y., et al.: Using genetic algorithms to rehabilitate distribution systems. Journal of the American Water Works Association 93, 74–85 (2001)

Zecchin, A.C., et al.: Application of two ant colony optimization algorithms to water distribution system optimization. Mathematical and Computer Modelling 44, 451–468 (2006)

Soft Computing in Bioinformatics: Genomic and Proteomic Applications

James Malone

AIAI, University of Edinburgh, Edinburgh, EH8 9LE, UK
j.malone@ed.ac.uk

1 The Age of Bioinformatics

Bioinformatics has been described as the science of managing, mining, and interpreting information from biological sequences and structures (Li *et al* 2004). The emergence of the field has been largely attributed to the increasing amount of biomedical data created and collected and the availability and advancement of high-throughput experimental techniques. One recent example of this is the advancement of 'lab-on-a-chip' (see Figure 1) technology which allows experimentation to be performed more rapidly and at lower cost, whilst introducing the possibility of observing new phenomena or obtaining more detailed information from biologically active systems (Whitesides 2006). Such advances enable scientists to conduct experiments which result in large amounts of experimental data over a relatively short period of time. The need to analyse such experimental data has often necessitated a similarly high-throughput approach in order to produce rapid results, employing the use of efficient and flexible analysis methods and, in many areas, driving the need for every improving data analysis techniques. It is for this reason bioinformatics draws upon fields including, but not limited to, computer science, biology (including biochemistry), mathematics, statistics and physics.

Bioinformatics research, however, has demonstrated that the field is not limited to merely data handling and analysis. Techniques and technologies, for example, for image processing in electrophoresis, have been used at the stages of experimental data collection as well as later in the curation, manipulation and/or analysis of stored databases. The work of Datta *et al* (2007) for instance, uses signal processing techniques from engineering for the processing of genomic signals. Representations of data have also been of great interest in recent time such as the popular gene ontology (GO), as well as other representations (Aitken, 2005; McGarry *et al*, 2007).

An important consideration for a bioinformatician, those working in the biomedical industry or anyone working with data from the bioinformatics field are the issues of imprecision and uncertainty. Data produced as a result of experiments from techniques such as microarrays and gel electrophoresis intrinsically contain sources of experimental variability which can be propagated through to the downstream analysis of the data (Rattray *et al* 2006).

Experimental variation is not a recent phenomenon and has been traditionally dealt with using statistical measures or intervention from experts using heuristics learnt with experience. However, the introduction of large scale analysis using computers presents new issues to those working within the field. The efficacy with which many computational data analysis techniques can be applied to analyse these sorts of biomedical data sets is reduced if such levels of variation can not be accounted for.

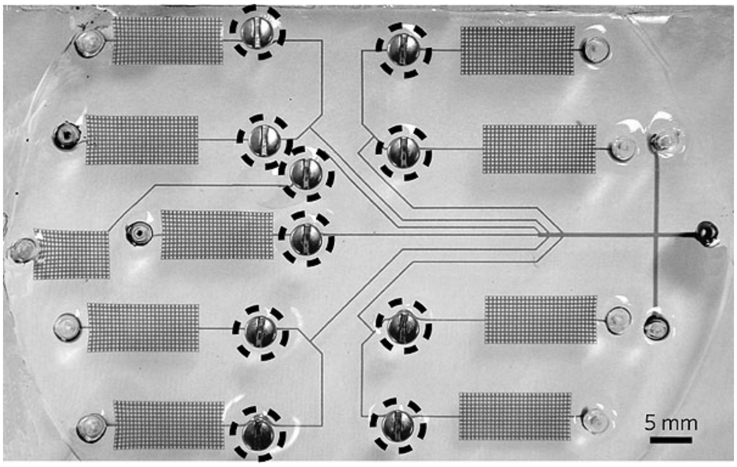


Fig. 1. A simple lab-on-a-chip diagnostic device that is both inexpensive and portable and can produce large quantities of biological experimental data. Taken from Whitesides (2006).

Within the area of soft computing, these types of problems are being explored with a view to offering some potential solutions to such issues. At the heart of soft computing lies the objective of tractable and robust computational reasoning with tolerance of imprecision, uncertainty, partial truths and approximation. The guiding principle is to devise methods of computation that lead to an acceptable solution at low cost by seeking an approximate answer to an imprecisely/precisely formulated problem (Mitra *et al* 2002). It would therefore seem appropriate that the intersection of these two research fields, bioinformatics and soft computing, could be of potential interest to scientists from both domains.

Bioinformatics is a large and ever-growing area of academic and industrial research. As such, it would clearly not be feasible to discuss all of the areas in which soft computing can, and have, been applied. Instead what is discussed in this chapter is an overview of some of the areas of importance and great interest in bioinformatics within the fields of genomics and proteomics. Within genomics, soft computing applications in *gene expression classification* and *time series modelling* are described. Within proteomics, *clinical proteomics* and *protein identification* are discussed. All of these areas have seen activity in the application of soft computing techniques to improve the process of performing, and analysing the data from, experiments.

2 Soft Computing in Genomics

Genomics is defined as the study of the full complement of genes in a given organism, including their functions. Mutations, deletions and alterations that are directly or indirectly altered in gene expression can be uncovered from genome analysis which has played a major role in areas such as clinical cancer research (Chung *et al* 2007). Advances in technology have played a large role in the success of genomics, providing scientists with the capability to analyse thousands of gene expressions within a single experiment. Arguably the most important technique in this area is that of DNA microarrays which are able to perform experiments analysing thousands of gene expressions. The increasing ease and number of genes which can be analysed with these experiments has resulted in large volumes of gene expression data being collected. This data is of great interest to the bioinformatics community since it potentially holds interesting and novel knowledge, but requires appropriate methods to perform an analysis. Elucidating the patterns hidden in gene expression data offers a tremendous opportunity for enhancing understanding of functional genomics (Jiang *et al* 2004).

The issue of genomic data analysis is not always as straightforward, however, as conceiving ever more efficient computational or statistical algorithms. Other considerations include robustly handling noise and experimental variation in the data, producing hybrid models to marry multiple techniques and the incorporation of an expert's hypothesis about the data in question (e.g. null hypothesis testing).

Although no de facto standard has been developed to date, the question of how to effectively analyse this genomic data is a very active area of research. From the field of soft computing, there are numerous examples of approaches to the interpretation and analysis of genomic data using computational techniques. In particular, soft computing techniques such as neural networks and fuzzy systems are of interest because of their demonstrable strength in handling noise and imprecise information. Some of the domain applications in genomics are considered next.

2.1 Gene Expression Classification

Another useful application of soft computing techniques is in the area of classifying gene expression data into categories of interest. Techniques have been applied to automatically assign such data into classes of interest, reducing the time to perform analysis whilst maintaining a high level of confidence, all of which are of great value to the genetics industry. This includes examples such as functional interest, potential candidate drug targets and disease-linked genes.

Work by Ritchie *et al* (2007), investigated the use of a genetic programming neural network (GPNN) to identify genes that influence the risk of common, complex diseases through their interaction with other genes and the environment. The approach uses the evolutionary features of genetic programming to optimise and evolve the architecture of the neural network. The genetic programming was constrained using heuristics from Koza and Rice (1991) to ensure that the neural network maintained a recognisable neural network structure. Figure 2 illustrates an example of a network generated using GPNN.

The network is created using a ten-fold cross-validation training phase, creating binary expression tree representations of a neural network (such as in Figure 2). New solutions are then created using a mixture of retaining some of the best solutions from the training phase and using some for crossover for. Good solutions are evaluated using a fitness measure based on the classification error. The generation of new solutions is halted when a threshold is reached, such as a classification error or epoch number.

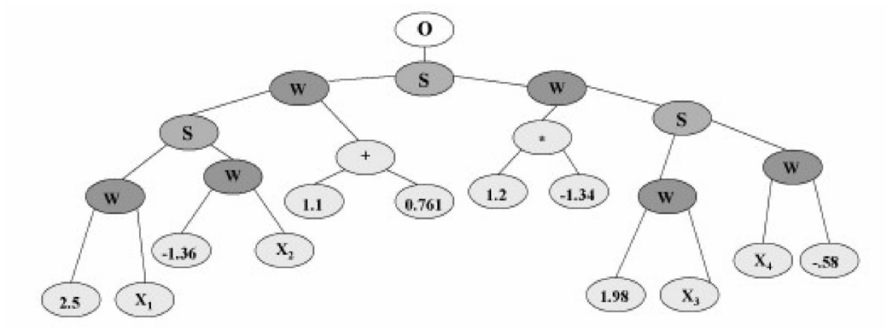


Fig. 2. A network evolved using GPNN. The O is the output node, S indicates the activation function, W indicates a weight, and X1-X4 are the NN inputs. Taken from Ritchie *et al* (2007).

An evaluation of this approach was conducted using data from a variety of epistasis models for which functional gene variation were single-nucleotide polymorphisms (SNPs). The models are known to contain interaction effects, without any main effects, in order to test the 'worst case' scenario in which there are minimal main effects.

An analysis of the results of this process reveals that the GPNN was, in almost all cases, a more powerful technique for accurately detecting gene-gene interactions, although with some lower accuracy results for three locus models. Furthermore, in a comparison with Classification and Regression Trees (CART) the GPNN also produced a lower number of false positives as compared with the CART. The disadvantage of using the GPNN is considered to be the time to learn the models, which is significantly higher than many other techniques. The obvious trade-off in this study is accuracy versus computational effort. Ergo, the more computational intense technique is more accurate in identifying gene-gene interaction, the less computationally expensive techniques are less success at identifying such interactions.

Another area in which soft computing techniques have been applied concerns the classification of cancers into specific diagnostic classes based on their gene expression signatures. Tumours are usually diagnosed and placed into an accepted category, based on by histology and immunohistochemistry based on their morphology and protein expression. This diagnostic process is a crucial step since treatment may vary dependant upon the type of tumour. However, this can be a complex task for a number of reasons. There is a wide spectrum in cancer morphology and many tumours are atypical or lack morphologic features that are useful for differential diagnosis and, additionally, clinical information can be incomplete or missing (Ramaswamy *et al*, 2001).

Khan *et al* (2001) looked at this problem in the context of small, round, blue cell tumours (SRBCTs) of childhood. In their approach, the authors used a neural network with no hidden layers, which is only capable of linear discrimination, to classify the tumour data into four categories of SRBCT. Specifically, these were Ewing family (EWS), rhabdomyosarcoma (RMS), Burkitt lymphomas (BL) and neuroblastoma (NB). Here, the authors applied neural networks to analyse gene-expression signatures of each of the SRBCTs in order to classify them into one of each of the categories, above. The methodology is illustrated in Figure 3.

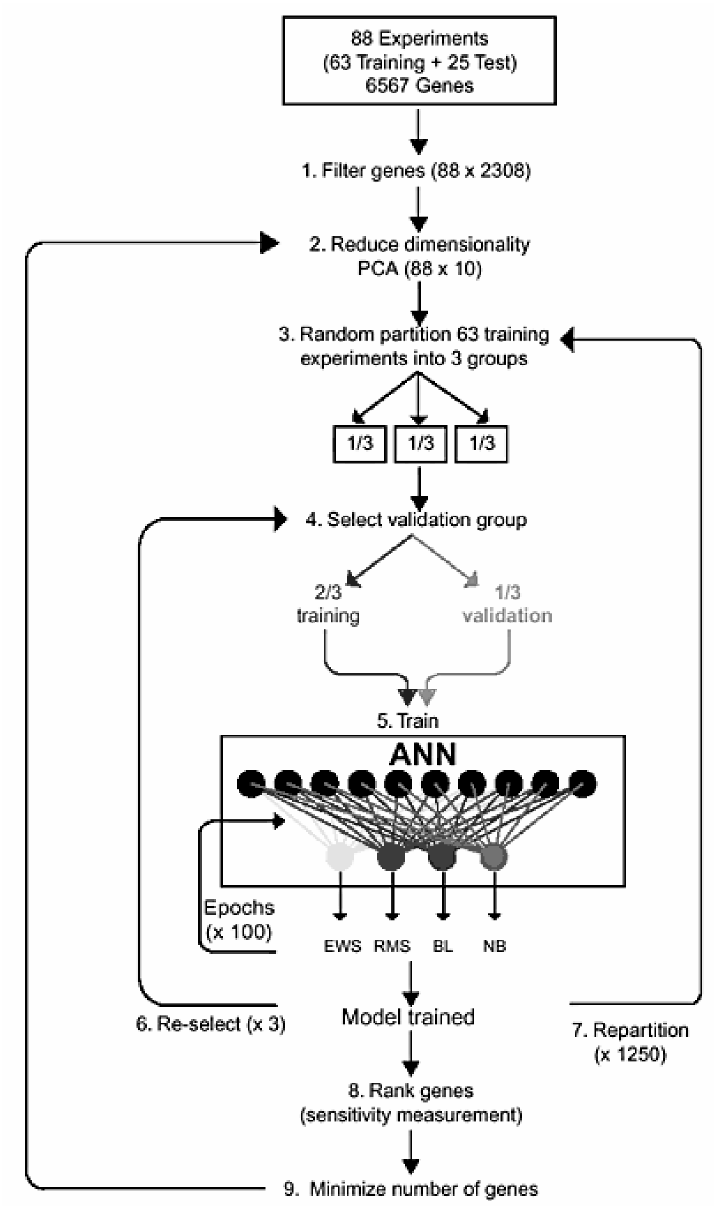


Fig. 3. An illustration of the methodology used to classify SRBCT gene expression data. Taken from Khan *et al* (2001).

The neural network models were trained using gene-expression data from cDNA microarrays containing 6567 genes from a total of 63 training samples. In order to reduce the dimensionality a filtering of very low level expression was performed followed by PCA. Next, a three-fold cross-validation was used with the training data to create a total of 3,750 neural network models. A committee vote (average of all neural network outputs) was used to determine the class of the instance.

Using this method the approach was able to classify instances into their appropriate category with 100% accuracy – including 20% of the set aside instance which were used as unseen test examples. A further validation was performed to ensure that the rejection of classification of a sample into one of these groups could occur if it does not belong. To ensure this, an empirical probability distribution for the distance of samples was calculated. Using these distributions, samples were only classified as a particular category of cancer if they lay within the 95th percentile.

Such an approach would seem useful in achieving high accuracies, even given the imbalance of input features (gene expression) to samples. However, there could be a potentially large overhead in using neural networks in this manner. In particular, the creation of this number of neural networks as in this example (3750 neural network models) can be a time consuming effort, although time to train is only shown in epochs in the Khan *et al* (2001) example. Furthermore, consideration should be given to the type of neural network employed, i.e. a simple input and output layer structure. Such a linear classification mechanism would not appear to use one of the strengths of this type of system, specifically, the ability to classify nonlinear relationships.

Another approach to classifying gene expression data is by using systems that can incorporate explicit expert knowledge into the learning cycle. One such method is described by Neagu and Palade (2003) in which they use a neuro-fuzzy approach to classifying genes by function. Fuzzy techniques offer the advantage of robustness with noise in data, although this is not unique to only fuzzy systems; this is also true of many neural network approaches. They also offer the advantage of handling, to some degree, uncertainty and/or vagueness in data (see Gan *et al* (2005) for a more comprehensive discussion on this topic). The explicit expert knowledge is incorporated into the system in the form of rules proposed by human experts (whilst implicit knowledge is considered encapsulated in the form of trained neural networks). In one of the approaches discussed, the implicit and explicit knowledge is integrated into a single network by giving each the behaviour of fuzzy modules which give the general output of the system using a *T-conorm* of fuzzy outputs of each module. The system is seen as equivalent to a set of fuzzy rules; the resulting final output is determined using a firing rules first method.

There are several advantages in applying systems that incorporate fuzzy logic which render it particular suitable for analysing bioinformatics data. Besides coping well with noise in the data, as previously discussed, the system described by Neagu and Palade (2003) allows the integration, in a natural way, of an expert's knowledge (in this instance, three genomics experts) into the process of data interpretation and analysis. Finally, and of particular relevance to those working in bioinformatics data analysis, the approach results in a set of rules that can be extracted following the training phases. These rules offer insight into the classification process, helping to characterise how the input is correlated to the output. For instance:

NFN7H:
IF Norm(log(GlcLBAvgVol)) is:Med
THEN Fuzzy-FunctionalGroup is:FG5 (75.18%)
IF Norm(log(GlcLBAvgVol)) is:High
THEN Fuzzy-FunctionalGroup is:FG1 (100.00%)

These rules were extracted from an architecture using seven hidden neurons (NFN7H). The first rule describe a correlation between Fuzzy-FunctionalGroup FG5 and a medium value of the normalised log of the GlcLBAvgVol input, with a confidence value of ~75% for this rule. Similarly, the second rule shows a correlation when this same input is high, this time with the FG1 class and with a stronger, 100% confidence level. Naturally, if such information should be assessed by experts in the area to be correct, they in turn could be used as explicit knowledge in rules added to further iterative models.

2.2 Gene Expression Time Series Modelling

DNA microarray experiments that perform an analysis over time series expression are an increasingly popular method for studying a wide range of biological systems. In static microarray experiments, a snapshot of the expression of the genes from samples is taken and analysed. Temporal gene expression experiments capture data from samples over time; in the simplest form this could be before and after some event, such as from a sample population that is exposed to a pathogen at some given interval. However, this could be a more complex monitoring over many time points, creating a much larger series.

As with the analysis of static gene expression data, time series modelling presents similar problems, specifically noise and experimental variation. Additionally, missing time points within a series for any given gene can also present difficulties when attempting interpolation (Bar-Joseph 2004). Further problems can arise from non-uniform sampling throughout a time series and the intrinsic variability within the actual underlying biological process. For instance, a biological process can take longer between subjects or between different species of organism.

An approach by Amato *et al* (2006) makes use of a multi-step approach to time series analysis and gene expression clustering. A feature extraction pre-processing step was used to reduce the dimensionality of the data. Since the features extraction process was based on a non-linear PCA neural network method which can estimate the eigenvectors from unevenly sampled data this was used to also compensate for missing values within the data set. Such methods can prove powerful when analysing unevenly sampled data.

The work of Ao and Ng (2006) considers the modelling problem of gene expression data captured at various points in a time series and also consider the advantages of using a pre-processing step. The authors use the expression values of the time points in each stage of a cell cycle as input vectors for a principal components analysis (PCA) algorithm and then use these processed vectors as inputs to a neural network. The use of PCA in this case is to reduce the dimensionality associated with temporal DNA microarray data. The overall architecture for performing the analysis is shown in Figure 4.

The proposed model in Figure 4 has the advantage of extracting features from the gene expressions time series as well as prediction accuracy. Following the initial step of applying PCA, the features can be seen, such as is illustrated in Figure 5. In this illustration, for instance, we can see the specific component values for gene expression

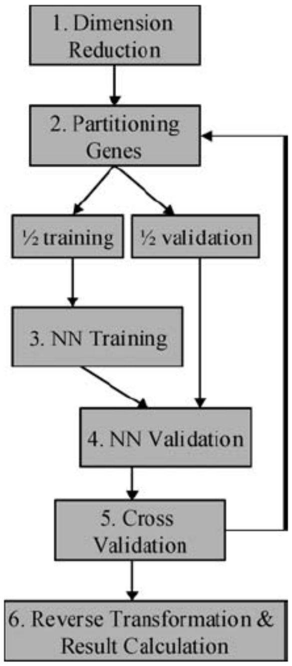


Fig. 4. A system architecture for analysing gene expression time series data. Taken from Ao and Ng (2006).

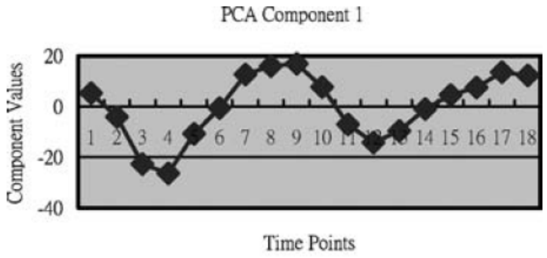


Fig. 5. A graph following PCA on gene expression data describing the first component of variation. Taken from Ao and Ng (2006).

at all of the given time points, giving some insight into the variance levels in the series.

The model in Ao and Ng (2006) provides some interpretation of the initial trends in the data which are used to classify the gene expression data. Such an explanation feature is of great value in bioinformatic domains where decision-making processes often require scrutiny, such as from governmental bodies. This approach is aimed at helping practitioners to gain a better understanding of a cell cycle, and to find the dependency of genes, which is useful for drug discoveries.

3 Soft Computing in Proteomics

Following the explosive growth of research into genomics, the study of the proteome has become a fundamental part of industrial and academic biochemical research. Proteomics is defined as the large-scale identification and characterisation of the proteins encoded in an organism's genome (Alberts *et al*, 2002). More specifically, proteomics is concerned with the analysis of the transcription, structure and function of proteins, protein-protein interactions and post-translational modifications to proteins which can in turn have an effect on structure, function and interaction networks. For our convenience, we will consider all work that is more generally concerned with any aspect of protein analysis, even on smaller-scales, without entering in to any debate about the boundaries of proteomics research.

There are several application areas which have benefited from research in proteomics. For instance, drug discovery through the identification of new drug targets and clinical diagnostics through the identification of protein disease markers (Vitzthum *et al*, 2005). The latter of these areas has led to the growth of what is commonly called 'clinical proteomics' which aims to identify disease-associated proteins and characterise the information flow, allowing for more accurate diagnostic testing and appropriate therapy tailoring. Furthermore, because proteins can alter between cells and with interactions with one another and the environment they are inclined to offer more insight into an organisms physiology than the genome, which gives a static snapshot of what is a largely constant body of information.

Considering the proteomic technologies that are currently used, there are a number of limitations and technical challenges that are still to be satisfactorily addressed. Kopec *et al* (2005) list a number of these, including the time required for data analysis and integration. Furthermore, the ability to generate data on a large scale from experimentation outstrips the ability to analyse it. For this reason, the analysis of data from proteomics has long been considered the Achilles heel of proteomics bioinformatics (Patterson 2003).

Soft computing has some relevance when considering solutions to address some of these issues. As with genomics, approaches that can automate data analysis to some degree can deal effectively with noise and can provide interpretable reasoning are of interest within proteomics. Some of these applications are described next.

3.1 Clinical Proteomics

The field of clinical proteomics has seen some of the most rapid growth within proteomic studies. The aim of disease diagnosis and/or tailoring treatment is not a new one; indeed, this was one of the hoped outcomes of genomics (for instance, pharmacogenomics). However, the dynamic nature of proteins enables a more detailed picture to be gleaned which offers insight into, for instance, pathogen presence and progression and information on networks that interact with organ and circulatory systems.

Clearly, analysis of this sort is a complex business, both in terms of the dimensionality of the data as well as the expert decision making. And, as we have seen with other bioinformatic areas, noise and uncertainty can often play a part in the quality of the data, and hence the decision making process.

Considering clinical proteomics in cancer diagnostics/prognostics, Ball *et al* (2002) apply the use of a multi-layer perceptron (MLP) neural network for the identification of valid biomarkers. In their work, the authors analyse data derived from Matrix-assisted laser desorption/ionization (MALDI) mass spectrometry (MS) experiments. MALDI-MS is a technique used to measure the mass-to-charge ratio of ions and is often used to find the composition of a physical sample. In essence, the theory can be simplified to three steps; (i) ionising the sample molecules, (ii) separation of the molecules in the analyser by their mass-to-charge (m/z) ratio and, (iii) detection and measurement of the separated ion particles where m/z ratios are stored as data. In the work, the data produced from these experiments is analysed for 'phenomic fingerprints' that is, patterns in the data that are demonstrable of specific medical conditions (in this context cancer).

Issues which arise when considering this type of data include the intrinsic levels of background noise, the high dimensionality of the input space and the sometimes complex interactions. MLP neural networks are well suited to analysing this type of data. However, one of the criticisms often levelled at neural networks is their apparent lack of transparency when attempting to understand the underlying reasoning process. As we have seen earlier, such issues are of particular relevance to bioinformatics data, where the decision making process is arguably as important as the accuracy of the final model.

One method developed to overcome this obstacle is the identification of important inputs which have a strong influence on the neural network model. Ball *et al* (2002) and later Lancashire *et al* (2005) use this approach to interpret neural network models following training on mass spectrometry proteomics data. By building multiple models using a sliding window approach (i.e. taking a group of inputs and moving along to the next subset and building a new model) and subsequently taking a summation of the absolute weights from input to output, a 'relative importance value' was derived. From this, the highest x values can be selected to construct a final neural network model, one which will be used for identification of the proteomics biomarkers for clinical diagnostics/prognostics. The process of extracting important weights can again be repeated one last time for this final model which would be used to interpret the important inputs (in this instance m/z values from the sample) and hence the decision making process of this solution.

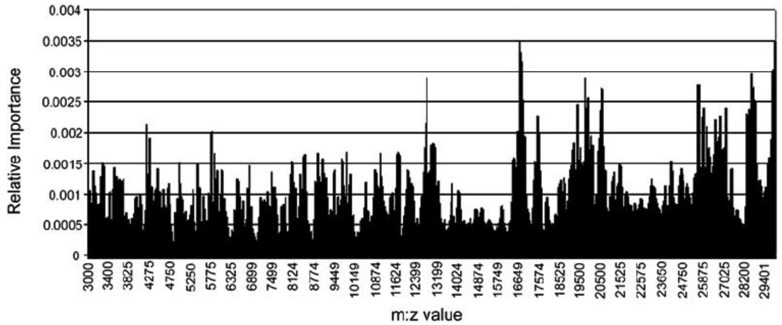


Fig. 6. Results of using the weight interpretation of neural networks for cancer diagnostics, m/z values can be seen on the x axis, with relative importance on the y axis. From Lancashire *et al* (2005).

An example of the results from this process is shown in Figure 6. From this graph we can clearly see those inputs m/z values which correspond to inputs which have an important role in the diagnostic classifier.

As part of the CLINPROT system for clinical diagnostics (which, in this context is considered as data pre-treatment, elementary visualization, statistics, pattern determination, evaluation and classification), Schleif *et al* (2006) investigated the use of a supervised fuzzy classification method for identifying biomarkers. The approach uses a self-organising map (SOM) for data mining and visualising of proteomic data which classifies and interprets the data into useful using graphically labelled categories.

Experimentation was conducted using 60 plasma samples from 30 acute lymphatic leukaemia (ALL) patients and 30 controls with the aim of successfully classifying samples into these categories. The data was then fed into a fuzzy labelled (FL) SOM, following some initial pre-processing (see Schleif *et al* (2006) for more details). The FL-SOM has topologically preserving properties, which ensures that the input space topology is retained in the output space. This is an advantageous property for visualisation, since the output 'map' organises the space so that closely related instances are spatially close to one another. In this sense, the regions can be seen to be groups or clusters in that there is a spatial relationship. This property is drawn on in the FL-SOM model with the integration of the label information such that the prototype locations are determined with respect to the auxiliary data. This information is relevant for the interpretation since the classification can be visualised by means of the underlying SOM topology adapted towards the labelling. The results of this experimentation can be shown by the visualisation in Figure 7, which illustrates clearly that prototype label locations lie well within correctly corresponding classes.

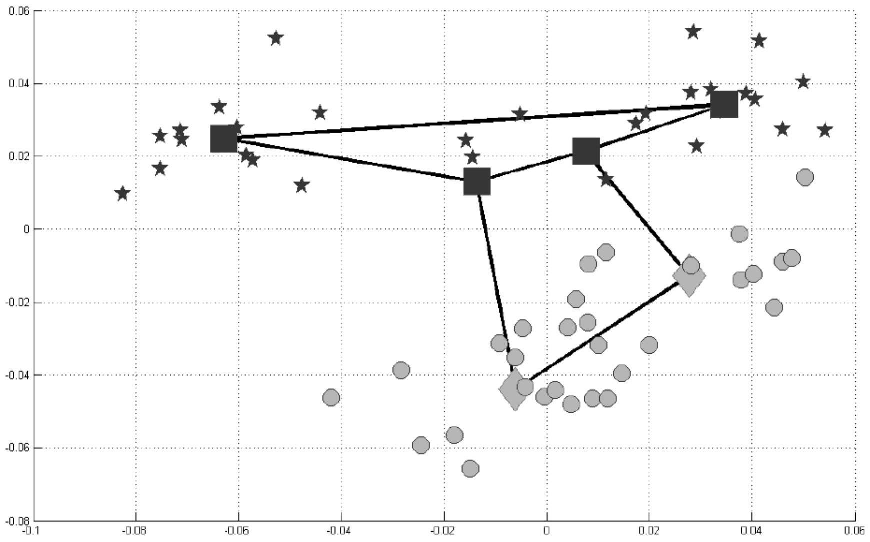


Fig. 7. Visualisation of control instances "o" cancer instances "*" with prototype locations for labelling shown for control "□" and cancer "♦" Schleif *et al* (2006)

The end result of the FL-SOM is a robust classifier where efficient learning with fuzzy labelled and high dimensional data is possible. Furthermore, the integration of labelling into the location of prototypes in the SOM allows a visualization of those parts of the data relevant for the classification and hence interpretation of the reasoning process.

Clinical proteomics is of great interest to those working in proteomics and especially those aiming to improve the so called “benchside to bedside proteomics”. Future development of clinical proteomic tools will have important translational applications for early detection, as a supplement to existing and co-evolving technological advances in diagnostic imaging, and will also provide opportunities for providing real patient tailored therapy (Petricoin *et al*, 2004). The incorporation of a wide range of techniques, including some of those from soft computing discussed here, will potentially have some role to play in the progression of this area.

3.2 Protein Identification

The identification of proteins concerns both the biological experimentation of extracting proteins from a sample and the later analysis of the data produced from these experiments. Since our focus is with soft computing, we consider only the latter of these two aspects in this section. Specifically, we are concerned with those approaches that analyse data to identify proteins of interest, given a particular bioinformatic context. This is different, therefore, from clinical proteomics in that was are concerned with identifying proteins that may be of biological significance within a particular experiment, rather than identifying proteins with the aim of some patient diagnostic or prognostic significance.

Early work by Yen *et al* (1996) applied soft computing techniques including fuzzy rule-based modelling to identify components within a metabolic pathway. More recently, Malone *et al* (2006) used pre-processing methods including PCA, as transformed input data into a neural network (cf. genomics work described previously by Khan *et al* (2001)). In this work, the authors created a hybrid approach to identifying proteins that corresponded to expert heuristics, such that these proteins would be considered ‘interesting’ and hence worthy of further analysis. This consisted of two key stages, (i) the optimisation of expert heuristic rules and, (ii) the automated trend analysis of two-dimensional gel electrophoresis (2-DE) proteomic data to identify proteins corresponding to the heuristics.

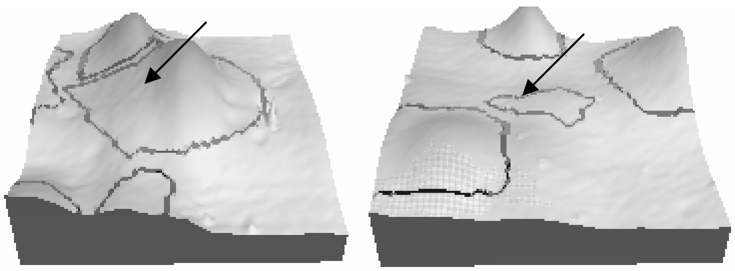


Fig. 8. Visualisations of the same protein spot expressed using two-dimensional electrophoresis from two different tissues samples. The protein spot on the left (identified by arrow) in control sample and, on right, in diseased sample.

2-DE is used to separate proteins form a sample in two dimensions, by isoelectric point and mass. In this way they are uniquely expressed within a two-dimensional space and can be observed, such as across a series of experiments for variations. It is these variations that are often of interest to experts analysing the data and which experts often use intrinsic heuristics. In the first stage of the Malone *et al* (2006) approach, following knowledge acquisition sessions with experts, heuristic knowledge was gleaned and represented in the form of fuzzy rules. Such rules have the advantage of being able to represent intuitive terms in a form close to natural language and have no precise thresholds, reducing brittleness (Malone *et al*, 2004). In order to optimise these expert heuristics, an Adaptive Nero-Fuzzy Inference System (ANFIS) was used. As earlier described in Neagu and Palade (2003), the rules can be incorporated into the system which is then trained on examples to learn functions for discriminating between the interesting and un-interesting proteins. Following this training (and testing), 14 rules were extracted from the network which help to describe the reasoning process.

Following the optimisation stage, the proteins corresponding to each of the 14 'interesting' rules were identified from the expert's analyses, which had been previously conducted, and training and testing datasets were created. These data sets were used to train and test the automated tend analysis architecture which consisted of a variance analysing data mining technique to pre-process the data and, subsequently, a neural network to classify the data into categories of interest (14 in total, corresponding to the 14 rules). Figure 8 gives an illustration of the types of trends that such a system hopes to automatically identify. In this instance, the same protein spot, which has been expressed using 2-DE, is shown in a control sample on left and, on right, has decreased in size and has also moved position in the diseased sample.

This approach has two primary advantages. Firstly, trends are identified, i.e. data analysed, in a semi-automated fashion and, secondly, that these trends correspond to those experts have flagged as interesting. These are important considerations since the data analysis aspect of proteomics is often seen as a bottleneck to the field.

Another instance of protein identification using soft computing methods is described by Junker *et al* (2004). In their work, they focus on identifying proteins that are linked to malignant tumours, specifically (RCC), the most common form of kidney cancer. Following the extraction of proteins using mass spectrometry (see earlier description) a fuzzy clustering and rule extraction was performed. A simplified two cluster approach was used for segregating high and low abundance proteins. This was followed by rule extraction to find features (peaks of expression) that have the most similar membership distribution compared to the given sample class distribution. The rules extracted from this process, following testing, were able to classify to 97.9% accuracy, and with the obvious advantage of transparent reasoning (i.e. the rules).

The identification of proteins of interest is currently, and for the foreseeable future will remain, an important issue within proteomics. This is in part due to the increasing efficacy with which experimental techniques can produce proteomic data, creating the "analysis bottleneck" which today exists.

4 Discussion

As ever more innovate and efficient techniques for performing biological experiments are conceived, similarly, the demand for innovative and efficient bioinformatic techniques for analysing the data produced increases. This propagation of data leads to some fundamental challenges within bioinformatics. The forecast increase in computing power through Moore's Law, even if sustained, is outpaced by the growing amount of raw data produced in proteomics and genomics experiments. This necessitates innovative solutions and approaches in fields such as soft computing are arousing interest within the bioinformatics community.

Considering the area of soft computing, there are a growing number of examples where techniques that can handle uncertainty, noise and imprecision have distinct advantages. Bioinformatic data from areas such as genomics and proteomics often contain intrinsic experimental variation, such as from noise. This would therefore seem a potentially fruitful partnership worthy of interest.

Another aspect to some of the soft computing research described in this review is the transparency of decision making. Accountability is a hugely important factor when analysing and interpreting data from biological experiments – especially those that may have a real effect on human patients, such as in clinical proteomics. Such aspects of data analysis are sometimes overlooked in favour of efficient or accurate models. Rattray *et al* (2006) stated that popular analysis tools could also be adapted to include information about measurement uncertainties using approaches such as those in soft computing. However, this should always be considered within a more holistic view of bioinformatic data analysis. That is, the data, the process of collection, the methods used to analyse and interpret and the biological impact.

Finally, as with most data analysis and machine learning problems, the quality of the final model and results depends a great deal on the quality of the data. When considering bioinformatics data, this is of special relevance since the models are built using observations usually from biological experiments. Applying soft computing techniques can assist with the analysis, interpretation and knowledge discovery from this data, but this should always be considered with reference to the actual biological domain. Ultimately soft computing is a potentially powerful aid towards solving some bioinformatic problems and is not a panacea and, hence, should be applied appropriately.

References

Aitken, S.: Formalising concepts of species, sex and developmental stage in anatomical ontologies. Bioinformatics 21(11), 2773–2779 (2005)

Alberts, B., et al.: Molecular biology of the cell. Garland Science, New York (2002)

Amato, R., et al.: A multi-step approach to time series analysis and gene expression clustering. Bioinformatics 22(5), 589–596 (2006)

Ao, S.I., Ng, M.K.: Gene expression time series modeling with principal component and neural network. Soft Computing 10(4), 351–358 (2006)

Ball, G., et al.: An integrated approach utilizing artificial neural networks and SELDI mass spectrometry for the classification of human tumours and rapid identification of potential biomarkers. Bioinformatics 18, 395–404 (2002)

Bar-Joseph, Z.: Analyzing time series gene expression data. Bioinformatics 20(16), 2493–2503 (2004)

Chunga, C.H., et al.: Genomics and proteomics: Emerging technologies in clinical cancer research. Critical Reviews in Oncology/Hematology 61(1), 1–25 (2007)

Craighead, H.: Future lab-on-a-chip technologies for interrogating individual molecules. Nature 442, 387–393 (2006)

Datta, A., et al.: Control Approaches for Probabilistic Gene Regulatory Networks. IEEE Signal Processing Magazine 24(1), 54–63 (2007)

Gan, M.T., Hanmandlu, M., Tan, A.H.: From a Gaussian mixture model to additive fuzzy systems. IEEE Tran. On Fuzzy Systems 13(3), 303–316 (2005)

Jiang, D., Tang, C., Zhang, A.: Cluster analysis for gene expression data. IEEE Trans on Knowledge and Data Engineering 16(11), 1370–1386 (2004)

Junker, K., et al.: Identification of protein pattern in kidney cancer using ProteinChip arrays and bioinformatics. Journal of Molecular Medicine 15, 285–290 (2005)

Khan, J., et al.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine 7(6), 673–679 (2001)

Kopec, K.K., Bozyczko-Coyne, D., Williams, M.: Target identification and validation in drug discovery: the role of proteomics. Biochemical Pharmacology 69(8), 1133–1139 (2005)

Koza, J.R., Rice, J.P.: Genetic generation of both the weights and architecture for a neural network 2, 397–404 (1991)

Lancashire, L.J., et al.: Current developments in the analysis of proteomic data: artificial neural network data mining techniques for the identification of proteomic biomarkers related to breast cancer. Current Proteomics 3(4), 15–29 (2005)

Li, J., Wong, L., Yang, Q.: Data mining in bioinformatics. IEEE Intelligent Systems 20(6), 16–18 (2004)

McGarry, K., et al.: Integration of hybrid bio-ontologies using bayesian networks for knowledge discovery. In: International Joint Conference on Artificial Intelligence (IJCAI 2007), Hydrabad, India, January 6-12, 2007 (2007)

Malone, J., McGarry, K., Bowerman, C.: Automated trend analysis of proteomics data using an intelligent data mining architecture. Expert Systems with Applications 30(1), 24–33 (2006)

Malone, J., McGarry, K., Bowerman, C.: Using an adaptive fuzzy logic system to optimise knowledge discovery in proteomics. In: 5th International Conf. on Recent Advances in Soft Computing (RASC), pp. 80–85 (2004)

Mitra, S., Pal, S.K., Mitra, P.: Data mining in soft computing framework: a survey. IEEE Trans. on Neural Networks 13(1), 3–14 (2002)

Neagu, D., Palade, V.: A neuro-fuzzy approach for functional genomics data interpretation and analysis. Neural Computing and Applications 12(3-4), 153–159 (2003)

Patterson, S.D.: Data analysis—the Achilles heel of proteomics. Nature Biotechnology 21, 221–222 (2003)

Petricoin, E., et al.: Clinical proteomics: revolutionizing disease detection and patient tailoring therapy. Journal of Proteome Research 3(2), 209–217 (2004)

Ramaswamy, S., et al.: Multiclass cancer diagnosis using tumor gene expression signatures. Proc. Natl. Acad. Sci. USA 98(26), 15149–15154 (2001)

Rattray, M., et al.: Propagating uncertainty in microarray data analysis. Briefings in Bioinformatics 7(1), 37–47 (2006)

Schleif, F.M., et al.: Analysis and Visualization of Proteomic Data by Fuzzy Labeled Self-Organizing Maps. In: Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems, pp. 919–924 (2006)

Vitzthum, F., et al.: Proteomics: from basic research to diagnostic application. A review of requirements & needs. Journal of Proteome Research 4(4), 1086–1097 (2005)

Whitesides, G.M.: The origins and the future of microfluidics. Nature 442, 368–373 (2006)

Yen, J., Lee, B., Liao, J.C.: A soft computing approach to the metabolic modelling. In: Fuzzy Information Processing Society 1996 Biennial Conference of the North American, pp. 343–347 (1996)

Zwir, I., Zaliz, R.R., Ruspini, E.H.: Automated biological sequence description by genetic multiobjective generalized clustering. Ann. N.Y. Academy of Sciences 980, 65–82 (2002)

Evolutionary Regression and Neural Imputations of Missing Values

Pawan Lingras[1], Ming Zhong[2], and Satish Sharma[3]

[1] Saint Mary's University, Halifax, Nova Scotia, Canada
pawan@cs.smu.ca
[2] University of New Brunswick, Fredericton, New Brunswick, Canada
ming@unb.ca
[3] University of Regina, Regina, Saskatchewan, Canada
Satish.Sharma@uregina.ca

1 Introduction

While the information age has made a large amount of data available for improved industrial process planning, occasional failures lead to missing data. The missing data may make it difficult to apply analytical models. Data imputation techniques help us fill the missing data with a reasonable prediction of what the missing values would have been.

The generic techniques available for short term forecasting can also be used for data imputations. Researchers have explored the possibilities of applying advanced techniques to fully utilize available information for more accurate predictions. Various data mining functionalities have been applied to summarize data, analyze missing value distribution, classify and mine patterns, and visualize data (Cabena et al. 1998; Han and Kamber 2001). Sophisticated models based on genetic algorithms (GAs), time delay neural networks (TDNN), and locally weighted regression (LWR) analysis have been used for predictions in various fields. GAs are adaptive algorithms for finding practical solutions among an enormous number of possibilities. GAs have been widely explored in machine learning, scientific modeling, optimization, and engineering (Mitchell 1999). TDNN is a variant of neural network, which is particularly useful for modeling time series. Neural networks are good at recognizing patterns, generalizing and predicting trends. They are fast, tolerant to imperfect data and do not need formulas and rules. Neural networks have been used to make predictions and understand data in investments, medicine, science, engineering, marketing, manufacturing, and management (Lawrence 1993). LWR is a nonparametric regression analysis, which uses local variables for forecasting. Recent studies (Gorinevsky and Connolly 1994; Schaal and Atkeson 1994) have shown that LWR achieved higher accuracy than other parametric approaches. LWR have been increasingly applied in control and prediction (Atkeson et al. 1997).

The techniques described above can be used for short term forecasting as well as data imputations. However, the modeling process for forecasting and imputations needs to be different for a number of reasons. The short term forecasting problem usually has a fixed prediction window. For example, prediction of electricity demand for next two hours, which can help in resource planning in a power distribution company. The failure of data recorders, on the other hand, can be of variable lengths. The data imputation process needs to be prepared for prediction of data for a variable time period. Another important difference between forecasting and imputations is the available data. The forecasting involves prediction of future values based on past values in real-time, whereas the data imputation models have access to the data from before and after the failure. The data imputation is usually a necessary step prior to the offline data mining analysis.

This chapter desribes how the techniques described above, namely, GAs, LWR, and TDNN can be used to develop sophisticated models for estimating missing data. The hourly volumes of traffic counts are used to test the effectiveness of these models. Traffic volume is one of the most useful traffic data. Highway agencies usually employ permanent traffic counters, seasonal traffic counters, and short-period traffic counters to collect traffic volume data. Permanent traffic counters are used to monitor the traffic for 24 hours per day, 365 days per year. Seasonal traffic counters and short-period traffic counters are used to monitor the traffic for a short period ranging from a few hours to a few days (Garber and Hoel 1988). Accurate estimation of missing hourly volumes is of importance for highway agencies to carry out traffic analysis at both operation and planning level.

2 Review of Literature

Many researchers have studied the problem of missing values (Beveridge 1992; Bole et al. 1990; Gupta and Lam 1996; Little and Rubin 1987; Singh and Harmancioglu 1996; Wright 1993; Rubin, 1987, 1996; Schafer, 1997, 1999; Scahfer and Olsen, 1998). Ratner (2003) reported that there were four popular imputation methods for updating missing values:

1. Deductive Imputation. Missing values are deduced with certainty, or with high probability, from other information on the case.
2. Hot-deck Imputation. For each imputation class, missing values are replaced with values from “closest matching” individuals.
3. Mean-value Imputation. The sample mean of each imputation class is used to fill in missing values.
4. Regression-based Imputation. Missing values are replaced by the predicted values from a regression analysis.

This study focuses on neural and regressive imputations of missing values in a traffic data time series.

2.1 Locally Weighted Regressive Analysis

Regression analysis consists of graphic and analytic methods for exploring relationships between one variable, referred to as a response variable or dependent variable, and one or more other variables, called predictor variables or independent variables. Regression analysis is distinguished from other types of statistical analyses in that the goal is to express the response variable as a function of the predictor variables. Once such an expression is obtained, the relationship can be utilized to predict values of the response variable, identify which variables most affect the response, or verify hypothesized causal models of the response (Mendenhall and Sincich 1995). The regression finds a linear combination of input variables to minimize the sum of square errors between observed and predicted values.

A variation of regression analysis called locally weighted regression (LWR) is used in this study. Locally weighted regression is a form of instance-based (or memory-based) algorithm for learning continuous mappings from real-valued input vectors to real-valued output vectors. In locally weighted regression, points are weighted by proximity to the current **x** in question using a kernel. Local methods assign a weight to each training observation that regulates its influence on the training process. The weight depends upon the location of the training point in the input variable space relative to that of the point to be predicted. Training observations closer to the prediction point generally receive higher weights (Friedman 1995). A regression is then computed using the weighted points. This is achieved by weighting each data point according to its distance to the query point: a point very close the query gets a weight of one, and a point far away gets a weight of zero.

Locally weighted regression has been increasingly used in control and prediction. Zografski and Durrani (1995) explored the use of locally weighted regression in robot control and modeling time series, and also compared it to neural networks and other methods. Gorinevsky and Connolly (1994) compared several different approximation schemes, such as neural nets, Kohonen maps, radial basis functions, to local polynomical fits on simulated robot inverse kinematics with added noise. They found that local polynomical fits were more accurate than all other methods. One recent study demonstrated that locally weighted regression was suitable for real-time control by constructing a locally-weighted-regression-based system that learned a difficult juggling task (Schaal and Atkeson 1994).

2.2 Time Delay Neural Network

The variant of neural network used in this study is called time delay neural network (TDNN) (Hecht-Nielsen 1990). Fig. 1 shows an example of a TDNN, which are particularly useful for time series analysis. The neurons in a given layer can receive delayed input from other neurons in the same layer. For example, the network in Fig. 1 receives a single input from the external environment. The remaining nodes in the input layer get their input from the neuron on the left delayed by one time interval. The input layer at any time will hold a part of the time series. Such delays can also be incorporated in other layers. Neurons process input and produce output. Each neuron takes in the output from many other neurons. Actual output from a neuron is calculated using a transfer function. In this study, a sigmoid transfer function is chosen because it produces a continuous value in the range [0,1].

It is necessary to train a neural network model on a set of examples called the training set so that it adapts to the system it is trying to simulate. Supervised learning is the most common form of adaptation. In supervised learning, the correct output for the output layer is known. Output neurons are told what the ideal response to input signals should be. In the training phase, the network constructs an internal representation

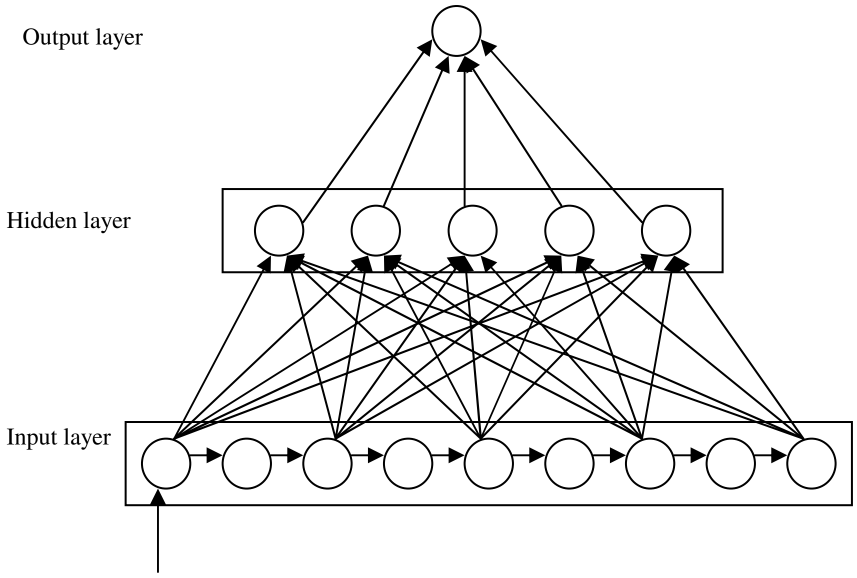


Fig. 1. Time Delay Neural Network (TDNN)

that captures the regularities of the data in a distributed and generalized way. The network attempts to adjust the weights of connections between neurons to produce the desired output. The backpropagation method is used to adjust the weights, in which errors from the output are fed back through the network, altering weights to prevent the repetition of the error.

2.3 Genetic Algorithms

Genetic algorithms are heuristic searching algorithms for finding near optimum solutions from vast possibilities. The origin of genetic algorithms (GAs) is attributed to Holland's work (Holland 1975) on cellular automata. There has been significant interest in GAs over the last two decades (Buckles and Petry 1994). The range of applications of GAs includes such diverse areas as: job shop scheduling, training neural networks, image feature extraction, and image feature identification. Previous research shows that GAs consistently outperform both classical gradient search techniques and various forms of random search on more difficult problems, such as optimizations involving discontinuous, noisy, high-dimensional, and multimodal objective functions (Grefenstette 1986).

The genetic algorithm is a model of machine learning, which derives its behavior from a metaphor of the processes of evolution in nature. In practice, the genetic model of computation can be implemented by having arrays of bits or characters to represent the chromosomes $(c_i \mid 1 \le i \le n)$, where c_i is called a *gene*. Simple bit manipulation operations allow the implementation of crossover, mutation and other operations. The crossover operator creates new individuals called *offspring*, by recombining the genetic material of two individuals, deemed the *parents*. Individuals with higher fitness scores are selected with greater probability to be parents and "pass on" their genes to the next generation. The mutation operator randomly alters one or more genes in an individual. Mutations add genetic diversity to the population.

The GAs attempt to construct a good individual by starting with a population of randomly generated individuals. From there on, the genetic operations, in concert with the fitness measure, operate to improve the population.

2.4 GAs for Designing Neural Network and Regression Models

Many researchers have used GAs to determine neural network architectures. Harp, et al. (1989) and Miller, et al. (1989) used GAs to determine the best connections among network units. Montana and Davis (1989) used GAs for training the neural networks. Chalmers (1991) developed learning rules for neural networks using GAs.

Hansen, et al. (1999) used GAs to design time delay neural networks (TDNN), which included the determination of important features such as the number of inputs, the number of hidden layers, and the number of hidden neurons in each hidden layer. Hansen, et al. (1999) applied their networks to model chemical process concentration, chemical process temperatures, and Wolfer sunspot numbers. Their results clearly showed advantages of using GAs configured TDNN over other techniques including conventional autoregressive integrated moving average (ARIMA) methodology as described by Box and Jenkins (1970).

Hansen et al.'s approach (1999) consisted of building neural networks based on the architectures indicated by the fittest chromosome. The objective of the evolution was to minimize the training error. Such an approach is computationally expensive. Another possibility that is used in this study is to choose the architecture of the input layer using genetic algorithms.

Time series modeling is based on the assumption that the historical values of a variable provide an indication of its value in the future. Based on the available information and pattern analysis, it is possible to make a reasonable assumption about the appropriate length of the history that may be useful for predicting the value of a variable. If all the historical values in the input layer were used in TDNN calculations, it would lead to an unwieldy network. Such a complex network may hinder the training process. A solution to such a problem is selective pruning of network connections. Lingras and Mountford (2001) proposed the maximization of correlation between input variables and the output variable as the objective for selecting the connections between input and hidden layers. Since such an optimization is not computationally feasible for large input layers, GAs were used to search for a near optimal solution. It should be noted here that since the input layer has a section of time series, it is not possible to eliminate intermediate input neurons. They are necessary to preserve their time delay connections. However, it is possible to eliminate their feedforward connections. Lingras and Mountford (2001) achieved superior performance using the GAs designed neural networks for the prediction of inter-city traffic.

Lingras et al. (2002) also showed that GAs designed regression models had more accurate short-term traffic predictions than simple regression models on a recreational road. For example, the average errors for simple regression models used by Kalyar (1998) are usually more than 4% in the training sets, whereas those for genetically

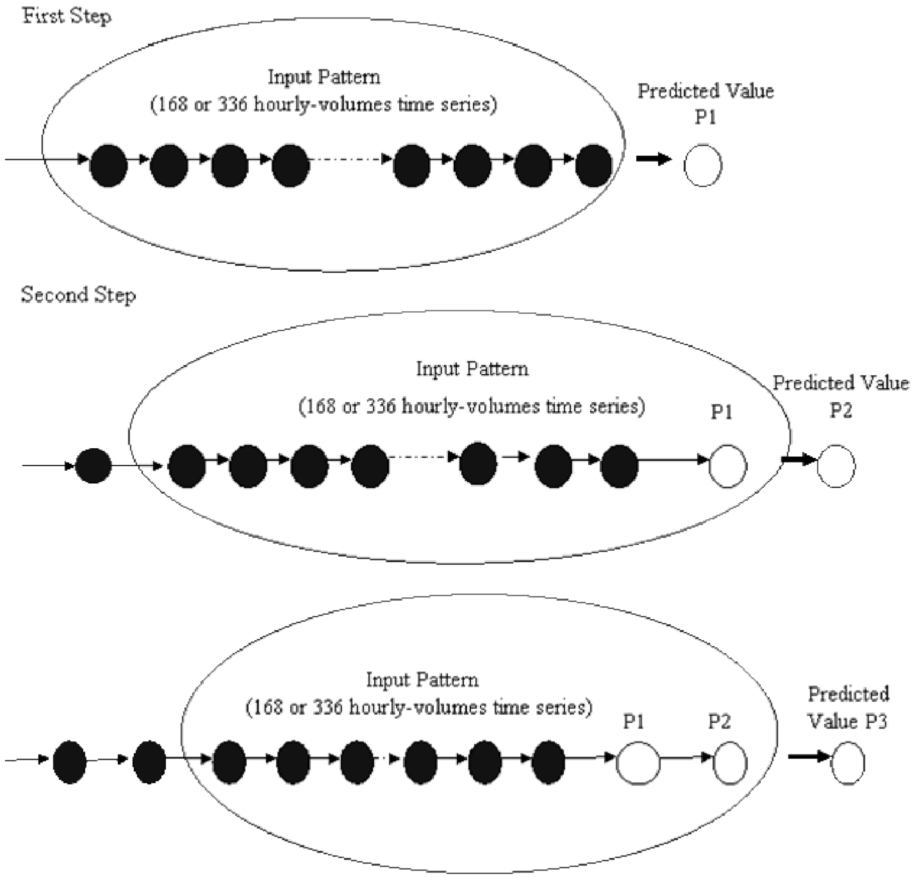

First Step
Input Pattern
(168 or 336 hourly-volumes time series)
Predicted Value
P1
Second Step
Input Pattern
(168 or 336 hourly-volumes time series)
P1
Predicted Value
P2
Input Pattern
(168 or 336 hourly-volumes time series)
P1
P2
Predicted
Value P3

Fig. 2. Prototype of genetically designed models

designed regression models are less than 0.6%. For test sets, the 95th percentile errors for normal regression models are usually more than 30%. The errors for some hours are as high as 54%. However, for genetically designed regression models, they are all lower than 8% and most of them are less than 4%.

The present study summarizes the extensive experimentation reported by Zhong (2003). It uses the same objective function for the development of regression and neural network models. These models are enhanced to estimate successive missing hourly volumes for 12 hours and up to a week (168 hours). The models are improved to be able to use the data from both before and after the failure period. The developed models are used to update missing values of traffic counts.

3 The Prototype of Genetically Designed Models

Two types of genetically designed models are developed in this study. The first type is short-term prediction models, which only use the data from before the failure as the input. For this type of models, a week-long ($7 \times 24 = 168$) hourly volumes before the first missing value are used as the candidate inputs. The second type is both-side models, which use the data from both before and after the failure as the input. For models using the data from before and after the failure, one week-long hourly volumes from each side of the occurrence of missing value(s) are used as the candidate inputs. Totally $168 \times 2 = 336$ hourly volumes are presented to GAs for selecting final inputs. The number of final input variables is decided to be 24 based on experiments with different numbers. For neural network models, there are 168 or 336 neurons in the input layer, but only 24 of them have the connections with hidden layer. In addition, there are 12 neurons in the hidden layer and 1 neuron in the output layer. For regression models, there are 168 or 336 independent variables. However, only 24 of them are selected by GAs and used to predict dependent variable.

Fig. 2 shows the prototype of genetically designed models used in this study. First, assuming there is only one missing value P_1 in traffic counts, candidate inputs of models are presented to GAs for selecting 24 final input variables. These 24 hourly volumes are chosen because they have the maximum correlation with the traffic volume of the missing hour, among all the combinations of 24 variables from candidate inputs. The GAs selected variables are used to train neural network and regression models for traffic prediction for the missing hour. The trained neural network or regression models are used to estimate the missing hourly volume P_1. If there were more than one successive missing value, same techniques would be used to predict missing value of the next hour P_2. However, at this stage, the candidate pattern presented to GAs for selecting final inputs includes estimated volume of the first hour P_1, as shown in Fig.2. P_1 may or may not be chosen as a final input because there are different input selection schemas for different hourly models. Fig. 3 shows a TDNN model with inputs selected from a week-long hourly-volume time series. Corresponding regression model also used same inputs for prediction.

A top-down design was used to search for the models with reasonable accuracy. First 24-hour universal models were established to test their ability, then they were further split into 12-hour universal models, single-hour models, seasonal single-hour models, day-hour models, and seasonal day-hour models. The characteristics of these models are as follows:

1. 24-hour universal models: This approach involved a single TDNN and a single regression model for all the hours across the year. The advantage of such models is the simplicity of implementation during the operational phase.
2. 12-hour universal models: In order to improve models' accuracy, 12-hour universal models were developed based on 12-hour (from 8:00 a.m. to 8:00 p.m.) observations. In other words, the observations from late evenings and early mornings were eliminated from the models.
3. Single-hour models: 12-hour universal models were further split into 12 single-hour models. That is, every hour had a separate model.
4. Seasonal single-hour models: Seasons have definite impact on the travel. Further dividing single-hour models into seasonal models may improve models' accuracy. In this study, yearly single-hour models were further split into May-August single-hour models and July-August single-hour models.

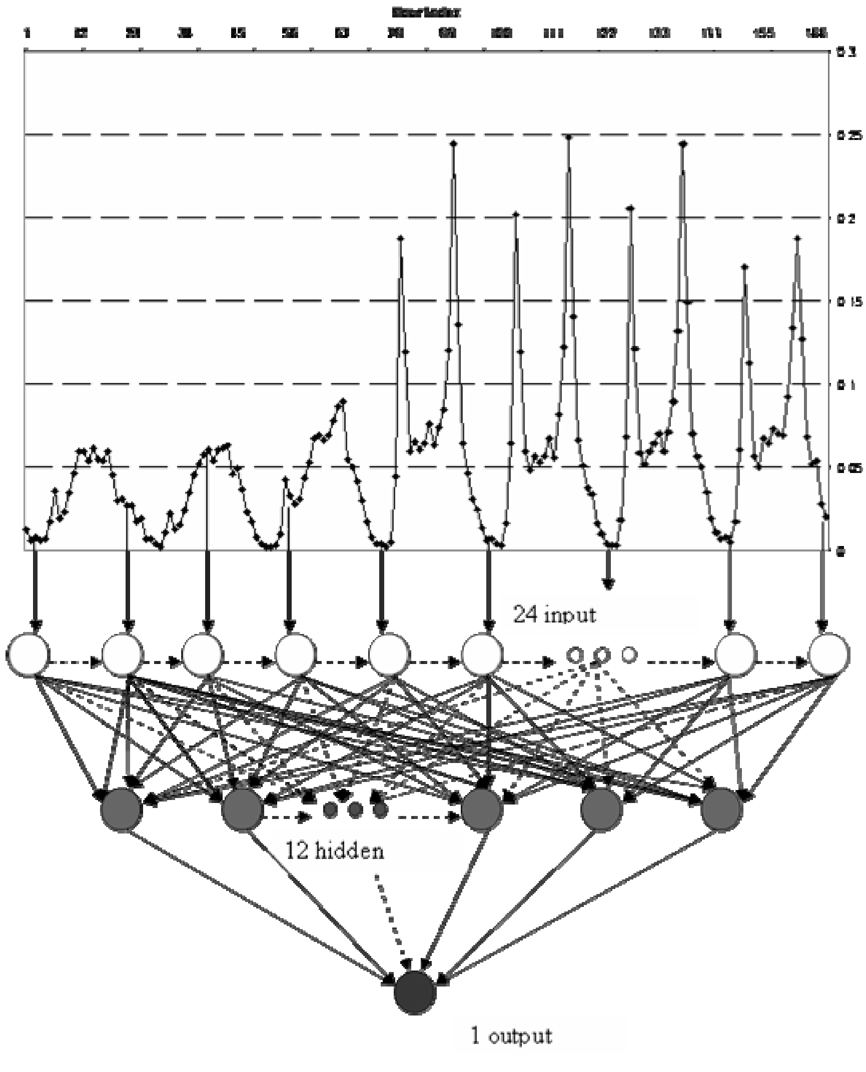


Fig. 3. TDNN model used for prediction

5. Day-hour models: Travel patterns also vary by the day of the week. Further classification of observations into groups for different days (e.g., Wednesday or Saturday) may improve models' accuracy.
6. Seasonal day-hour models. Day-hour models were further split into seasonal day-hour models (e.g., July-August day-hour models) to explore the models.

4 Experiments

Two large traffic volume databases are used to discover the nature of missing values. The missing values discussed in this study are restricted to missing hourly volumes.

The analyses show that these data are stored in various formats and have significant missing portions. The data are cleaned and integrated into a data warehouse. The 1998 data from Alberta are used to study the efficiency distribution of traffic counting program in detail. Cluster analysis is used to classify complete traffic counts into trip pattern groups. Sample counts from different pattern groups and functional classes are selected for testing imputation models proposed in this chapter.

4.1 The Nature of Missing Values

The data sets from two highway agencies in North America are used in this study. The data set from the Alberta Transportation spans 17 years, and the other from the Minnesota Department of Transportation has one-year data. The data are in the form of hourly volumes for one-way or both directions. Zhong (2003) provide detailed description of the format of the data. The analyses for missing values were applied to these data sets. Missing values are usually represented by zero hourly volumes or blanks in counter files. The machine failures result in blanks in the records. There are many reasons for this, including power surges, lightening, loss of battery power, solar panel failure, vandalism, and environmental fatigue, such as storms and frost heaves. Long-range zero hourly volumes indicate that the connections between sensors and counting machines were cut off, while the machines were still working normally. Lightening is a major reason for this kind of disconnection.

Analysis of missing data for Alberta indicated that large numbers of permanent traffic counts (PTCs) had missing values. Over seven years, more than half of total counts have missing values. For some years, the percentage is as high as 70% to 90%. Minnesota data also showed more than 40% counts having missing values. Detailed analyses of the missing data can be found in (Zhong 2003). The analysis clearly shows the extent of missing values and a need for data imputation.

The PTCs were clustered into various categories based on the predominant traffic on the corresponding highway sections. Six PTCs were selected from 4 out of 5 groups: two from the commuter group, two from the regional commuter group, one from the rural long-distance group, and one from the summer recreational group. Due to insufficient data, no counts were selected from the winter recreational group. These counts were used to test the accuracy of imputation models described earlier.

4.2 Analysis of Results

The performance of various models used in this study is expressed in terms of absolute percentage errors (APE). Previously, Sharma et al. (1999) used APE to evaluate the performance of neural networks and factor methods for estimating AADTs from short-period traffic counts. Lingras et al. (2002) used APE to assess the accuracy of genetically designed regression and neural network models for forecasting short-term traffic. The APE used in this chapter is defined as:

$$APE = \frac{|actual \quad volume - estimated \quad volume|}{actual \quad volume} \times 100 \qquad (1)$$

The key evaluation parameters consist of the average, 50th, 85th, and 95th percentile errors. These statistics give a reasonable idea of the error distributions by including (e.g., when calculating average errors) or excluding (e.g., when calculating the percentile errors) large errors caused by outliers or special events.

A detailed analysis of results for all the counters is described in Zhong (2003). Due to space limitations we only describe the summary of results and detailed reauslts for a regional commuter highway section in Alberta in this chapter.

It was found that the regression models tended to perform better than the TDNN models. Among the six categories of models described in the previous section, the refined models in the hierarchy tended to perform better. For example, the seasonal day hour models performed the best, while the 24 hour universal model performed the worst. The prediction accuracy for stable traffic patterns such as commuter traffic was higher than that for more variable recreational traffic patterns. As expected, models that were based on data from before and after the failure performed better than the models that only used the historical data. Table 1 shows an example of the prediction errors for updating 12 successive missing values with July-August day-hour models for regional commuter count C022161t. First column in the table shows the hour of the day. The remaining columns give various prediction errors. The column entitled "B" correspond to the data only from before the failure, while those entitled "B-A" correspond to the data from before and after the failure. It is clear that in this case the use of data from before and after the failure is advantageous. It is also interesting to note that the overall errors are very small. For example, the average errors for models based on data from before and after failure range from 0.8 to 2.1%. Even the 95th percentile errors are fairly low, ranging from 1.5% to 5.7%. That means less than 5% of the errors will be more than 6%.

Table 1. Errors for imputing 12 successive missing traffic volumes with GA-LWR

Hour	**Prediction Errors**							
	Average		**50th %**		**85th %**		**95th %**	
(1)	**B** (2)	**B-A** (3)	**B** (4)	**B-A** (5)	**B** (6)	**B-A** (7)	**B** (8)	**B-A** (9)
07-08	1.66	1.20	1.59	0.86	2.51	2.00	4.18	2.51
08-09	3.26	1.49	2.37	1.32	5.33	2.88	9.05	3.05
09-10	2.03	1.24	2.25	1.23	2.92	1.91	2.94	2.23
10-11	1.93	1.36	1.62	1.40	3.28	196	3.61	2.83
11-12	1.29	0.82	0.80	0.37	2.71	1.79	3.00	2.24
12-13	2.76	2.11	2.82	1.74	3.99	3.08	4.97	5.66
13-14	2.83	0.83	2.68	0.75	4.38	1.44	4.80	1.79
14-15	1.82	1.54	1.64	1.66	2.76	1.81	4.38	3.06
15-16	0.82	1.15	0.73	0.66	1.37	1.78	1.70	3.00
16-17	2.12	1.91	1.69	1.55	3.53	2.79	4.33	3.69
17-18	1.47	0.82	1.49	0.80	2.66	1.14	2.95	1.49
18-19	2.91	1.00	2.71	1.14	5.95	1.71	6.68	1.82

5 Summary and Conclusions

In order to make meaningful use of the available data, it may be necessary to provide a reasonably accurate estimation of missing data. Otherwise, it will be difficult to apply many of the data mining and knowledge discovery models. The analysis of traffic data (used in this study for illustration) shows that a significant number of data sets have some missing values.

While the techniques for short-term forecasting can be adapted for data imputation of missing values, there are inherent differences between the two problems. First of all, the duration for which the estimations are needed is variable in imputations, but fixed for forecasting. This fact makes the imputation problem a little more complicated. This chapter shows how imputed data values can be used for the estimation of the subsequent values. The second important difference between forecasting and imputation is the availability of data. The forecasting is limited by the fact that only historical data can be used for estimating future values. The accuracy of imputations, on the other hand, can be improved by the use of data from before and after failures.

The chapter describes how genetically designed regression and neural network models can be used for data imputation. The extensive study reported in detail in Zhong (2003) used a top down approach to improve generic universal models to more refined models specific to a particular time period. The results show that the genetically designed regression models outperform the genetically designed neural network models. The difference between the results is especially significant for stable temporal patterns from commuter highways as opposed to more variable recreational highways. It is possible that the neural networks may perform better for noisy and large data sets.

References

Allison, P.D.: Missing data. Sage Publications, Thousand Oaks (2001)

Atkeson, C., Moore, A., Schaal, S.: Locally Weighted Learning for Control. Artificial Intelligence Review 11, 75–113 (1997)

Beveridge, S.: Least Squares Estimation of Missing Values in Time Series. Communications in Statistics: Theory and Methods 21(12), 3479–3496 (1992)

Bole, V., Cepar, D., Radalj, Z.: Estimating Missing Values in Time Series. Methods of Operations Research 62 (1990)

Box, G., Jenkins, J.: Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco (1970)

Buckles, B.P., Petry, F.E.: Genetic Algorithms. IEEE Computer Press, Los Alamitos (1994)

Cabena, P., et al.: Discovering Data Mining: From Concept to Implementation. Prentice Hall, New Jersey (1998)

Chalmers, D.: The Evolution of Learning: An Experiment in Genetic Connectionism. In: Touretzky, D., et al. (eds.) Connectionist Models: Proceedings of the 1990 Summer School, pp. 81–90. Morgan Kaufmann, San Mateo (1991)

Friedman, J.H.: Intelligent Local Learning for Prediction in High Dimensions. In: International Conference on Artificial Neural Networks, Paris, France (1995)

Garber, N.J., Hoel, L.A.: Traffic and Highway Engineering. West Publishing Company, New York (1998)

Grefenstette, J.: Optimization of Control Parameters for Genetic Algorithms. IEEE Transactions on Systems, Man, and Cybernetics 16(1), 122–128 (1986)

Gorinevsky, D., Connolly, T.H.: Comparison of Some Neural Network and Scatter Data Approximations: The Inverse Manipulator Kinematics Example. Neural Computation 6, 521–542 (1994)

Gupta, A., Lam, M.S.: Estimating Missing Values using Neural Networks. Journal of the Operational Research Society 47(2), 229–239 (1996)

Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2001)

Harp, S., Samad, T., Guha, A.: Towards the Genetic Synthesis of Neural Networks. In: Shaffer, D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 360–369. Morgan Kaufmann, San Mateo (1989)

Hansen, J.V., McDonald, J.B., Nelson, R.D.: Time Series Prediction with Genetic Algorithm Designed Neural Networks: An Experimental Comparison with Modern Statistical Models. Computational Intelligence 15(3), 171–184 (1999)

Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

Kalyar, I.: Prediction of Sunday Afternoon Traffic Using Neural Network and Regression Analysis. Unpublished M.A.Sc. thesis. Faculty of Engineering, University of Regina, Regina, Saskatchewan, Canada (1998)

Lawrence, J.: Introduction to Neural Networks: Design, Theory, and Applications. California Scientific Software Press, Nevada City (1993)

Lingras, P.J., Mountford, P.: Time Delay Neural Networks Designed Using Genetic Algorithms for Short Term Inter-City Traffic Forecasting. In: Proceedings of the Fourteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Budapest, Hungary, pp. 292–299 (2001)

Lingras, P.J., Sharma, S.C., Zhong, M.: Prediction of Recreational Travel using Genetically Designed Regression and Time Delay Neural Network Models. Transportation Research Record 1805, pp. 16–24 (2002)

Little, R.J.A., Rubin, D.B.: Statistical analysis with missing data. John Wiley & Sons, New York (1987)

Mendenhall, W., Sincich, T.: Statistics for Engineering and Science. Prentice-Hall, NY (1995)

Miller, G., Todd, P., Hedge, S.: Designing Neural Networks using Genetic Algorithms. In: Shaffer, D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 53–60. Morgan Kaufmann, San Mateo (1989)

Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press, Boston (1999)

Montana, D., Davis, L.: Training Feedforward Networks using Genetic Algorithms. In: Sridhara, N. (ed.) Proceedings of Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan, pp. 762–767 (1989)

Pickles, A.: Missing data, problems and solutions. In: Kempf-Leonard, K. (ed.) Encyclopedia of social measurement, pp. 689–694. Elsevier, Amsterdam (2005)

Ratner, B.: CHAID as a Method for Filling in Missing Values. In: Statistical Modeling and Analysis for Database Marketing: Effective Techniques for Mining Big Data, Chapmall & Hall/CRC Press (2003)

Rubin, D.B.: Multiple imputation for nonresponse in surveys. John Wiley & Sons, New York (1987)

Rubin, D.B.: Multiple imputation after 18+ years. Journal of the American Statistical Association 91, 473–489 (1996)

Schaal, S., Atkeson, C.: Robot Juggling: An Implementation of Memory-based Learning. Control Systems 14, 57–71 (1994)

Schafer, J.L.: Analysis of incomplete multivariate data. Book No. 72, 72th edn. Chapman & Hall series Monographs on Statistics and Applied Probability. Chapman & Hall, London (1997)

Schafer, J.L.: Multiple imputation: A primer. Statistical Methods in Medical Research 8, 3–15 (1999)

Schafer, J.L., Olsen, M.K.: Multiple imputation for multivariate missing-data problems: A data analyst's perspective. Multivariate Behavioral Research 33, 545–571 (1998)

Singh, V.P., Harmancioglu, N.B.: Estimation of Missing Values with Use of Entropy. In: NATO Advanced Research Workshop, Izmir, Turkey (1996)

Wright, P.M.: Filling in the Blanks: Multiple Imputation for Replacing Missing Values in Survey Data. In: SAS 18th Annual Conference, New York (1993)

Zhong, M.: Data mining applications for updating missing values of traffic counts. Unpublished Ph.D. thesis. Faculty of Engineering, University of Regina, Regina, Saskatchewan, Canada (2003)

Zografski, Z., Durrani, T.: Comparing Predictions from Neural Networks and Memory-based Learning. In: Proceedings of ICANN '95/NEURONIMES '95: International Conference on Artificial Neural Networks, Paris, France, pp. 211–226 (1995)

Glowworm Swarm Optimization Algorithm for Hazard Sensing in Ubiquitous Environments Using Heterogeneous Agent Swarms

K.N. Krishnanand and D. Ghose

Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560012, India
krishna@aero.iisc.ernet.in, dghose@aero.iisc.ernet.in

1 Introduction

Ubiquitous computing based environments may be defined as human surroundings that are furnished with a network of intelligent computing devices, which could be either stationary or mobile (or an assortment of both), in order to service certain human-generated or needed tasks. Mark Weiser introduced ubiquitous computing, in its current form, in 1988 at the Computer Science Lab at Xerox PARC and wrote some of the earliest papers on ubiquitous computing (Weiser 1999). Ubiquitous computing based environments have several applications to the industry like environmental monitoring (Kim et al. 2007), ubiquitous factory environments (Jabbar et al. 2007), and self-sensing spaces (El-Zabadani et al. 2007). Kim et al. (Kim et al. 2007) develop a framework that uses ubiquitous sensor networks for atmospheric environment monitoring. Jabbar et al. (Jabbar et al. 2007) present methods that integrate latest technologies like RFID, PDA, and Wi-Fi in order to transform a nuclear power plant into an ubiquitous factory environment where effective data communication among local area operators and control room and minimization of work duration and errors in wake of safety requirements are achieved. El-Zabadani et al. (El-Zabadani et al. 2007) propose a novel approach to mapping and sensing smart spaces in which a mobile platform equipped with on-board RFID modules identifies and locates RFID-tags that are embedded in the carpet in the form of a grid.

An important application of particular interest is that of hazard sensing in an environment where the hazard can snowball into a major disaster. Several examples of such scenarios exist today. One such example could be a nuclear installation where the slightest leakage can signal a major potential disaster unless the leak is quickly controlled. Other similar scenarios in a factory environment are health-risk causing gas or inflammable gas leakage in factories, uncontrolled and unwanted vibrations or other mechanical phenomena, gradual failure of critical structures, etc. Even at homes, a large number of fires are caused by electrical short circuits leading to electrical discharges caused by malfunctioning wire joints.

A ubiquitous environment can be envisaged to have an assortment of a large group of virtually invisible nano-agents that can be used to detect potential hazard situations and, if possible, neutralize them. This group is composed of two types of agents: *(i) Stationary nano-agents* that are embedded into the environment and act only as fixed beacons at locations of their deployment and *(ii) Mobile nano-agents* that use the information cues received from stationary and/or mobile neighbors in order to detect, taxis towards, and gather at (thereby making themselves visible to an external sensor) potential hazard locations in order to prevent an impending disaster. Moreover, if there are multiple sources of hazard then this group of mobile nano-agents should be able to sense, automatically split into sub-groups, and simultaneously home on to these individual sources. In our earlier work (Krishnanand and Ghose 2007), we have considered a group of homogeneous mobile-agents for the same application. However, the advantages of using an assortment of stationary and mobile agents, instead of using a homogeneous mobile swarm, are two fold: Firstly, since the stationary beacons could be built at very low costs, they can be deployed in large numbers. This results in a need for using only a few number of mobile agents to accomplish a hazard sensing task that would otherwise require a relatively large number of mobile agents for its completion. Secondly, even though the mobile agents may be deployed at locations that are far away from the sources (e.g., corners of the region), as the stationary beacons pervade the entire environment and provide information about the signal profile at the locations of their deployment, the mobile agents use such information to find their way to a hazard source and, once in the vicinity of the source, the mobile agents exchange information with each other in order to home on to the source location.

In this paper, we will describe the modification to an algorithm called the glowworm swarm optimization (GSO) algorithm, which is based upon some basic bio-inspired swarm intelligence concepts, and demonstrate its efficacy in such hazardous situations. The basic GSO algorithm (Krishnanand and Ghose 2006) enables a homogeneous swarm of mobile agents to split into subgroups, exhibit simultaneous taxis towards, and rendezvous at multiple optima (not necessarily equal) of a multimodal function. The basic premise of the GSO algorithm is in the same spirit as the ant-colony optimization (ACO) algorithms, but with several significant differences (Bilchev and Parmee 1995), (Bonabeau et al. 1999). The agents are initially deployed randomly, according to a uniform distribution, in the objective function space. Similar to how agents in the ACO technique use pheromonal deposits to effect stigmergic communication among the members of the ant colony, the agents in the GSO algorithm carry a luminescence quantity called luciferin along with them for a similar purpose. Agents are thought of as glowworms that emit a light whose intensity of luminescence is proportional to the associated luciferin. Each glowworm uses the luciferin to (indirectly) communicate the function-profile information at its current location to the neighbors. A key feature of the GSO algorithm is the use of an adaptive local-decision domain, which is used effectively to detect the multiple optimum locations of the multimodal function. The glowworms have a finite sensor range which defines a hard limit on the local-decision domain used to compute their movements. Each glowworm selects a neighbor that has a luciferin value more than its own, using a probabilistic mechanism, and moves towards it. That is, they are attracted to neighbors that glow brighter. These movements, that are based only on information conditioned by the local-decision domain, enable the glowworms to split into subgroups and exhibit a simultaneous taxis-behavior towards optimum locations. A robotic implementation of sound source localization using the GSO algorithm is described in (Krisnanand et al. 2006). GSO has been used in our earlier work (Krishnanand and Ghose 2007) for ubiquitous environments that consist of homogeneous mobile agents. In this paper, we will modify GSO and implement it in a heterogeneous agent swarm composed of both mobile and stationary agents.

2 The Glowworm Swarm Optimization (GSO) Algorithm

The GSO algorithm is loosely based on Bilchev and Parmee's approach (Bilchev and Parmee 1995), which is a special variant of ACO to solve continuous optimization problems, but with several significant modifications. In the conventional ACO algorithms, used for combinatorial optimization, ants are randomly placed on the nodes of a generic graph. However, Bilchev and Parmee consider randomly placing a set of finite regions in the search space and an ant selects a region with a probability proportional to the amount of pheromone associated with the path between the nest and the region. In the glowworm algorithm, instead of finite regions randomly placed in the search space (as in (Bilchev and Parmee 1995)), we consider physical entities (agents) that are randomly distributed in the workspace. The agents in the glowworm algorithm carry a luminescence quantity called *luciferin* along with them, which corresponds to the *pheromone* associated with the path between the nest and each region in (Bilchev and Parmee 1995). Agents are thought of as glowworms that emit a light whose intensity is proportional to the associated luciferin and have a variable decision range r_d^i bounded by a circular sensor range r_s $(0 < r_d^i \le r_s^i)$. Each glowworm is attracted by the brighter glow of other neighboring glowworms. A glowworm *i* identifies another glowworm *j* as a neighbor when *j* is located within the current local-decision domain of *i*. Agents in the glowworm algorithm depend only on information available in the local-decision range to make their decisions. For instance, in Fig. 1(a), agent *i* is in the sensor range of (and is equidistant to) both *j* and *k*. But they have different decision-domains. Hence, only *j* uses the information of *i*.

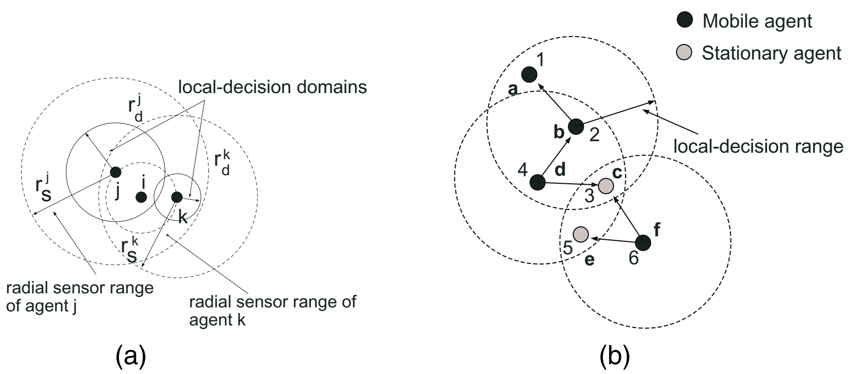


Fig. 1. a) $r_d^k < d_{ik} = d_{ij} < r_d^i < r_s^k < r_s^i$. Agent *i* is in the sensor range of (and is equidistant to) both *j* and *k*. But, it is contained within the decision-domain of *j* and outside the decision-domain of *k*. Hence, only *j* uses the information of *i* **b)** Emergence of a directed graph based on whether the agent is stationary or mobile, the relative luciferin level of each agent, and availability of only local information. Agents are ranked according to the increasing order of their luciferin values. For instance, the agent *a* whose luciferin value is highest is ranked '1'.

2.1 GSO Variant for a Heterogeneous Swarm of Agents

The GSO algorithm prescribes the placement of the stationary and mobile agents equipped with limited sensing and computing capabilities, randomly in the environment so that they are well dispersed. Let S_{agents} and M_{agents} represent the sets of all stationary and mobile agents, respectively. Let $S_a = |S_{agents}|$ and $M_a = |M_{agents}|$ be the number of stationary and mobile agents, respectively.We define the *assortment ratio* a_r as the ratio of number of mobile agents M_a to the total number of agents n:

$$a_r = \frac{M_a}{M_a + S_a} \tag{1}$$

Initially, all the agents (which represent the glowworms in the GSO algorithm) contain an equal quantity of luciferin ℓ_0. Each iteration consists of a luciferin-update phase followed by a movement-phase based on a transition rule.

Luciferin-update phase: The luciferin update depends on the function value at the agent position and so, even though all agents start with the same luciferin value during the initial iteration, these values change according to the function values at their initial positions. During the luciferin update phase, each agent adds, to its previous luciferin level, a luciferin quantity proportional to the measured value of the sensed profile (temperature, radiation level, etc.) at its current location, which changes with movements in the case of a mobile agent and remains fixed for a stationary agent. Also, a fraction of the luciferin value is subtracted to simulate the decay in luciferin with time. Accordingly, the luciferin update rule for a mobile agent $j \in M_{agents}$ is given by:

$$\ell_j(t+1) = (1-\rho)\ell_j(t) + \gamma J(x_j(t+1)) \tag{2}$$

where, ρ is the luciferin decay constant ($0 < \rho < 1$), γ is the luciferin enhancement constant, $x_j(t) \in \mathbb{R}^m$ is the location of agent j, at time t, in the m-dimensional real space $\mathbb{R}^m$, and $J(x_j(t))$ represents the value of the signal-profile at mobile agent j's location x_j at time t.

The luciferin update rule for a stationary agent $j \in S_{agents}$ is given by:

$$\ell_j(t+1) = (1-\rho)\ell_j(t) + \gamma J(x_j) \tag{3}$$

where, $J(x_j)$ represents the value of the signal-profile at the stationary agent j's location x_j, which remains fixed for all time t.

Movement-phase: A mobile agent does not require the information that which of its neighbors are mobile and which are stationary. That is, it does not distinguish between mobile and stationary neighbors. Accordingly, during the movement-phase, each mobile agent decides, using a probabilistic mechanism, to move towards a neighbor (stationary or mobile) that has a luciferin value more than its own. That is, in the context of glowworms, they are attracted to neighbors that glow brighter. Note that the stationary agents do not execute this phase. Fig. 1(b) shows the emergence of a directed graph among a set of six agents based on whether the agent is stationary or mobile, their relative luciferin levels, and availability of only local information. In the figure, *a, b, d*, and *f* are mobile agents where *c* and *e* are stationary agents. Since agent *a* has the highest luciferin among all the agents, it remains stationary. Mobile agent *b* has only one direction to move whereas agent *d* could move towards either *b* or *c*. Similarly, agent *f* has two possible directions to move. Note that the out-degree of *c* and *e* is zero as they represent stationary agents. For each mobile-agent *i*, the probability $p_j(t)$ of moving towards a neighbor *j* is given by:

$$p_j(t) = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \in N_i(t)} \ell_k(t) - \ell_i(t)} \tag{4}$$

where, $j \in N_i(t)$, $N_i(t) = \{ j : j \in S_{agents} \cup M_{agents}; d_{ij}(t) < r_d^i(t); \ell_i(t) < \ell_j(t) \}$ is the neighborhood of agent *i* at time *t*, $d_{ij}(t)$ represents the Euclidian distance between agents *i* and *j* at time *t*, $\ell_j(t)$ represents the luciferin level associated with agent *j* at time *t*, r^i_d (*t*) represents the variable local-decision range associated with agent *i* at time *t*, and r_s represents the radial range of the luciferin sensor. Let the agent *i* select an agent $j \in N_i(t)$ with a probability $p_j(t)$ given by Eq.(4). Then the discrete-time model of the agent movements is given as:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left\| x_j(t) - x_i(t) \right\|} \right) \tag{5}$$

where, *s* (> 0) is the step-size and ||.|| represents the Euclidian norm operator.

Local-decision range update rule: When the agents depend on only local information to decide their movements, it is expected that the number of signal sources captured would be a strong function of the radial sensor range. For instance, if the sensor range of each agent covers the entire workspace, all the agents move to the signal source with the highest intensity and all other signal sources (emitting at lower intensity) are ignored. Since we assume that *a priori* information about the signal source locations and the external distribution of the signal intensity in the environment is not available, in order to detect multiple sources, the sensor range must be made a varying parameter. For this purpose, we associate each agent *i* with a local-decision domain whose radial range r_d^i is dynamic in nature $(0 < r_d^i < r_s^i)$ (Krishnanand and Ghose 2006). A suitable function is chosen to adaptively update the local-decision domain range of each mobile agent. This is given by:

$$r_d^i(t+1) = \min\{ r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\} \} \tag{6}$$

where, β is a constant parameter and n_t is used as a threshold parameter to control the number of neighbors.

The leapfrogging effect: According to the GSO algorithm, a mobile agent with the maximum luciferin at a particular iteration remains stationary during that iteration. Ideally, the above property leads to a dead-lock situation when all the agents are located such that the peak-location lies outside the convex-hull formed by the agent positions. Since the agent movements are restricted to the interior region of the convex-hull, all the agents converge to an agent that attains maximum luciferin value during its movements within the convex-hull. As a result, all the agents get co-located away from the peak-location. However, the discrete nature of the movement-update rule automatically takes care of this problem which could be described in the following way. During the movement phase, each mobile-agent moves a distance of finite step-size s towards a neighbor. Hence, when an agent i approaches closer to an agent j that is located nearest to a peak such that the inter-agent distance becomes less than s, i leapfrogs over the position of j and becomes a leader to j. In the next iteration, i remains stationary and j overtakes the position of i thus regaining its leadership. This process of interchanging of roles between i and j repeats until they reach the peak. This phenomenon was also observed in simulations in (Krishnanand and Ghose 2006).

2.2 Convergence Results

We provide some theoretical proofs for the two luciferin update rules proposed in the previous section. We derive expressions for the luciferin levels at any time t for both stationary and mobile agent cases, respectively. For stationary agents, we extend the respective result to find their luciferin levels at steady state. For mobile-agents, we prove that, due to luciferin decay, the maximum luciferin level $\ell^{\max}$ is bounded asymptotically. This proof is similar to Stützle and Dorigos' proposition (Stützle and Dorigo 2002) proving the bounded nature of pheromone levels in their ant-colony algorithm. Moreover, we show that the luciferin ℓ_j of all mobile-agents, co-located at a peak x_i^*, converge to the same value ℓ^*_i.

Theorem 1. *For each stationary agent $i \in S_{agents}$, assuming that the update rule Eq. (3) is used, the luciferin value $\ell_i(t)$ at any time t is given by:*

$$\ell_i(t) = (1-\rho)^t \ell_0 + \left\{\frac{1-(1-\rho)^{t-1}}{\rho}\right\}\gamma J(x_i) \tag{7}$$

where, ℓ_0 is the initial luciferin level of agent i. Moreover, $\ell_i(t)$ asymptotically converges to $(\gamma/\rho)\ J(x_i)$.

Proof : Given that the initial luciferin value of the stationary agent i is ℓ_0, its luciferin level at iteration 1, using the luciferin update rule Eq. (3), is given by:

$$\ell_i(1) = (1-\rho)\ell_0 + \gamma J(x_i)$$

At iteration 2:

$$\ell_i(2) = (1-\rho)^2 \ell_0 + [(1-\rho)+1]\gamma J(x_i)$$

Proceeding in a similar way, at iteration *t*:

$$\ell_i(t) = (1-\rho)^t \ell_0 + [1+(1-\rho)+(1-\rho)^2 + \cdots + (1-\rho)^{t-1}]\gamma J(x_i)$$

$$= (1-\rho)^t \ell_0 + \left[\frac{1-(1-\rho)^{t-1}}{\rho}\right]\gamma J(x_i) \tag{8}$$

Since $0 < \rho < 1$, it is evident from Eq. (8) that as $t \to \infty$, $\ell_i(t) \to (\gamma/\rho)\, J(x_i)$. □

Theorem 2. *For each mobile-agent $i \in M_{agents}$, assume that the update rule (2) is used and define $P(t)$ as $P(t) = (1 - \rho)^t$. Now, the luciferin value $\ell_i(t)$, at any time t, is given by:*

$$\ell_i(t) = P(t)\ell_0 + \gamma \sum_{k=0}^{t-1} P(k) J(x_i(t-k)) \tag{9}$$

where, ℓ_0 is the initial luciferin level of agent i.

Proof : Given that the initial luciferin value of the mobile-agent *i* is ℓ_0, its luciferin level at iteration 1, using the update rule Eq. (2), is given by:

$$\ell_i(1) = (1-\rho)\ell_0 + \gamma J(x_i(1))$$

At iteration 2:

$$\ell_i(2) = (1-\rho)^2 \ell_0 + (1-\rho)\gamma J(x_i(1)) + \gamma J(x_i(2))$$

Proceeding in a similar way, at iteration *t*:

$$\ell\ (t) = (1-\rho)^t + (1-\rho)^{t-1}\gamma J(x_i(1)) + (1-\rho)^{t-2}\gamma J(x_i(2))$$

$$+ \cdots + (1-\rho)\gamma J(x_i(t-1)) + \gamma J(x_i(t))$$

$$= (1-\rho)^t + \gamma \sum_{k=0}^{t-1} (1-\rho)^k J(x_i(t-k)) \tag{10}$$

Using the definition $P(t) = (1 - \rho)^t$, we simplify Eq. (10) as below:

$$\ell_i(t) = P(t)\ell_0 + \gamma \sum_{k=0}^{t-1} P(k) J(x_i(t-k)) \tag{11}$$

Thus we prove the result in Eq. (9). Note that the second term in the above equation is a discrete-convolution of $\gamma P(t)$ and $J(x_i(t))$ between the summation limits 0 and t–1. □

Theorem 3. *Assuming that the update rule in Eq. (2) is used, the luciferin level $\ell_j(t)$, for any mobile-agent $j \in M_{agents}$, is bounded above asymptotically as follows:*

$$\lim_{t \to \infty} \ell_j(t) \le \lim_{t \to \infty} \ell^{\max}(t) = \left(\frac{\gamma}{\rho}\right) J_{\max} \tag{12}$$

where, J_{max} is the signal value at the strongest signal source.

Proof : Given that the maximum value of the signal at any point is J_{max} and the luciferin update rule in Eq. (2) is used, the maximum possible quantity of luciferin added to the previous luciferin level at any iteration t is γJ_{max}. Therefore, at iteration 1, the maximum luciferin of any mobile-agent j is $(1 - \rho)\ell_0 + \gamma J_{max}$. At iteration 2, it is $(1 - \rho)^2\ell_0 + [1 + (1 - \rho)]\gamma J_{max}$, and so on. Generalizing the process, at any iteration t, the maximum luciferin $\ell^{\max}(t)$ of any mobile-agent j is then given by:

$$\ell^{\max}(t) = (1-\rho)^t \ell_0 \sum_{i=0}^{t-1} (1-\rho)^i \gamma J_{\max} \tag{13}$$

Clearly,

$$\ell_j(t) \le \ell^{\max}(t) \tag{14}$$

Since $0 < \rho < 1$, from Eq. (13) we have that

$$t \to \infty \Rightarrow \ell^{\max}(t) \to \left(\frac{\gamma}{\rho}\right) J_{\max} \tag{15}$$

Using Eqs. (14) and (15), we have the result in Eq. (12). □

Theorem 4. *For all agents j co-located at signal source locations x_i^* associated with signal values $J(x_i^*) \le J_{\max}$ (where, i = 1, 2, …, m, with m as the number of signal sources), if the update rule in Eq. (2) is used, then $\ell_j(t)$ increases or decreases monotonically and asymptotically converges to $\ell_i^* = (\gamma/\rho)\, J(x_i^*)$.*

Proof : According to Eq. (2), $\ell_j(t) \ge 0$ always. The stationary luciferin ℓ_i^* associated with a signal source i satisfies the following condition:

$$\ell_i^* = (1-\rho)\ell_i^* + \gamma J(x_i^*) \Rightarrow \ell_i^* = \left(\frac{\gamma}{\rho}\right) J(x_i^*) \tag{16}$$

If $\ell_j(t) < \ell_i^*$ for agent *j* co-located at source location x_i^*, then using Eq. (16) we have

$$J(x_i^*(t)) > \left(\frac{\rho}{\gamma}\right)\ell_j(t) \tag{17}$$

Now,

$$\ell_j(t+1) = (1-\rho)\ell_j(t) + \gamma J(x_i^*) \tag{18}$$

$$> (1-\rho)\ell_j + \gamma\left(\frac{\rho}{\gamma}\right)\ell_j(t)$$

$$\Rightarrow \ell_j(t+1) > \ell_j(t) \tag{19}$$

i.e., $\ell_j(t)$ increases monotonically. Similarly, if $\ell_j(t) > \ell_i^*$ for agent *j* co-located at source location x_i^*, then using Eqs. (16) and (18), it is easy to show that

$$\ell_j(t+1) < \ell_j(t) \tag{20}$$

i.e., $\ell_j(t)$ decreases monotonically.

Now we prove the convergence of the sequence $\ell_j(t)$ by showing that the fixed point ℓ_i^* of the system in Eq. (18) is asymptotically stable. From Eq. (18), we can deduce the following relation:

$$\left|\ell_j(t) - \frac{\rho}{\gamma}J(x_i^*)\right| = (1-\rho)\left|\ell_j(t-1) - \frac{\rho}{\gamma}J(x_i^*)\right| \tag{21}$$

$$= (1-\rho)^2\left|\ell_j(t-2) - \frac{\rho}{\gamma}J(x_i^*)\right|$$

Proceeding in a similar way, we get

$$\left|\ell_j(t) - \ell_i^*\right| = (1-\rho)^t\left|\ell_j(0) - \ell_i^*\right| \tag{22}$$

Therefore,

$$\lim_{t\to\infty}\left|\ell_j(t)-\ell_i^*\right| = \lim_{t\to\infty}(1-\rho)^t\left|\ell_j(0)-\ell_i^*\right|$$

$$= 0, \because 0<(1-\rho)<1 \tag{23}$$

From Eq. (4), it is clear that the luciferin $\ell_j(t)$ of agent *j*, co-located at a peak-location x_i^*, asymptotically converges to ℓ_i^*. □

3 Experimental Results

We consider an environment where multiple sources of hazard, representative of several examples discussed earlier, radiate signals that pervade the environment and the nano-agents spread in the environment could sense the signal-strength at their respective locations. For our purpose, we use the following function to create the signal profile (Fig. 2):

$$\begin{aligned} J_1(x,y) &= c_1(a_1-x)^2\exp(-(x^2)-(y+a_2)^2) \\ &-c_2(a_3x-x^3-y^5)\exp(-x^2-y^2)-c_3\exp(-(x+a_4)^2-y^2) \end{aligned} \tag{24}$$

We consider a workspace of (-3, 3) × (-3, 3). The above signal-profile has peaks of different values (representing the signal level) at three different locations. We

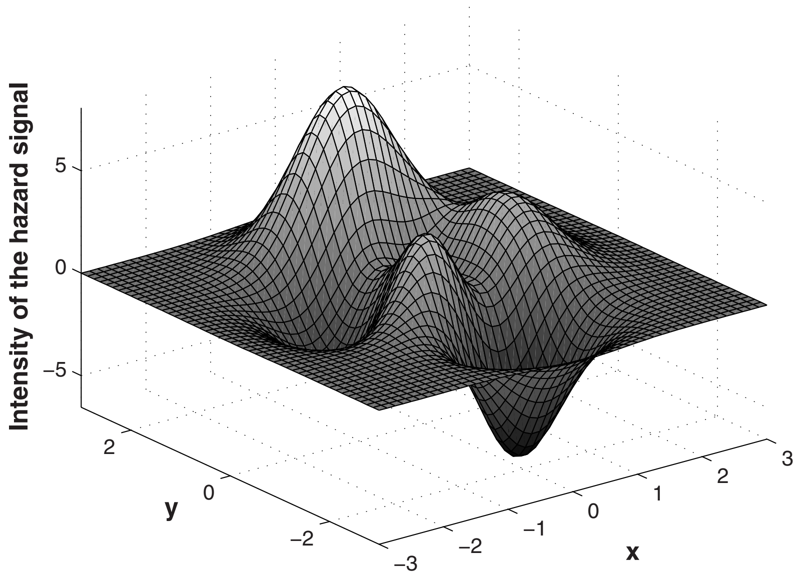


Fig. 2. The signal-profile chosen for the experiments

Table 1. Values of parameters of the signal profile chosen for the two experiments

c_1	c_2	c_3	a_1	a_2	a_3	a_4
3	10	0.33	1	1	0.20	1

Table 2. Values of algorithmic constants used in the first three experiments

ρ	γ	s	ρ_σ	$n_\dagger$	e	β
0.4	0.6	0.03	3	5	0.05	0.08

assume that the signal peaks are co-located with the signal source locations. We also assume that propagation of the signal is instantaneous, which means any decay or enhancement of signals is immediately felt in the same way in the environment and the qualitative nature of the signal profile remains unchanged. Note that the distance between the peak locations and signal-strength at each peak depends on the variables that parameterize the above function. We consider an agent to be co-located at a source location when it is located within a small ϵ-distance from the source location. The values of parameters used to generate the signal profile are given in Table 1 and the algorithmic constants used in the simulations are shown in Table 2. We consider three experimental scenarios, where we use random placement of agents, in order to demonstrate the working of the algorithm in heterogeneous swarms. We conduct another set of experiments where we examine the behavior of the algorithm when a grid based placement of stationary agents is used.

3.1 Experiment 1: Effect of Assortment Ratio on Number of Captured Signal Sources

The goal of the first set of experiments is to study the influence of assortment ratio a_r as defined in Eq. (1), on the number of signal sources detected. Figs. 3(a), 3(b), and 4(a) show the emergence of movements of the mobile agents and their eventual co-location at the hazard source(s) for assortment ratios of 0.2, 0.3, and 0.4, respectively. A total number of 100 agents was considered for all these experiments. Note that the number of sources detected increases with the increase in the assortment ratio for a fixed population size. In the third case, all the sources are detected within 200 iterations corresponding to a computation time of 2.67 sec. (The algorithm is coded using Matlab 7.0 on a 2GB RAM, 3 GHz Intel P4, Windows XP machine). However, the number of iterations and simulation time give only a qualitative idea and the time taken for the movements of mobile agents have to be considered in order to compute the time spent by the swarm to complete the task in a realistic hazard sensing application. Moreover, the algorithm would run parallel in all the agents and each agent needs to perform only its own computations, thereby reducing the overall computation time.

Fig. 5(a) shows the number of signal sources detected as a function of a_r for different population sizes. We observe that the assortment ratio required for 100%

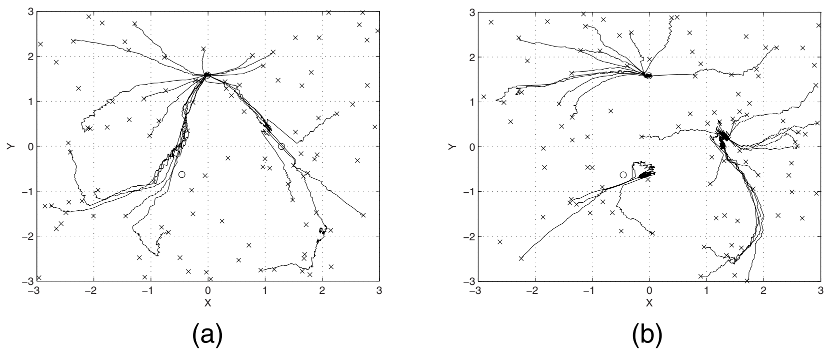


Fig. 3. a) Emergence of the movements of the mobile agents and their eventual co-location at the hazard sources; $n = 100$, $a_r = 0.2$ **b)** Emergence of the movements of the mobile agents and their eventual co-location at the hazard sources; $n = 100$, $a_r = 0.3$

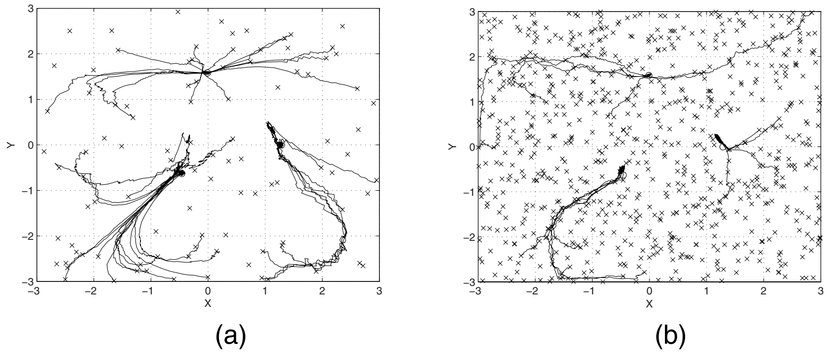


Fig. 4. a) Emergence of the movements of mobile agents and their eventual co-location at the hazard sources; $n = 100$, $a_r = 0.4$ **b)** Emergence of the movements of mobile agents and their eventual co-location at the hazard sources; $n = 800$, $a_r = 0.0187$

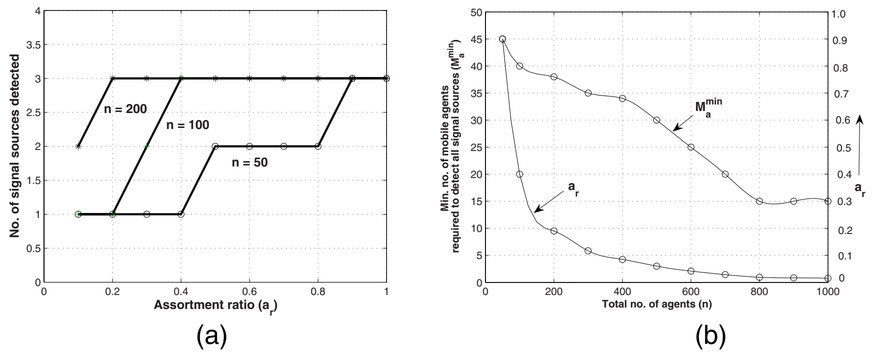


Fig. 5. a) Number of signal sources detected as a function of assortment ratio a_r for $n = 50$, 100, and 200 **b)** Minimum number of mobile agents required for 100% source-capture as a function of the total number of agents

source-capture decreases with increase in the total number of agents. Note that the number of mobile agents does not reduce in the same proportion as the assortment ratio. For instance, complete source-capture requires about 40 to 45 mobile agents for all three cases of population sizes (n = 50, 100, 200). However, we show in the next set of experiments that when the environment is densely populated with stationary agents, the minimum number of agents required for 100% source-capture begins to show a reducing trend with increase in the population size.

3.2 Experiment 2: Assortment Ratios for Capturing All Sources

We conduct the following set of experiments for different population sizes and assortment ratios corresponding to environments that are densely populated with stationary agents and sparingly populated with mobile agents. The basic motivation for this experimental scenario lies in the fact that the stationary sensors could be built at considerably low costs and hence they can be deployed in very large numbers. Consequently, we plot the minimum number of mobile agents required to detect all the signal sources as a function of the population size (Fig. 5(b)). Note that the change in slope at n = 400 indicates a steeper decline in the minimum number of mobile agents for 100% source-capture with increase in population size. Fig. 4(b) shows the emergence of movements of the mobile agents and their eventual co-location at the hazard sources when n = 800 and the number of mobile agents is only 15 (a_r = 0.0187). We can infer from this result and Fig. 5(b) that by using a large number of stationary sensors, only a few number of mobile agents is sufficient to accomplish a hazard sensing task that would otherwise require a relatively large number of mobile agents for its completion (Fig. 4(a)).

3.3 Experiment 3: Pre-defined Deployment of Mobile Agents

We consider heterogeneous swarms where stationary agents pervade the entire environment while the initial placement of the mobile-agents is restricted to regions such as corners and edges of the environment. We compare the results with that of homogeneous swarms in order to demonstrate the efficacy of using an assortment of stationary and mobile agents in these placement scenarios. In particular, we consider the following three deployments:

Left-bottom corner region: Initially, we deploy a homogeneous group of five mobile-agents in the region (–3,–2)×(–3,–2) (Experiments show that a minimum number of five agents is found to be necessary for a homogeneous mobile swarm in order to perform a robust search or foraging in the environment). Fig. 6(a) shows the emergence of agent-movements and co-location of the agents at a signal-source within 600 iterations. However, when a heterogeneous swarm of 800 agents is considered, where a set of 795 stationary agents is deployed in the entire environment and the 5 mobile agents are deployed in the region (–3,–2)×(–3,–2), the mobile-agents take only 100 iterations to capture the same source (Fig. 6(b)).

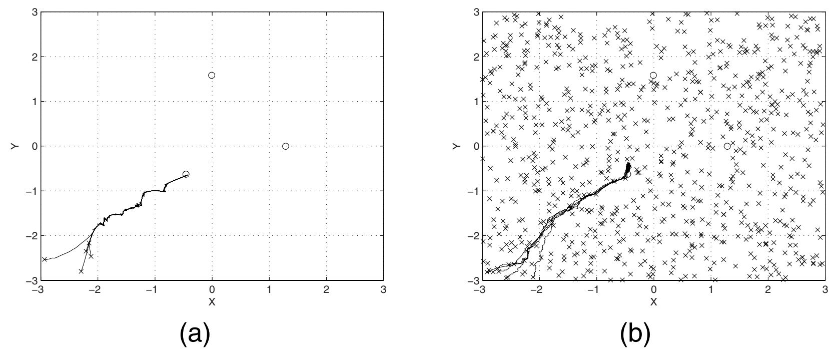


Fig. 6. a) Emergence of the agent movements when a homogeneous group of 5 mobile agents is deployed in the left-bottom corner of the environment **b)** Emergence of mobile agent movements when a heterogeneous group of 800 agents is deployed with 5 mobile agents deployed in the left-bottom corner of the environment

Right-edge region: In both the homogeneous and heterogeneous cases, a set of 20 mobile-agents is deployed in the region (2, 3) $\times$ (-3, 3), which represents the right-edge region of the environment. In the heterogeneous agents case, an additional set of 780 stationary agents is deployed in the entire environment. Figs. 7(a) and 7(b) show the emergence of mobile-agent movements in the homogeneous and heterogeneous cases, respectively. The mobile-agents are able to capture only one signal-source in the absence of the stationary agents, which can be explained in the following way: Note from Fig. 7(a) that the direction of swarm-advancement is heavily influenced by the initial location of a mobile-agent that is closest to a nearby source, which does not move until another agent (with a relatively less luciferin value) reaches it and performs a leapfrogging move (described in Sect. 2.1) in order to over-take it. Consequently, the property of the leapfrogging mechanism enforces all the swarm members to move in a single group and eventually home on to a single signal-source.

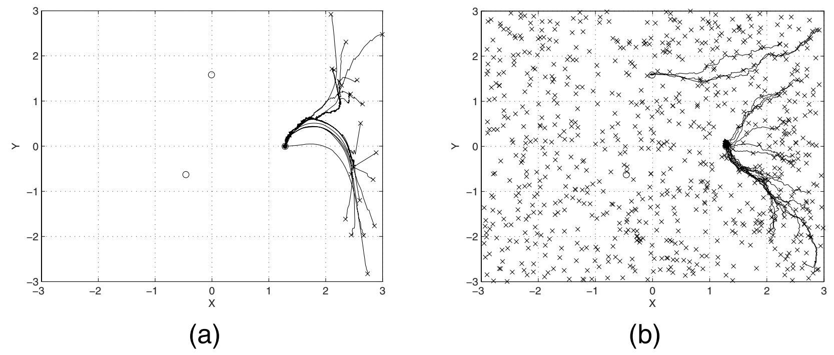


Fig. 7. a) Emergence of the agent movements when a homogeneous group of 20 mobile agents is deployed in the right-edge region of the environment **b)** Emergence of mobile agent movements when a heterogeneous group of 800 agents is deployed with 20 mobile agents deployed in the right-edge region of the environment

However, when stationary agents pervade the entire environment, they give rise to attraction-beacons at different locations corresponding to multiple sources, thereby enabling the exploration of the mobile-swarm in different directions in order to capture multiple sources. In Fig. 7(b), the mobile-swarm splits into two subgroups and eventually captures two of the three signal-sources.

Four Corners: In the first set of experiments, groups of 6 and 3 mobile-agents at each corner are deployed to constitute homogeneous swarms. Emergence of agent-movements within 500 iterations for each case is shown in Figs. 8(a) and 9(a), respectively. In the next set of experiments, heterogeneous groups of 800 agents are considered where 6, 3, 2, and 1 mobile-agent(s), respectively, are deployed at each corner and the rest of them, which are stationary agents, pervade the entire environment. Figs. 8(b), 9(b), 10(a), and 10(b) show the emergence of mobile-agent movements within 200 iterations for each case. Note from Fig. 10(b) that even a single

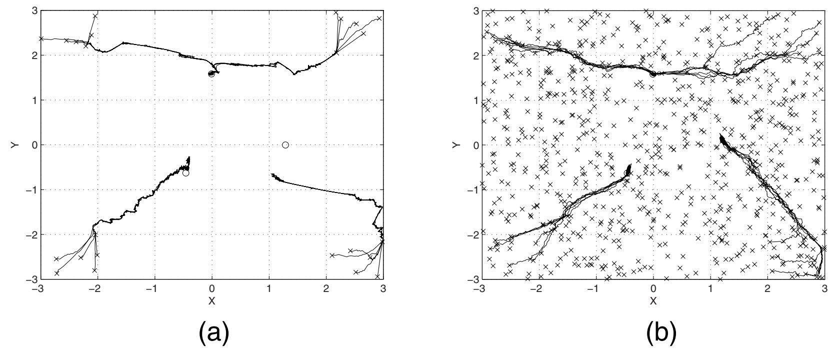


Fig. 8. a) Emergence of the agent movements when a homogeneous group of 6 mobile agents is deployed in each corner of the environment **b)** Emergence of mobile agent movements when a heterogeneous group of 800 agents is deployed with 6 mobile agents deployed in each corner of the environment

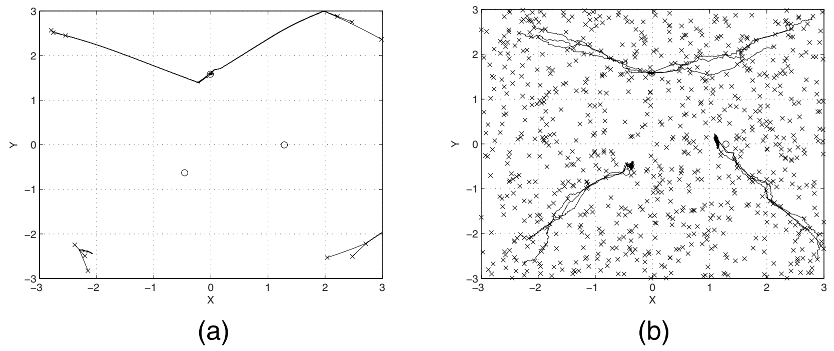


Fig. 9. a) Emergence of the agent movements when a homogeneous group of 3 mobile agents is deployed in each corner of the environment **b)** Emergence of mobile agent movements when a heterogeneous group of 800 agents is deployed with 3 mobile agents deployed in each corner of the environment

mobile-agent from a corner location could find its way to a nearby source unlike in the homogeneous agent case where a minimum number of five agents is required for the swarm to acquire a search capability.

3.4 Experiment 4: Deployment of Stationary Agents in a Uniform Grid

In this set of experiments, we examine the algorithm's behavior when the stationary agents are deployed in a grid-configuration. We use the functions $J_1(x, y)$ in Eq. (24) and $J_2(x, y)$ (Fig. 11) given below for these experiments.

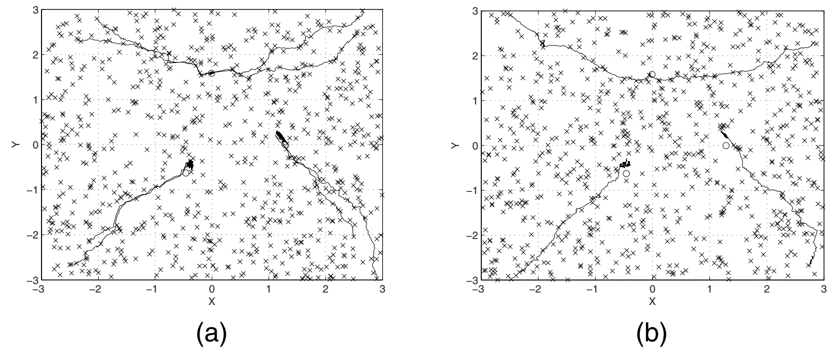


Fig. 10. a) Emergence of mobile agent movements when a heterogeneous group of 800 agents is deployed with 2 mobile agents deployed in each corner of the environment **b)** Emergence of mobile agent movements when a heterogeneous group of 800 agents is deployed with 1 mobile agent deployed in each corner of the environment

$$J_2(x,y) = \sum_{i=1}^{Q} a_i \exp(-b_i((x-x_i)^2 + (y-y_i)^2)) \qquad (25)$$

where, Q represents the number of peaks and (x_i, y_i) represents the location of each peak. A set of five peaks ($Q = 5$) is considered for the purpose. The values of a_i, b_i, x_i, and y_i are generated according to the following equations:

$$a_i = 1 + 2rand()$$
$$b_i = 2 + rand()$$
$$x_i = -5 + 10rand()$$
$$y_i = -5 + 10rand()$$

where, $rand()$ is a uniform random variable within the interval [0, 1].

Note that a_i, b_i, x_i, and y_i are random variables and each set of instantiations of these random variables gives rise to a different multimodal function profile. The corresponding values of $\{a_i, b_i, x_i, y_i, i = 1,\ldots, 5\}$ used to generate the function $J_2(x, y)$ are shown in Table 3.

Table 3. Values of a_i, b_i, x_i, and y_i used to generate the function profile $J_2(x, y)$

	1	2	3	4	5
a_i	1.7206	2.0970	1.5235	2.1947	1.0986
b_i	2.1421	2.4017	2.9246	2.5010	2.4800
x_i	–0.6813	1.3427	3.0303	–4.1612	4.4546
y_i	4.1594	1.0199	–2.4644	3.7345	0.1340

Table 4. Values of algorithmic constants used in the fourth experiment

ρ	γ	s	ρ_σ	n_t	ϵ	β
0.4	0.6	0.03	1	2	0.05	0.08

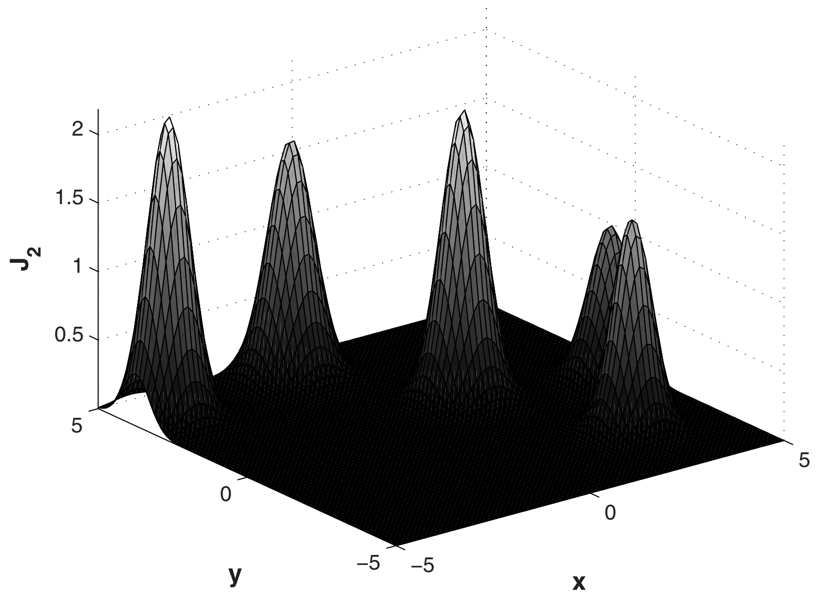


Fig. 11. The signal profile used to examine algorithm's behavior when stationary agents are deployed in a grid-configuration

A square-shaped workspace is used. The total number of stationary agents S_a in the workspace when they are deployed in a grid-configuration (Fig. 12) is given by:

$$S_a = \left(1 + \frac{w}{d}\right)^2 \tag{26}$$

where, w is the width of the workspace and d is the spacing between adjacent grid points. Values of $w = 10$ and $r_s = 1$ are used. The values of the algorithmic constants used in this set of experiments are shown in Table 4. Note from Eq. (26) and Fig. 12 that for fixed values of w and r_s, the number of stationary-agent-neighbors of a mobile agent generally increases with a decrease in the value of d. However, when d approaches the values of $\sqrt{2}r_s$, r_s, and $r_s/\sqrt{2}$, there is a leap-increment in the number of neighbors of a mobile agent. For instance, when $d > \sqrt{2}r_s$, a mobile agent that is located at the center of a grid-cell can only have other mobile agents as possible neighbors. However, when $d = \sqrt{2}r_s$, it acquires a set of four stationary agents as neighbors (Fig. 13 (a)). Similarly, a mobile agent co-located at a stationary agent's location acquires four stationary-agent-neighbors when $d = r_s$ (Fig. 13 (b)). It acquires an additional set of four neighbors when $d = r_s/\sqrt{2}$ (Fig. 13 (c)). The influence of the spacing d on the number of neighbors of a mobile agent, as described above, leads to multiple phase-transitions in the heterogeneous-swarm behavior, which can be

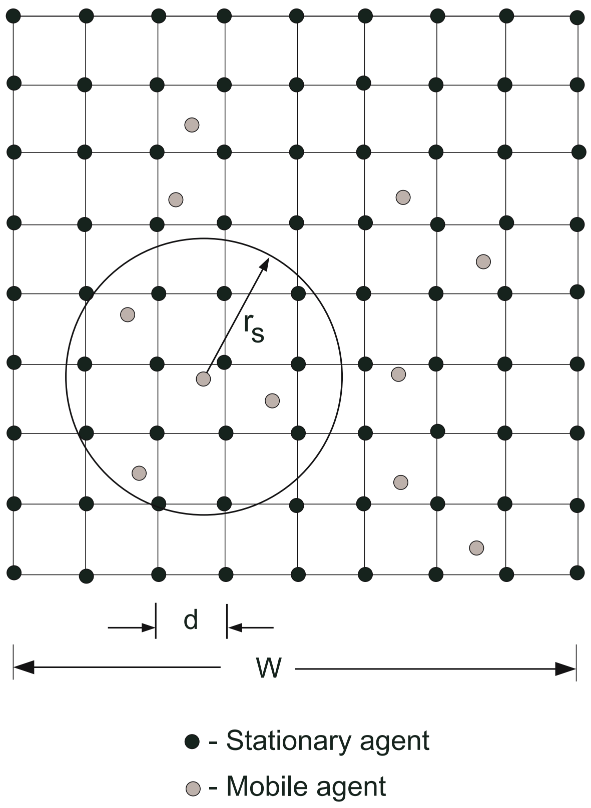


Fig. 12. Stationary agents deployed in a grid-configuration. The number of stationary agents that are within the sensing range of a mobile agent is a function of the spacing between the grid points.

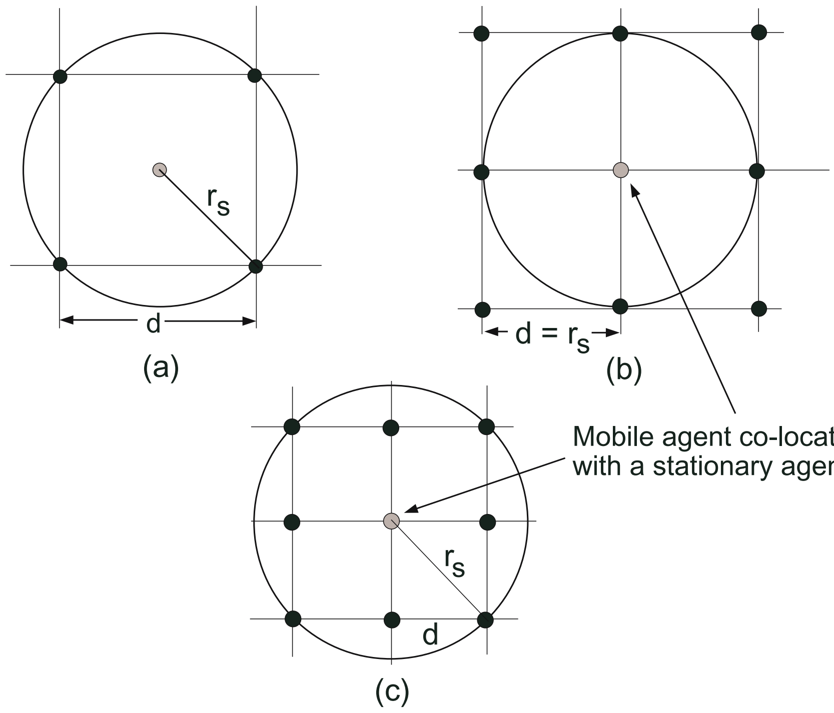


Fig. 13. The nature of the local neighborhood of a mobile agent for three different values of d: **a)** When $d = \sqrt{2}r$, a mobile agent at the center of a grid-cell acquires four stationary agents as neighbors **b)** When $d = r_s$, a mobile agent that is co-located with a stationary agent acquires four stationary agents as neighbors **c)** When $d = r_s/\sqrt{2}$, a mobile agent that is co-located with a stationary agent acquires four additional stationary agents as neighbors (that is, totally eight neighbors)

captured when we plot the minimum number of mobile agents required to capture all the sources as a function of the number of stationary agents.

In each experiment, the number of stationary agents used is kept as a perfect-square in order to ensure a symmetric deployment of the stationary agents. The values of S_a that are perfect-squares and corresponding to values of d that are nearest to $d = \sqrt{2}$, 1, and $1/\sqrt{2}$ (since $r_s = 1$) are 64, 121, and 225, respectively. Figs. 14 and 15 show the plots of the minimum number of mobile agents $M_a^{\min}$ required to capture all the sources as a function of S_a when the functions $J_1(x, y)$ and $J_2(x, y)$ are used, respectively. At each value of S_a, a set of ten experimental trials is conducted and $M_a^{\min}$ is chosen as the minimum number of mobile agents for which 100% source-capture is achieved in all the ten trials. Note from Figs. 14 and 15, that the occurrence of phase-transitions, in the neighborhoods of $S_a = 64$, 121, and 225, shown in simulations are in perfect agreement with the analytical predictions made above. Moreover,

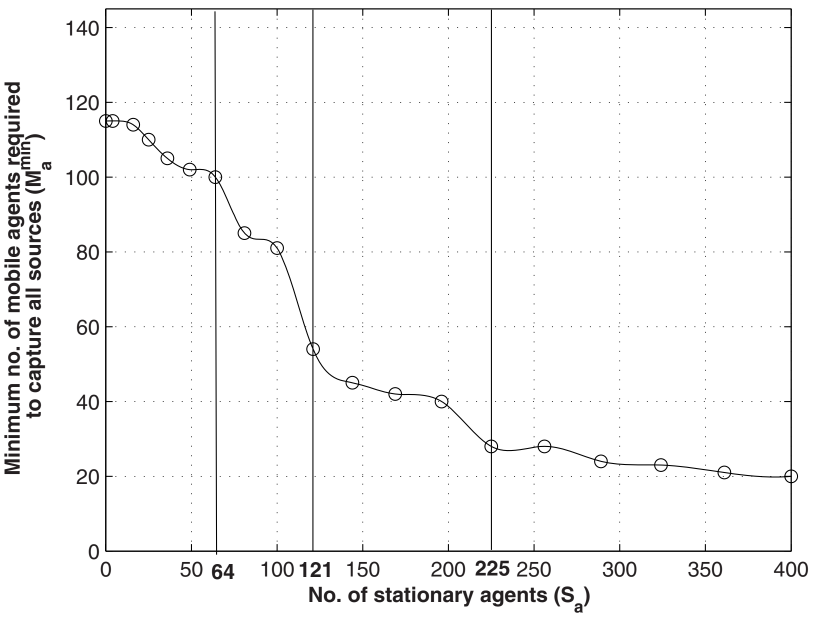


Fig. 14. The minimum number of mobile agents $M_a^{\min}$ for 100% source-capture as a function of the number of stationary agents S_a, when the signal-profile $J_1(x, y)$ is used

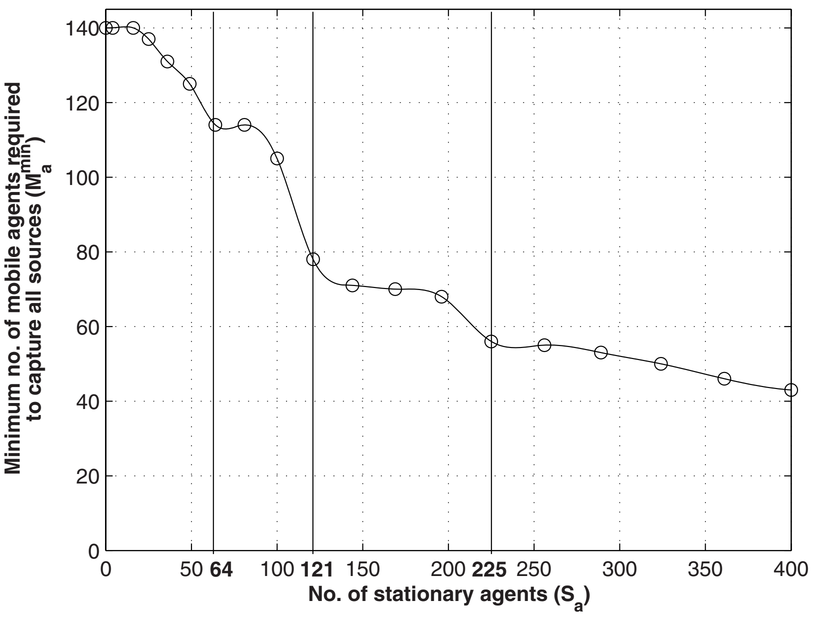


Fig. 15. The minimum number of mobile agents $M_a^{\min}$ for 100% source-capture as a function of the number of stationary agents S_a, when the signal-profile $J_2(x, y)$ is used

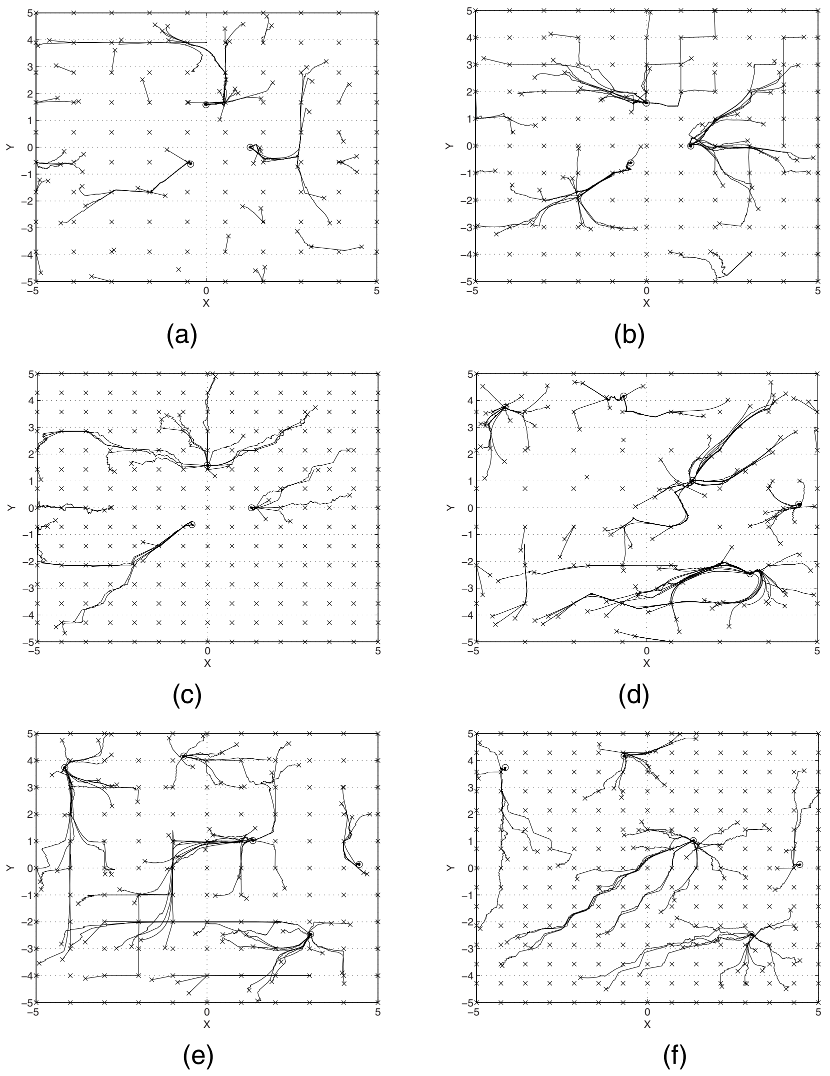


Fig. 16. Emergence of agent movements for the following cases: **a)** Function $J_1(x, y)$, $S_a = 64$, $M_a = 100$ **b)** Function $J_1(x, y)$, $S_a = 121$, $M_a = 54$ **c)** Function $J_1(x, y)$, $S_a = 225$, $M_a = 28$ **d)** Function $J_2(x, y)$, $S_a = 64$, $M_a = 114$ **e)** Function $J_2(x, y)$, $S_a = 121$, $M_a = 78$ **f)** Function $J_2(x, y)$, $S_a = 225$, $M_a = 56$

the similarities between plots in Fig. 14 and Fig. 15 indicate the fact that phase transition behavior of the heterogeneous-swarm is invariant to the kind of multimodal signal profile spread in the environment. A minor difference between the two plots is a vertical shift of the graph in Fig. 15, which may be attributed to the fact that the function $J_2(x, y)$ has more peaks than the function $J_1(x, y)$ and hence requires relatively more number of mobile agents for 100% peak-capture. The emergence of agent-movements and eventual capture of all the sources at various phase-transition points are shown in Fig. 16. Note from Figs. 16 (b) and 16(e) that some of the agents move on the edges of the grid-cells. This occurs due to the fact that in these cases, since $d = r_s$, an agent that is co-located with a stationary agent has stationary-agent-neighbors on the edges of four neighboring grid-cells as shown in Fig. 13 (b) and hence moves towards one of them if there are no other mobile-agent-neighbors.

4 Concluding Remarks

We address the problem of hazard sensing in ubiquitous environments by using a heterogeneous swarm of nano-agents that implement a modification to an algorithm called the glowworm swarm optimization (GSO) algorithm in order to sense and eventually co-locate at the source locations of the hazard. The heterogeneous swarm is composed of an assortment of stationary and mobile agents where the stationary agents are embedded into the environment and act only as stationary sensors, while the mobile agents use the information cues received from stationary and/or mobile neighbors in order to find their way to the source locations. Extensive simulations in various experimental scenarios and comparison of results with that of homogeneous swarms demonstrate the significance of supplementing a mobile-agent group with a large number of inexpensive stationary agents for hazard sensing applications. We also show that a grid based deployment of stationary agents leads to multiple phase-transitions in the heterogeneous swarm behavior. Further work includes testing the performance of the algorithm on signal-profile models that are more representative of realistic hazard situations and conducting real experiments with a heterogeneous swarm of stationary sensors and real mobile robots.

References

Bilchev, G., Parmee, I.C.: The ant colony metaphor for searching continuous design spaces. In: Proceedings of AISB Workshop on Evolutionary Computing. LNCS, pp. 25–39. Springer, Heidelberg (1995)

Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems, pp. 183–203. Oxford University Press, Oxford (1999)

El-Zabadani, H., Helal, A., Yang, H.: A mobile sensor platform approach to sensing and mapping pervasive spaces and their contents. In: Proceedings of the International Conference on New Technologies of Distributed Systems (2007)

Jabbar, H., et al.: Ubiquitous factory An automation of nuclear power plant. In: Proceedings of International Conference on Ubiquitous Information Technologies and Applications, pp. 1266–1273 (2007)

Kim, H.-C., et al.: A framework for atmospheric environment monitoring with ubiquitous sensor network. In: Proceedings of International Conference on Ubiquitous Information Technologies and Applications, pp. 1240–1248 (2007)

Krishnanand, K.N., Ghose, D.: Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. Multiagent and Grid Systems: Special Issue on Recent Progress in Distributed Intelligence, 209–222 (2006)

Krishnanand, K.N., et al.: Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources. In: Proceedings of IEEE International Conference on Robotics and Automation, Orlando, pp. 958–963 (2006)

Krishnanand, K.N., Ghose, D.: Glowworm swarm optimization algorithm for hazard sensing in ubiquitous Environments. In: Proceedings of International Conference on Ubiquitous Information Technologies and Applications, pp. 1499–1501 (2007)

Stützle, T., Dorigo, M.: A short convergence proof for a class of ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation, 358–365 (2002)

Weiser, M., Gold, R., Brown, J.S.: Origins of ubiquitous computing research at PARC in the late 1980's. IBM Systems Journal, 693–696 (1999)

Self-organization in Evolution for the Solving of Distributed Terrestrial Transportation Problems

Jean-Charles Créput and Abderrafiaâ Koukam

Systems and Transportation Laboratory
University of Technology of Belfort-Montbeliard, 90010 Belfort, France
{Jean-Charles.Creput, Abder.Koukam}@utbm.fr

1 Introduction

The method presented in this chapter has its origin in adaptive meshing, using planar honeycomb structures as a tool to dimension radio-cellular network according to mobile traffic (Créput et al. 2000, 2005; Créput and Koukam 2006). Here, the approach has been transferred and generalized to a terrestrial transportation context. The transport mesh is a geometric structure, in the plane, that adapts and modifies its shape according to traffic demands. By separating the transportation network from the underlying demands, the approach theoretically allows to deal with noisy or incomplete data as well as with fluctuating demand. Furthermore, continuous visual feedback during simulations is naturally allowed.

Here, a natural tool which is exploited to implement adaptive meshing is the self-organizing map (SOM) (Kohonen 2001) algorithm, a neural network approach dealing, when applied in the plane, with visual patterns moving and adapting to brut distributed data. Its main emergent property is to allow adaptation by density and topology preservation of a planar graph (transport mesh) to an underlying data distribution (demand set). Then, the positioning of distributed transportation facilities and infrastructures of the network will reflect the data density distribution and preserve a given topology for the connected network components. Hence, the SOM naturally deals with incomplete or noisy data as well as with stochastic demands. We exploit also the natural property of the SOM of being a center based clustering algorithm which adds topologic relationships between cluster centers. The topologic relationships naturally represent routes and integrate clustering and vehicle routing in a unified way.

This paper generalizes the adaptive meshing concept and summarizes several applications already developed for distributed terrestrial optimization problems (Créput and Koukam 2007a, 2007b; Créput et al. 2007; Hayari et al. 2005). A general clustering and routing optimization problem encompassing the different applications is given. It is called unified clustering and routing problem (UCRP).

To solve the optimization problem, following hybridization of meta-heuristics as done for example in (Boese et al. 1994; Gambardella et al. 1999; Mühlenbein 1991), we present an evolutionary framework which incorporates self-organizing maps as internal operators. It is called memetic SOM by reference to memetic algorithms (Moscato 1999), which are hybrid evolutionary algorithms (EA) incorporating a neighborhood search.

The evolutionary dynamics consists of interleaving the SOM execution with specific operators, such as fitness evaluation, selection operators and greedy insertion/move operators. Operators have a similar structure based on closest point findings and distributed local moves performed in the plane. Theoretically, self-organization is intended to confer robustness according to demand fluctuation and noisy data. Evolution through selection is intended to guide a population based search toward near-optimal solutions.

We show that the approach lets extend self-organizing neural networks to a large class of applications and improve its performance. We present applications to transportation network dimensioning, clustering problems as clustering k-median problem (Arora 1998), combined clustering and routing problems (UCRP), and classical vehicle routing problems such as travelling salesman problem (TSP), capacitated and time duration vehicle routing problems (VRP, DVRP) (Christofides et al. 1979), vehicle routing problem with time windows (VRPTW) (Solomon 1987).

The chapter is organized as follows. Section 2 presents the combined vehicle clustering and routing problem. Objectives and constraints are given. Section 3 describes the optimization framework, which allows to configure and execute evolutionary algorithms embedding neural networks. Section 4 presents applications to concrete problems of clustering and routing. Evaluation against SOM based approaches and against state of the art heuristics are reported. Finally, the last section is devoted to the conclusion and further research.

2 Problem Statement

2.1 Basic Concepts

1) Unit grid. The unit grid is a rectangular grid of size $X \times Y$ matching some geographical area. It defines the finite set L of possible locations in the plane, called pixels. Pixels are referred by their coordinates in the grid. Using integer coordinates rather than floating point values is intended to allow computational efficiency. The metric is the usual Euclidean distance, denoted *d*(*p*, *p'*) for points *p* and *p'* of the plane.

2) Requests. We denote by $V = \{r_1, ..., r_n\}$, the finite set of customer demands, called requests. Each request $r_i \in V$ has a geographic location $l_i \in L$. It has a non-negative demand q_i, a service time s_i and a time window (a_i, b_i). If a vehicle arrives at a location where request r_i is intended to be served, the vehicle can not begin the service before a_i. The vehicle has to arrive before b_i. Service is done with service time s_i.

3) Transport mesh. Let $B = \{n_1, ..., n_k\}$ being a finite set of cluster centers, also called transport points or bus-stops, localized by their coordinates in the unit grid. A transport mesh is a set of interconnected routes. Formally, it is a collection $R = \{R_1, ..., R_m\}$ of *m* routes, where each route is a sequence $R_i = (n_0^i, ..., n_j^i, ..., n_{k_i}^i)$, $n_j^i \in B$, of k_i+1 successive cluster centers. To each route is associated a single vehicle. We then denote and identify a vehicle with its route R_i.

4) Request assignment. In our approach, the main difference with classical vehicle routing modeling is that routes are defined by an ordering of cluster centers, rather than by an ordering of customer requests. It follows that each request r must be assigned to a single cluster center $n_r \in B$ in one of the m routes. Each vehicle has a load L_i defined as the sum of its assigned request quantities. A vehicle has a travel time T_i defined as the time for the vehicle starting from transport point n_0^i, and following intermediate straight line paths $\left(n_j^i, n_{j+1}^i\right)$, $j \in \{0, \ldots, k_i-1\}$, to reach its final transport point $n_{k_i}^i$ at a given constant speed, adding service time of its assigned requests. Vehicles have capacity C and maximum time duration D. We denote by $t_{arr}\left(n_r\right)$ the time of arrival of the vehicle to point n_r for each request $r \in V$.

5) Induced graph. Routes can share common transport points. They define a graph structure. The induced undirected graph $G_R = (N, E)$ of an interconnected set of routes $R = \{R_1, \ldots, R_m\}$ is defined as follows: N is the set of vertices composed by all cluster centers defining routes, E is the set of edges composed of any two successive centers from routes. The induced graph, or network, is an underlying intermediate structure between demands and vehicles. It can be used to compute shortest paths from an origin point to a destination point. Here, its main interest is of being the visual pattern on which the self-organizing map algorithm directly operates.

2.2 Unified Clustering and Routing Problem

The general problem considered is a unified clustering and routing problem. It is stated as follows:

Unified Clustering and Routing Problem (UCRP). The problem input is given by a set of requests $V = \{r_1, \ldots, r_n\}$ and a set of interconnected routes $R = \{R_1, \ldots, R_m\}$. Using notations and definitions of previous section, the problem consists of finding cluster center locations, except for some fixed transport points, and assignment of requests to cluster centers and vehicles, in order to minimize the following objectives:

$$length = \sum_{i=1,..,m,\ j=0,\ldots,k_i-1} d\left(n_j^i, n_{j+1}^i\right), \tag{1}$$

$$distortion = \sum_{i=1,\ldots,n} d\left(r_i, n_{r_i}\right), \tag{2}$$

subject to the capacity constraint:

$$L_i \leq C \,,\; i \in \{1,...,m\}, \tag{3}$$

and time duration constraint:

$$T_i \leq D \,,\; i \in \{1,...,m\}, \tag{4}$$

and time-window constraint:

$$\underset{r_i \in V}{Min}\left(b_i - t_{arr}\left(n_{r_i}\right)\right) \geq 0 \,. \tag{5}$$

Objective *length* in (1) is the routes total length. Objective *distortion* in (2) is the sum of distances from request locations to their assigned bus-stops, it is called distortion measure. The problem can be seen as a combination of a standard vehicle routing problem with the well-known Euclidean k-median problem (Arora 1998), using a transport mesh and adding time windows. Note that if we replace the not squared distances in objective *distortion* (2) by the squared distances, we retrieve the k-mean problem for which fast computational methods are k-mean algorithm and its stochastic version called vector quantization (VQ) algorithm (Kohonen 2001). As stated in (Kohonen 2001) the SOM algorithm itself extends VQ by adding topologic relationships between cluster centers. Replacing squared distances by maximum distance yields to the k-center problem. Here, we assume that requests are served in parallel inside each cluster, and that the cluster size is adjusted depending on the application under consideration.

Since the problem has two conflicting objectives of both length (1) and distortion (2), we have to take care of what is called an optimal solution. We say that a first (admissible) solution dominates an other (admissible) solution if the two objectives of the former are inferior to those of the latter, one of them being strictly inferior. Then, an optimal solution is a solution which is dominated by no other solution. The set of such non comparable optimal solutions are called Pareto optimal solutions or optimal non dominated solutions.

The set of Pareto optimal solutions are possibly numerous. Solutions with *distortion* = 0 and minimum *length* are solutions of an Euclidean VRPTW. Classical TSP and VRP are subclass problems of it. VRP is obtained by removing the time window constraint, TSP by considering a single route and no constraint. Solutions with minimum *distortion* (discarding *length*) are solutions of the Euclidean k-median problem. Whereas, considering a bound stated on one or two objectives yields to non comparable solutions which are possibly not a VRP nor a k-median optimal solution. Since objectives are possibly conflicting, these intermediate solutions are the compromises that are useful and interesting for application of combined clustering and routing. Discarding constraints and considering a complex graph of interconnected routes adapted according to (1) and (2), leads to what we call the network dimensioning problem. The problem can also be seen as a clustering version of the VRPTW (Solomon 1987).

3 Evolutionary Approach Embedding Self-organizing Map

3.1 Self-organizing Map

The self organizing map is a non supervised learning procedure performing a non parametric regression that reflect topological information of the input data set (Kohonen 2001). The standard algorithm operates on a non directed graph $G = (A, E)$, called the network, where each vertex $n \in A$ is a neuron having a location $w_n = (x, y)$ in the plane. The set of neurons A is provided with the d_G induced canonical metric, $d_G(n, n') = 1$ if and only if $(n, n') \in E$, and with the usual Euclidean distance $d(n, n')$.

```
Repeat niter times.
    1. Randomly extract a point p from the demand set.
    2. Perform competition to select the winner vertex n* according to p.
    3. Apply learning law to move the neurons of a neighborhood of n*.
    4. Slightly decrease learning rate α and radius σ of neighborhood.
End Repeat.
```

Fig. 1. Self-organizing map algorithm

The training procedure, summarized in Fig. 1, applies a given number of iterations *niter* to a graph network. The data set is the set of demands, or customers. Vertex coordinates are randomly initialized into an area delimiting the data set. Each iteration follows four basic steps. At each iteration t, a point $p(t)$ is randomly extracted from the data set (extraction step). Then, a competition between neurons against the input point $p(t)$ is performed to select the winner neuron n^* (competition step). Usually, it is the nearest neuron to $p(t)$ according to Euclidean distance.

$$w_n(t+1) = w_n(t) + \alpha(t).h_t(n^*, n).(p - w_n(t)) \tag{6}$$

Then, the learning law (6) (triggering step) is applied to n^* and to all neurons within a finite neighborhood of n^* of radius σ_t, in the sense of the topological distance d_G, using learning rate $\alpha(t)$ and function profile h_t. The function profile is given by the Gaussian in (7).

$$h_t(n^*, n) = \exp\left(-d_G(n^*, n)^2 / \sigma_t^2\right) \tag{7}$$

Finally, the learning rate $\alpha(t)$ and radius σ_t are slightly decreased as geometric functions of time (decreasing step). To perform a decreasing run within t_{max} iterations, at each iteration t coefficients $\alpha(t)$ and σ_t are multiplied by exp(ln(x_{final} / x_{init}) / $t_{\max}$), with respectively $x = \alpha$ and $x = \sigma$, x_{init} and x_{final} being respectively the values at starting and final iteration.

Examples of a basic iteration are shown in Fig. 2. We can see in (a)-(b), a basic iteration on a route, and in (c)-(d), how neighborhood influence can be subdivided

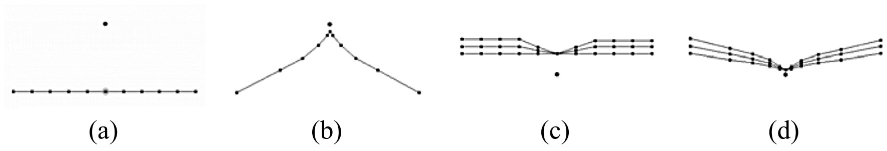


Fig. 2. A single SOM iteration with learning rate α and radius σ. (a)(c) Initial configuration. (b) $\alpha = 0.9$, $\sigma = 4$. (d) $\alpha = 0.75$, $\sigma = 4$.

along possibly interconnected routes. Application of SOM to a set of routes $R = \{R_1, ..., R_m\}$ consists of applying iterations to the undirected graph $G_R = (N, E)$ induced by R, with N the cluster center set, and E the set of edges from routes.

3.2 Evolutionary Embedding Strategy

A construction loop as well as an improvement loop is instantiated based on the generic memetic loop structure presented in Fig. 3. A construction loop starts its execution with solutions having randomly generated vertex coordinates into a rectangle area containing demands. The improvement loop starts with the single best previously constructed solution, which is duplicated in the new population. The main difference between the construction and improvement loops is that the former is responsible for creating an initial ordering from random initialization. It follows that SOM processes embedded in the construction loop will have a larger initial neighborhood, proportional to *n*, to deploy the initial network. The improvement loop, however, is intended for local improvements using SOM processes with small and constant initial neighborhood sizes.

```
Initialize population with Pop randomly generated individuals.
Do while not Gen generations are performed.
    1. Apply one or more standard SOM processes (denoted
    SOM), with their own parameter settings, to each individ-
    ual in population separately. /* Each SOM operator is ap-
    plied with probability prob, performing niter iterations
    at each generation, for each individual */
    2. Apply mapping operator MAPPING to each individual in
    population to assign each demand to a closest vertex.
    3. Apply fitness evaluation operator FITNESS to each in-
    dividual in population. /* The fitness value depends on
    the problem under consideration */
    4. Save the best individual encountered.
    5. Apply selection operator SELECT.
    6. Apply elitist selection operator SELECT_ELIT.
    7. Apply operators from set {SOMVRP, SOMDVRP, SOMTW},
    that are derived from the self-organizing map algorithm
    structure, to optimize individual routes, or to perform
    greedy insertion moves of the residual (not inserted) de-
    mands.
 End do.
 Report best individual encountered.
```

Fig. 3. The generic evolutionary loop embedding self-organizing map

Thus, SOM processes can play different roles within their embedding loop. Details of operators are the followings:

1) Self-organizing map operator. It is the standard SOM applied to the graph network. It is denoted by its name and its internal parameters, as $SOM\left(\alpha_{init}, \alpha_{final}, \sigma_{init}, \sigma_{final}, t_{\max}\right)$. One or more instances of the operator, with their own parameter values, can be combined. A SOM operator executes *niter* basic iterations by individual, at each generation. Parameter $t_{\max}$ is the number of iterations defining a long decreasing run performed in the stated generation number *Gen*, for each individual. Other parameters define the initial and final values of learning rate and neighborhood size. The operator can be used to deploy the network toward customers in construction phase, or to introduce punctual moves to exit from local minima in improvement phase.

2) SOM derived operators. Three operators are derived from the SOM algorithm structure for dealing with the VRP and VRPTW. The first operator, denoted *SOMVRP,* is a standard SOM restricted to be applied on a randomly chosen vehicle, using customers already inserted into the vehicle/route. It helps eliminate remaining crossing edges in routes. While capacity constraint is greedily tackled by the mapping/assignment operator below, two operators, denoted *SOMDVRP* and *SOMTW*, deal respectively with the time duration and time window constraints. They performs greedy insertion moves of cluster centers toward customers not already inserted in routes, a vehicle cluster center being selected if it leads to the smallest time increase.

3) Mapping/assignment operator. This operator, denoted *MAPPING*, generates solutions by inserting customers into routes and possibly modifying the shape of the network accordingly. The operator greedily maps customers to their nearest vertex, not already assigned, for which vehicle capacity constraint is satisfied. Then, specifically for classical vehicle routing problems for which *distortion* = 0, the operator moves the vertices to the location of their assigned customer (if exist) and dispatches regularly (by translation) other vertices along edges formed by two consecutive customers in a route.

4) Fitness operator, denoted *FITNESS*. Once the assignment of customers to routes has been performed, this operator evaluates a scalar fitness value for each individual that has to be maximized and which is used by the selection operator. The value returned is $fitness = sat - \alpha \times length - \beta \times distortion$, where α and β are weighting coefficients depending on the problem under consideration, and *sat* is the number of customers that are successfully assigned to routes. Admissible solutions are the ones for which $sat = n$, n being the number of customers.

5) Selection operators. Based on fitness maximization, at each generation the operator denoted *SELECT* replaces *Pop*/5 worst individuals, which have the lowest fitness values in the population, by the same number of bests individuals, which have the highest fitness values in the population. An elitist version *SELECT_ELIT* replaces *Pop*/10 worst individuals by the single best individual encountered during the current run.

4 Applications to Terrestrial Transportation Problems

4.1 Network Dimensioning Problems

The following example illustrates the visual specificity of SOM and the adaptive meshing concept for radio-cellular networks (Créput et al. 2005; Créput and Koukam 2006). The goal is to adjust an intermediate structure (the network) to an underlying distribution of demands, shown in Fig. **4**(a), subject to topology constraints. This is illustrated for cellular network dimensioning in Fig. **4**(b), with a honeycomb mesh representing cell transceiver base stations covering a territory. The terrestrial transportation case is illustrated in Fig. **4**(c), where interconnected lines stand for interconnected routes.

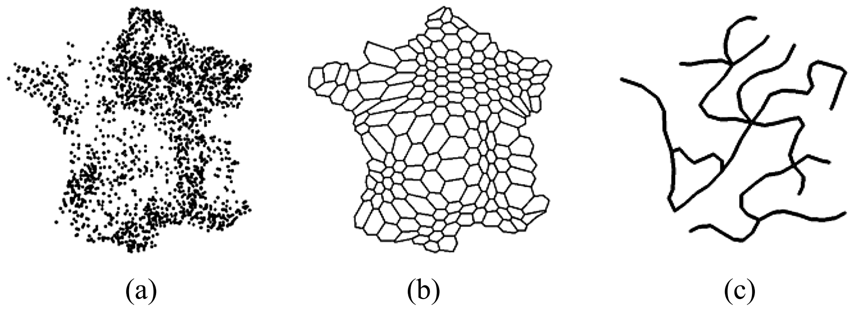


Fig. 4. Meshing of a territory. (a) Sampling of the demand (1000 dots) on a territory. (b) Adapted honeycomb mesh representing a radio-cellular network. (c) Adapted graph of interconnected routes representing a transportation network.

The demands and territory are globally cover, the distances of customers to transportation infrastructures are minimized and the density of the network infrastructures mirrors the underlying density distribution. As well, the network topology is preserved. Then, a customer would easily find a closest facility to communicate through the underlying cell base stations, or transport himself from an origin point to a destination point, following the interconnected routes which globally cover the territory.

Using visual patterns as intermediate structures that adapt and distort according to demands has several positive aspects. It takes into account the geometric nature of transportation routing and lets the user quickly and visually evaluate solutions as they evolve. In turn, the designer adjusts optimization parameters to direct the search toward useful compromises.

4.2 Combined Clustering and Routing Problems

4.2.1 Median Cycle Problem

From our knowing, the unified clustering and routing problem has never been studied previously. The closest problem encountered, called median-cycle problem (MCP), has been investigated very recently (Labbé et al. 2004, 2005; Renaud et al. 2005). It consists of finding a ring passing through a subset of the demands, minimizing objective length (1), subject to a bound on distortion (2). Such problems arise in the design of ring shape infrastructures such as circular metro lines or motorways, or in a telecommunication context to interconnect terminals via a set of concentrators.

The UCRP is Euclidean, whereas MCP and other related problems presented in (Labbé and Laporte 1986; Volgenant and Jonker 1987; Balas 1989) are defined on graphs, cluster centers being located at request locations. Hierarchical combinations of clustering and routing are studied in location-routing problems (LRP) (Min et al. 1998), but here the hierarchical order of clustering and routing is different. Here, routes visit cluster centers, whereas in LRP routes are built on separate clusters.

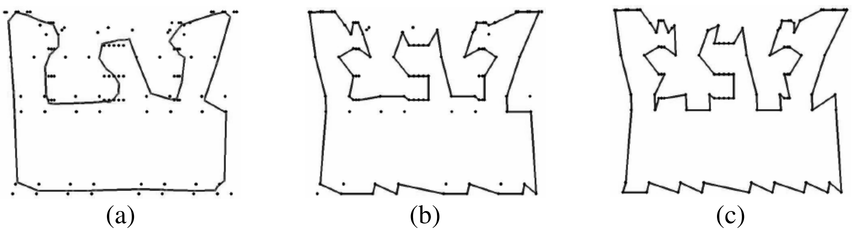


Fig. 5. Median cycle problems (MCP) using the lin105 TSPLIB problem. (a) Euclidean MCP. (b) Classical MCP. (c) Euclidean TSP.

The MCP is illustrated in Fig. 5, on the lin105 instance of TSPLIB (Reinelt 1991), considering three versions of it solved with the memetic SOM approach (Hayari 2005). In (a), is shown a Euclidean, or continuous, version of MCP, where the ring has cluster centers free to position anywhere in the plane. In (b), cluster centers are forced to be located on request locations, as it the case for the classical and discrete MCP. In (c), with a same number of centers and requests, MCP becomes a TSP.

4.2.2 Application to Bus Transportation

Here, the approach is illustrated on a real life case of combined clustering and vehicle routing (Créput and Koukam 2007a, 2007b). The goal is to locate bus-stops on roads and define the regional routes of buses to transport the 780 employees of a great enterprise in city of Belfort, east of France. Three configurations of the algorithm were built in order to solve the problem in a unified way, as well as sequentially by clustering first and routing second. We first apply a memetic SOM configuration to address the unified clustering and routing problem, thus in a unified way. Then, we apply two other configurations to tackle the problem sequentially. One addresses a k-median problem while the other addresses a VRP, using as input the obtained k-median cluster centers and their assigned quantities.

The real life case problem consists of a set of 780 employees dispatched over a geographic area of 73 × 51 km around the towns of Belfort and Montbeliard in the East of France. Their locations are shown by dots in Fig. 6(a), on a simulator written in Java, using a geographic information system. Each request represents a worker with quantity of one unit. Routes share a single common arrival point fixed at the enterprise location in city of Belfort. This common and fixed arrival point has some importance here, since it participates to transmit the SOM neighborhood influence. Each vehicle has a capacity of 45 units and a speed of 50 km / h.

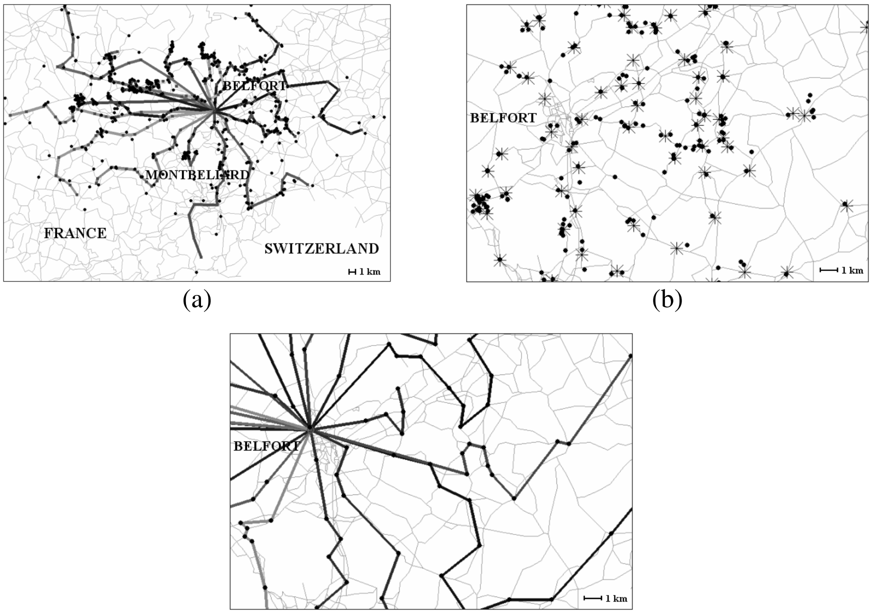


Fig. 6. Clustering and routing for the transportation of 780 customers (dots) of a great enterprise. (a) Unified clustering and routing problem solution. (b) Clustering k-median problem first (crosses are cluster centers). (c) Capacitated vehicle routing problem second (routes pass among cluster centers).

In Fig. 6(a-c), are shown visual patterns of solutions for the three types of memetic SOM applications. A transport mesh obtained for the unified clustering and routing problem is shown in Fig. 6(a), juxtaposed over requests and underlying roads. It illustrates the visual shape of a typical solution, where bus stops, located on roads, reflect demand distribution and constitute routes. In Fig. 6(b-c), a zoom is performed on the right side of the area to illustrate the two main steps for solving the problem sequentially. Fig. 6(b) presents cluster centers, symbolized by crosses, obtained for the k-median problem. **Fig. 6**(c) presents the VRP solution obtained subsequently, with routes exactly passing by the crosses. The numerical results obtained for the problem indicate that the (simple) unified approach competes with the more complex sequential method, yielding diverse non dominated solutions.

4.3 Classical Vehicle Routing Problems

4.3.1 Traveling Salesman Problem

In the literature, many applications of neural networks have addressed the traveling salesman problem. For more information on this subject, we refer the reader to the extensive survey of (Cochrane and Beasley 2003).

Here, we evaluate the memetic SOM performance on the Euclidean TSP. We compare it against the Co-Adaptive Net of (Cochrane and Beasley 2003), which is

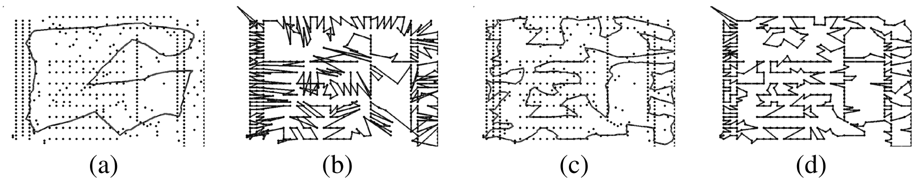

(a) (b) (c) (d)

Fig. 7. Traveling salesman problem using the pcb442 instance

considered at the date of writing as the best performing neural network application to the TSP. Experiments are conducted on large size TSPLIB instances from 1000 to up 85900 cities. Mainly, numerical results tend to show that memetic SOM competes with the Co-Adaptive Net, with respect to solution quality and/or computation time.

The memetic SOM principle is illustrated in Fig. 7(a-d) on the pcb442 instance of TSPLIB (Reinelt 1991). Two consecutive pictures respectively show the ring as it is distorted by a SOM operator (a, c), followed immediately by the ring as modified by the mapping operator (b, d), at a given generation. In (a-b) is shown the network at the first generation and in (c-d) at the middle of construction phase. While large neighborhood allows for solution diversification at the beginning of the simulation, decreasing neighborhoods allows for intensification (local improvements) at the final steps of the simulation.

Results are illustrated in Fig. 8, showing the performance of the memetic SOM for each TSPLIB problems of size larger than 1000 cities, with up to 85900 cities. Here, performance is the percentage deviation (%PDM) from optimum of the mean solution value over 10 runs. From the graphic, we can conclude that the memetic SOM yields better quality results on average than the Co-Adaptive Net. The memetic SOM yields 8.83 % of average deviation for the 33 test problems, whereas Co-Adaptive Net yields 9.49 %.

The experiments were conducted within similar (estimated) computation time. The approaches however scale differently. Co-Adaptive Net has a O(n^2.log(n)) time complexity, whereas memetic SOM a O(n^2) time complexity, n being the city number.

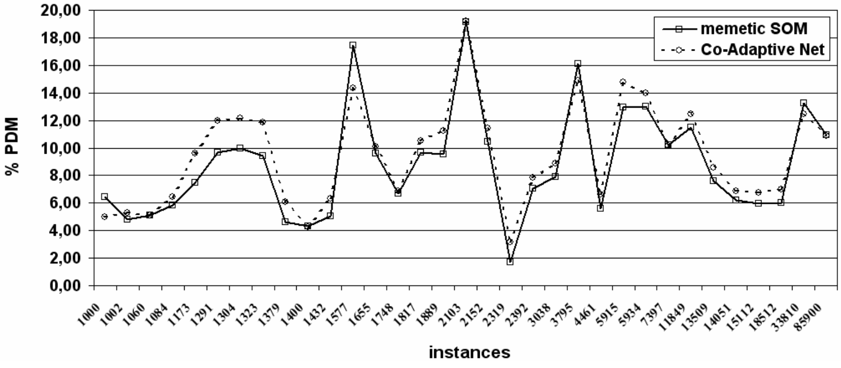


Fig. 8. Percentage deviation to optimum of the mean solution (over 10 runs) for the 33 great size TSPLIB problems with up to 85900 cities

Then, since the gain on computation time comes from largest instances, gain on solution quality is provided by almost all other instances. Considering the very large instance of size 85900, quality solutions are very similar.

However, neural networks do not compete with state of the art powerful heuristics of Operations Research for the TSP. But this domain benefits from the considerable effort spent over more than thirty years, still lacking in the neural community to the TSP. To be competitive with the best performing Lin and Kernighan heuristic implemented recently by (Helsgaun 2000), solution quality produced by neural networks, as well as computation time, would have to be improved both by a factor ten.

4.3.2 Vehicle Routing Problem

Extending SOM to the more complex vehicle routing problem remains a difficult task. Few works were carried out trying to extend SOM, or elastic nets, to the VRP. As far as we know, the most recent approaches are (Ghaziri 1996; Gomes and Von Zuben 2002; Matsuyama 1991; Modares et al. 1999; Schumann and Retzko 1995; Schwardt and Dethloff 2005; Vakhutinsky and Golden 1994). They are generally based on a complex modification of the internal learning law, altered by problem dependant penalties. Here, the SOM execution interleaves with other processes, or operators, following the evolutionary method. Then, operators can be designed independently and then combined.

Example of local improvement moves performed during the improvement optimization phase are visualized in Fig. **9**(a-d) on the clustered instances c11-14, from the publicly available Christofides, Mingozzi, and Toth (CMT) test problems (Christofides et al. 1979). The two instances on the right are time duration versions of the two on the left, hence they need more vehicles.

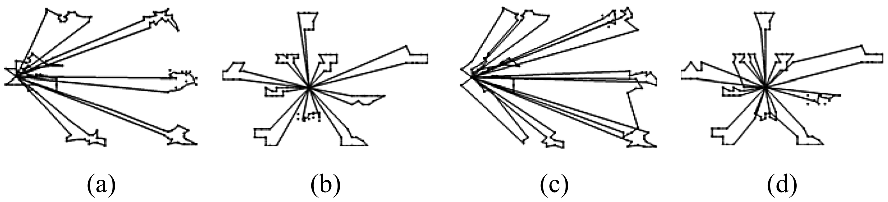


Fig. 9. VRP using c11 (a) and c12 (b) instances. VRP with time duration constraint using c13 (c) and c14 (d) CMT instances.

Evaluation of the proposed approach was performed against neural networks and Operations Research heuristics (Créput et al. 2007b). In the former case, we compared memetic SOM to the three representative approaches of (Ghaziri 1996; Modares et al. 1999; Schwardt and Dethloff 2005). Only these authors have made significant use of the CMT test problems. Other approaches typically used just a few and specific test problems and are hard to evaluate. We evaluated the percentage deviation (%PDM) to best known solution of the mean solution value over 10 runs, on CMT instances. Memetic SOM yields a gain of accuracy, from roughly 5 % for previous SOM based applications, to 3.3 % for short running times, and 1.20 % for long running times.

Considering Operations Research heuristics, we used the large test problems of (Golden et al. 1998), with up to 483 requests, and compared the approach to the Active Guided Evolution Strategy (AGES) (Mester and Bräysy 2005), Granular Tabu

Search (GTS) (Toth and Vigo 2003), Unified Tabu Search Algorithm (UTSA) (Cordeau et al. 2001) and Very Large Neighborhood Search (VLNS) (Ergun et al. 2003), which from our view point cover the range of heuristic performances. For example, as stated in (Cordeau et al. 2005), the AGES approach, which is considered complicated, has displayed the best performance according to the literature. The UTSA and VLNS are considered more simple and flexible but less performing, whereas GTS exhibits an intermediate tradeoff between solution quality and computation time.

Results are illustrated graphically in Fig. 10, showing %PDM for each test problem. We used two configurations of the memetic SOM "medium-fast" and "medium-long", for adjusting computation time at intermediate levels according to the compared approaches.

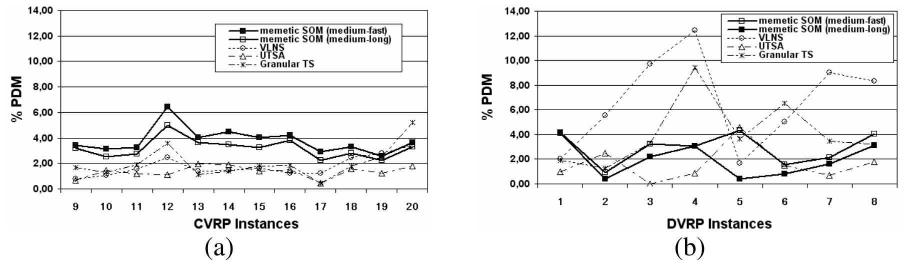


Fig. 10. Comparison with Operations Research heuristics on Golden et al. (1998) large size instances.

Memetic SOM reduces the gap from neural networks to classical heuristics. In Fig. 10(a), memetic SOM appears less accurate than other approaches. The instances have no time duration constraint (CVRP). In Fig. 10(b), memetic SOM appears more accurate. In that case, the instances have time duration constraint (DVRP). Average deviation is closed to UTSA deviation. Quality solution is better than for GTS and VLNS. Computation times were comparable to UTSA and lesser than for VLNS.

On the 20 test cases, Memetic SOM yields 2.70 % of average deviation in 39 minutes, whereas UTSA yields 1.45 % in 51 minutes. Minutes are normalized to our AMD Athlon 2000MHz computer. On average, the proposed memetic SOM performs better than VLNS, considering both quality solution and computation time. It yields 3.45 % of average deviation in 10 minutes, whereas VLNS 3,76 % in 22 minutes. For such instances, GTS computes very quickly yielding 2.87 % in 0.79 minutes. Referring to (Cordeau et al. 2005), the very powerful AGES yields a deviation of 0 % to best known value in 66 minutes, and 0.93 % in 0.58 minutes.

4.3.2 Vehicle Routing Problem with Time Windows

Self-Organizing Map has been extended to address the vehicle routing problem (VRP) but, as far as we know, it has not been applied yet to the vehicle routing problem with time windows (VRPTW). Here, a memetic SOM application allows to generate solutions to a clustering version of the classical VRPTW (Créput et al. 2007). The number of vehicles is an input and the generated solutions present walking distance from customers to their closest bus stops. Whereas, travel time are shorter. Solutions for the classical VRPTW are derived from such clustered solutions.

This is illustrated on Solomon's standard test problems (Solomon 1987). Problems are 100-customer Euclidean problems. The tests are embedded into a 100 km × 100 km geographic area, using vehicle speed of 60 km/h, time-windows being given in minutes. Visual patterns in Fig. 11(a-c) illustrate, on the rc201 test case, the different steps to generate a classical VRPTW solution from an obtained solution with non null distortion. The deformable pattern generated by the algorithm is shown in (a). An intermediate result obtained by removing empty clusters is shown in (b). Then, a VRPTW solution, as drawn in (c) is derived by projecting each cluster center to the location of its single assigned request.

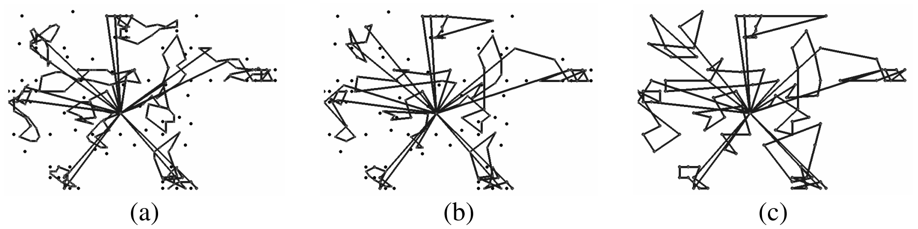


Fig. 11. Clustering Vehicle Routing Problem with Time Windows, using the rc201 Solomon test case. (a) Obtained solution. (b) Same solution after removing empty clusters. (c) A classical VRPTW solution obtained after projecting cluster centers to their (single) assigned requests.

Since SOM application to the VRPTW is new, and to allow further comparisons with memetic SOM, the Table 1 reports the average results on all the 56 Solomon instances, which are divided into six problem sets. Results are given for a single run, performed in roughly 5 minutes on average on our AMD Athlon 2000 MHz computer. The number of routes was set to the one of the best solution reported in the literature, for each problem. The first column indicates the instance name and the number of vehicles. Second column presents the best known length value. Then, the number of satisfied requests, the total route length and the average distortion (*distortion / n)* are given in columns "sat", "length" and "avg. dist.", respectively for the obtained clustering VRPTW solutions, and the derived standard VRPTW solutions. Percentage deviation to the best-known value is given within parenthesis.

The approach performs the best on clustered instances of classes C1 and C2, and mixed uniform/clustered instances of set RC1. Whereas, quality solution slightly decreases with uniformly distributed instances of set R1. It particularly diminishes with the problems which have a large horizon and need a small number of vehicles, of classes R2 and RC2.

Considering clustering VRPTW results, lengths that are obtained are lower than the best known lengths for the classical VRPTW, while average distortion is maintained within a narrow interval of less than 2 km by customer. We can appreciate visually on Fig. 11, how bus stop assignment takes place. Assignment of a request to its closest cluster center is an important characteristic of the solutions generated, since a customer would like to walk specifically to the closest bus stop, and since otherwise, finding the right assignment would become a difficult problem by itself. We think that the approach leads to a new way of thinking about the VRPTW by generating underlying patterns that dispatch through the requests, letting some place for dynamic adaptation to local modifications.

Table 1. Results for the six Solomon classes with 56 problems

	Best known	Clustering VRPTW solution			Derived VRPTW solution		
Problem set – vehicle number	length	sat	length	avg. dist.	sat	length	avg. dist.
C1–10	826.7	101	778.41 (–5.84%)	1.25	100.56	854.08 (+3.32%)	0.0
R1–11.92	1209.89	101	1109.58 (–8.52%)	2.07	99.83	1311.84 (+8.49%)	0.0
RC1–11.50	1384.16	100.75	1260.05 (–8.64%)	1.82	97.63	1424.22 (+3.45%)	0.0
C2–3	587.69	101	482.89 (–17.84%)	1.78	98.88	606.43 (+3.19%)	0.0
R2–2.75	867.54	100.83	1067.35 (+10.35%)	1.54	99.17	1187.97 (+23.88%)	0.0
RC2–3.25	1119.35	100	1340.46 (+18.05%)	1.35	99.38	1428.92 (+26.8%)	0.0

As usual with neural networks, when applied to vehicle routing problems, the approach is not yet competitive with regards to the complex and powerful Operations Research heuristics, specifically dedicated for the VRPTW. Such approaches are presented in the large survey of (Bräysy et al. 2004). For example, the tabu search adaptive memory of (Taillard et al. 1997) produces a solution of less than 1 % deviation to the best known within roughly less than 10 minutes by run, using a Sun Sparc 10 workstation. We are far from obtaining the same accuracy and execution time, except on clustered instances of class C. But, simplicity (easy to understand) and flexibility (easy to extend) are key points that are addressed, using neural networks combined in evolutionary algorithms. As well, intrinsic parallelism is an implicit advantage.

5 Conclusion

By incorporating SOM into an evolutionary algorithm, the approach extends and improves SOM based neural network applications. Operators can be interpreted as performing parallel and massive insertions, simulating spatially distributed agents which interact, having localized and limited abilities. The evolutionary framework helps directing the search toward problem goals.

This massive and natural parallelism at different levels differentiates the approach from classical Operations Research heuristics which operate on graphs (sequentially) and are often considered complex and difficult to implement. Since the communication times at the level of selection is relatively small, the long running times of independent SOM processes favor parallel execution of the method.

The approach has been applied to many spatially distributed transportation optimization problems, thus illustrating its flexibility and accuracy in regard to classical heuristics. Applications to dynamic and stochastic problems are questions to be addressed in the future. Exploiting the natural parallelism of the approach for multi-processor implantation is also a key point to address in further work.

References

Arora, S.: Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems. Journal of the ACM 45(5), 753–782 (1998)

Balas, E.: The prize collecting traveling salesman problem. Networks 19, 621–636 (1989)

Boese, K.D., Kahng, A.B., Muddu, S.: Adaptive Multi-Start Technique for Combinatorial Global Optimization. Journal of Operations Research Letters 16, 101–113 (1994)

Bräysy, O., Dullaert, W., Gendreau, M.: Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows. Journal of Heuristics 10(6), 587–611 (2004)

Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., et al. (eds.) Combinatorial Optimization, pp. 315–338. Wiley, Chichester (1979)

Cochrane, E.M., Beasley, J.E.: The co-adaptive neural network approach to the Euclidean Travelling Salesman Problem. Neural Networks 16, 1499–1525 (2003)

Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 52, 928–936 (2001)

Cordeau, J.F., et al.: New Heuristics for the Vehicle Routing Problem. In: Langevin, A., Riopel, D. (eds.) Logistics Systems: Design and Optimization, pp. 279–297. Springer, New York (2005)

Créput, J.C., Koukam, A.: Local search study of honeycomb clustering problem for cellular planning. International Journal of Mobile Network Design and Innovation 1(2), 153–160 (2006)

Créput, J.C., Koukam, A.: Interactive Meshing for the Design and Optimization of Bus Transportation Networks. Journal of Transportation Engineering, ASCE (in press, 2007a)

Créput, J.C., Koukam, A.: Transport Clustering and Routing as a Visual Meshing Process. Journal of Information and Optimization Sciences, Taru Publications (in press, 2007b)

Créput, J.C., Koukam, A., Hajjam, A.: Self-Organizing Maps in Evolutionary Approach for the Vehicle Routing Problem with Time Windows. International Journal of Computer Science and Network Security 7(1), 103–110 (2007)

Créput, J.C., Lissajoux, T., Koukam, A.: A Connexionnist Approach to the Hexagonal Mesh Generation. In: 16th IMACS World Congress, Lausanne (2000)

Créput, J.C., et al.: Automatic Mesh Generation for Mobile Network Dimensioning using Evolutionary Approach. IEEE Transactions on Evolutionary Computation 9(1), 18–30 (2005)

Ergun, Ö., Orlin, J.B., Steele-Feldman, A.: Creating very large scale neighborhoods out of smaller ones by compounding moves: a study on the vehicle routing problem. MIT Sloan Working Paper No. 4393-02, USA (2003)

Gambardella, L.M., Taillard, E., Agazzi, G.: MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 63–76. McGraw-Hill, UK (1999)

Ghaziri, H.: Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem. In: Osman, I.H., Kelly, J.P. (eds.) Meta-Heuristics: Theory & Applications, pp. 651–660. Kluwer, Boston (1996)

Golden, B.L., et al.: Metaheuristics in vehicle routing. In: Crainic, T.G., Laporte, G. (eds.) Fleet Management and Logistics, pp. 33–56. Kluwer, Boston (1998)

Gomes, L.C.T., Von Zuben, F.J.A.: Vehicle Routing Based on Self-Organization with and without Fuzzy Inference. In: Proc. of the IEEE International Conference on Fuzzy Systems, vol. 2, pp. 1310–1315 (2002)

Hayari, N., Créput, J.C., Koukam, A.: A Neural Evolutionary Algorithm for Geometric Median Cycle Problem. In: European Simulation and Modeling Conference, ESM 2005, EUROSIS, Porto, Portugal (2005)

Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research 126, 106–130 (2000)

Kohonen, T.: Self-Organization Maps and associative memory, 3rd edn. Springer, Berlin (2001)

Labbé, M., Laporte, G.: Maximizing user convenience and postal service efficiency in post box location. Belgian Journal of Operations Research, Statistics and Computer Science 26, 21–35 (1986)

Labbé, M., et al.: The Ring Star Problem: Polyhedral Analysis and Exact Algorithm. Networks 43, 177–189 (2004)

Labbé, M., Rodríguez Martín, I., Salazar González, J.J.: Locating Median Cycles in Networks. European Journal of Operational Research 160, 457–470 (2005)

Matsuyama, Y.: Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In: Proc. of the International Joint Conference on Neural Networks, Seatle, WA, pp. 385–390 (1991)

Mester, D., Bräysy, O.: Active Guided Evolution Strategies for Large Scale Vehicle Routing Problems with Time Windows. Computers & Operations Research 32, 1593–1614 (2005)

Min, H., Jayaraman, V., Srivastava, R.: Combined location-routing problems: A synthesis and future research directions. European Journal of Operational Research 108, 1–15 (1998)

Modares, A., Somhom, S., Enkawa, T.: A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. International Transactions in Operational Research 6, 591–606 (1999)

Moscato, P.: Memetic Algorithms: A Short Introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, McGraw-Hill, New York (1999)

Mühlenbein, H.: Evolution in Time and Space – The Parallel Genetic Algorithm. In: Rawlins, G. (ed.) Foundations of Genetic Algorithms, Morgan Kaufmann, Los Altos, CA (1991)

Reinelt, G.: TSPLIB-A traveling salesman problem library. ORSA Journal on Computing 3, 376–384 (1991)

Renaud, J., Boctor, F., Laporte, G.: Efficient heuristics for median cycle problems. Journal of the Operational Research Society 55, 179–186 (2004)

Schumann, M., Retzko, R.: Self-organizing maps for vehicle routing problems minimizing an explicit cost function. In: Proc. of the International Conference on Artificial Neural Networks, Paris, pp. 401–406 (1995)

Schwardt, M., Dethloff, J.: Solving a continuous location-routing problem by use of a self-organizing map. Int J. Physical Distribution & Logistics Management 35(6), 390–408 (2005)

Solomon, M.M.: Algorithms for the vehicle routing, and scheduling problems with time window constrains. Operations Research 35, 254–264 (1987)

Taillard, E.D., et al.: A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation science 31, 170–186 (1997)

Toth, P., Vigo, D.: The granular tabu search and its application to the vehicle routing problem. INFORMS Journal on Computing 15, 333–348 (2003)

Vakhutinsky, A.I., Golden, B.L.: Solving vehicle routing problems using elastic net. In: IEEE International Conference on Neural Network, pp. 4535–4540 (1994)

Volgenant, T., Jonker, R.: On some generalizations of the traveling salesman problem. Journal of the Operational Research Society 38, 1073–1079 (1987)

Statistical Forecasting of Indian Summer Monsoon Rainfall: An Enduring Challenge

Shivam Tripathi and Rao S. Govindaraju

School of Civil Engineering, Purdue University, West Lafayette, IN 47906, USA

1 Introduction

Forecasting All India Summer Monsoon Rainfall (AISMR), one or more seasons in advance, has been an elusive goal for hydrologists, meteorologists, and astrologers alike. In spite of advances in data collection facilities, improvements in computational capabilities, and progress in our understanding of the physics of the monsoon system, our ability to forecast AISMR has remained more or less unchanged in past several decades. On one hand, physically based *numerical prediction models* that are considered a panacea for daily weather forecasting have not evolved to a stage where they can realistically predict or even simulate annual variations in Indian monsoon. On the other hand, *statistical models* that have traditionally been used for making operational forecasts have failed in forecasting extreme monsoon rainfall years. It has been suggested that, in future, physically based models may improve to an extent where they can produce useful forecasts. However, until then, it would be prudent to develop statistical forecast models using state-of-the-art soft-computing techniques.

Statistical forecasting of AISMR has a long, venerable, and vulnerable history. The first ever scientific forecast of AISMR was made by H. F. Blanford for year 1878, after the great famine of 1877 that took a heavy toll on human lives. During initial years, forecasts issued were subjective and met limited success. Later, in early twentieth century the forecast skill of AISMR improved significantly mainly due to initiatives taken by Sir Gilbert Thomas Walker. Sir Walker, who was then the Director General of India Meteorology Department (IMD), collected and analyzed vast amounts of weather data from India and abroad. He discovered Southern Oscillation (SO), a major atmosphere phenomenon over tropical Pacific Ocean that was later linked to El Niño (Bjerknes 1969). The discovery of link between AISMR and El Niño / SO (ENSO) led to rapid development in statistical forecasting models. However, initial encouraging performance did not last long because the link between AISMR and ENSO started weakening in the 1980s (Kumar et al. 1999).

The work of Sir Walker encouraged researchers to find other atmospheric and oceanic variables over different parts of the world that can be used as potential predictors for AISMR. Some of the important predictors that came out these endeavors are : (i) global sea surface temperature (Sahai et al. 2003; Pai and Rajeevan 2006), (ii) Himalayan and Eurasian snow cover (Fasullo 2004; Kripalani et al. 2003), (iii) atmospheric circulation patterns like position of 500 hPa ridge over India (Prasad and Singh 1992), and wind anomalies (Bhalme et al. 1987; Gadgil et al. 2004), (iv) land surface conditions over Northern Hemisphere (Rajeevan 2002; Robock et al. 2003), and (v) the previous values of AISMR series (Kishtawal et al. 2003; Iyengar and Raghu Kanth

2005). However, recent evidences suggest that the relationship between most of these predictors and AISMR is not stationary but varies on decadal to interdecadal time scales (Rajeevan et al. 2007; Gadgil et al. 2002; Kumar et al. 1999; Clark et al. 2000). Further, studies indicate that some of the predictors have even lost their importance over the course of time.

The facts mentioned in the previous paragraphs indicate that the successful forecasting of AISMR needs a statistical model that not only updates the relationship between the predictors and the predictand in light of new data but also dynamically selects the appropriate set of predictors for making forecasts. However, to date, little has been done to address this problem. The current strategy is to constantly monitor the performance of the model, and subjectively change the structure of the model, its inputs, and even training period in case of model failure (Rajeevan et al. 2007). Obviously, updating the model in this way leaves no possibility for model validation, and consequently no confidence can be assigned to the model predictions.

This study seeks to address the above mentioned issue by developing a dynamic forecasting model under Bayesian framework. The model not only updates the relationship between the predictors and the predictand as and when new data become available, but also dynamically selects the appropriate set of predictors. Further, we also suggest a way for assessing forecasting skill of the model. The model has only one adjustable parameter that helps in reducing the level of subjectivity in making forecasts. The efficacy of the model is assessed by evaluating its performance in predicting AISMR from 1951 to 2005, using global sea surface temperature (SST) dataset as the only predictor.

The proposed model, in its first step, uses *probabilistic principal component analysis* (PPCA) (Roweis 1998; Tipping and Bishop 1999) in combination with *Bayesian model selection* (Minka 2001) to reduce the dimensionality of the SST data. In the second step, the model uses a sparse Bayesian learning algorithm (Tipping 2001) to select the appropriate set of predictors, and to learn the relationship between selected predictors and AISMR, the predictand. The sparse Bayesian model is known as *relevance vector machine* (RVM) owing to its capability to identify most relevant patterns or predictors for making forecast. The parameters of the RVM model are estimated using a sequential learning algorithm. It is to be emphasized that the RVM can be used to learn non-linear relationships between predictors and predictand, however for the sake of simplicity and interpretability of results, only linear relationships are investigated in this work.

The remainder of this paper is structured as follows. The mathematical formulations of PPCA, Bayesian model selection, and RVM are presented in Sect. 2. Following this, data used in the study are described in Sect. 3. Details of the proposed methodology are given in Sec. 4, and the results obtained are presented and discussed in Sect. 5. Finally, a set of concluding remarks and our recommendations are provided in Sect. 6.

2 Mathematical Formulation

This section presents the mathematical formulation for probabilistic principal component analysis (PPCA), Bayesian model selection, and relevance vector machine (RVM) in the context of forecasting AISMR.

2.1 Probabilistic Principal Component Analysis (PPCA)

Principal component analysis (PCA) and its variants like empirical orthogonal teleconnections (van den Dool et al. 2000), nonlinear principal component analysis (Monahan 2001), rotational techniques (Horel 1981), space-time principal component (Vautard et al. 1996) and closely related methods such as canonical component analysis (Shabbar and Barnston 1996), are arguably the most commonly used methods for data compression, data reconstruction, and developing prediction models in hydrologic and meteorologic literature. PCA has also been successfully used as a feature extraction method in developing forecast models for AISMR (Cannon and McKendry 1999; Rajeevan et al. 2000; Pai and Rajeevan 2006).

However, a serious limitation of conventional PCA, when applied to inherently noisy hydro-meteorologic data, is the absence of an associated probabilistic model for the observed data. Tipping and Bishop (1999) proposed a probabilistic approach to PCA that can overcome this limitation. Besides this, probabilistic principal component analysis (PPCA) offers a number of other advantages, including a principled way for handling missing values in the data, and an objective way to decide the optimum number of principal components.

Given a p dimensional observed data variable $\boldsymbol{x}$ $\left(\boldsymbol{x} \in \Re^{p}\right)$, the goal of PPCA is to find a q dimensional principal variable $\boldsymbol{z}$ $\left(\boldsymbol{z} \in \Re^{q}\right)$, such that the number of principal components q is less than p. *Assuming q is known*, the reconstruction of data variable from principal variable is given by[1]

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon} \tag{1}$$

where, $\boldsymbol{\varepsilon}$ is a p dimensional Gaussian noise, with zero mean and covariance $\sigma^2 \boldsymbol{I}_p$, and $\boldsymbol{\mu}$ is a p dimensional vector. $\boldsymbol{W}$ is a $p \times q$ transformation matrix whose columns span a linear subspace within the p dimensional observed data variable space.

Due to the assumption of Gaussian noise, the distribution of observed data variable $\boldsymbol{x}$ conditioned on $\boldsymbol{z}$ is given by

$$p(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_p\right) \tag{2}$$

If we assign zero mean, unit covariance Gaussian prior distribution to the principal vector $\boldsymbol{z}$, i.e.

$$p(\boldsymbol{z}) = \mathcal{N}\left(0, \boldsymbol{I}_q\right) \tag{3}$$

then the marginal distribution of the observed variable $p(\boldsymbol{x})$ also becomes Gaussian and is given by

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C}) \tag{4}$$

where, the covariance matrix $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} + \sigma^2 \boldsymbol{I}_p$.

[1] All vectors are column vectors. A d dimensional identity matrix is represented by $\boldsymbol{I}_d$.

Now consider a dataset with N observed data points i.e. $\boldsymbol{X} = [\boldsymbol{x}_n], n = 1,\ldots,N$. The log likelihood of the observed dataset, given the model (Eq. 1) is

$$\mathcal{L}(\boldsymbol{\mu},\boldsymbol{W},\sigma^2) = \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n) = -\frac{N}{2}\left\{p\ln(2\pi) + \ln(|\boldsymbol{C}|) + \mathrm{tr}(\boldsymbol{C}^{-1}\boldsymbol{S})\right\} \tag{5}$$

where $\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^{\mathrm{T}}$ is the data covariance matrix.

The model parameters $\boldsymbol{W}$, $\boldsymbol{\mu}$, and σ can be estimated by maximizing the likelihood function (Eq. 5) corresponding to these parameters. Tipping and Bishop (1999) showed that the maximum likelihood solution corresponds to principal component analysis of the dataset $\boldsymbol{X}$. The principal directions in $\boldsymbol{X}$ are contained in the columns of $\boldsymbol{W}$, while the principal components z can be calculated by Bayes' rule as

$$p(z \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{M}^{-1}\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{\mu}), \sigma^{-2}\boldsymbol{M}\right) \tag{6}$$

where $\boldsymbol{M} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} + \sigma^2\boldsymbol{I}_q$.

The maximum likelihood estimate of the parameters can be either obtained explicitly by using analytical expressions or by using expectation maximization (EM) algorithm (Roweis 1998; Tipping and Bishop 1999). For very high dimensional datasets like sea surface temperature (SST), EM algorithm has significant computational advantages and is therefore used in this study.

2.2 Bayesian Model Selection

In the above discussion of PPCA we have assumed that the dimensionality q of the principal vector z is known. In practice, subjective criteria like retaining certain percentage of the variance in the data or the point where the eigen value spectrum takes an elbow turn, are often used. However, these arbitrary thresholds cannot determine the true dimensionality of the principal vector and may result in models that are significantly different among different users.

An important advantage offered by the probabilistic interpretation of PCA is that an objective approach of Bayesian model selection can be used to determine the number of PCs. In Bayesian approach to model selection, the best model is the one which has maximum marginal likelihood over all possible values of model parameters.

In PPCA, for a given value of q, the marginal likelihood $p(\boldsymbol{x} \mid q)$ can be calculated by integrating out the model parameters $\boldsymbol{W}$, $\boldsymbol{\mu}$, and σ. To do this, suitable prior probabilities are assigned to model parameters. Minka (2001) proposed non informative prior distribution for $\boldsymbol{\mu}$ and conjugate priors for $\boldsymbol{W}$ and σ. Using these priors, an analytical expression for $p(\boldsymbol{x} \mid q)$ was derived. However, the estimate of marginal likelihood involves an integral over the Stiefel manifold that is difficult to compute exactly. To provide a practical solution, Minka (2001) applied Laplace's method that aims to find a Gaussian approximation of the marginal likelihood.

The optimal number of principal component $\hat{q}$ can then be obtained by using Bayesian model selection rule as

$$\hat{q} = \text{argmax}_q \left[p(\boldsymbol{x} \mid q) \right], \quad 1 \le q \le p \tag{7}$$

2.3 Relevance Vector Machine (RVM)

The principal components obtained from PPCA (Sect. 2.1) provide a compact representation of the raw data. However, out of the many PCs extracted from the data, only few are expected to be useful for a forecasting model. The next crucial step is therefore to select the relevant PCs by using a feature selection algorithm. These selected PCs form the predictor set for the model.

There are many feature selection algorithms available in the literature and most of them are restricted to static datasets. However, Indian summer monsoon is a time evolving system, where the predictors as well as their relationship with AISMR constantly changes with time. Therefore a *dynamic feature selection* algorithm is required. To this end, RVM model with sequential learning algorithm is adopted in this study. RVM algorithm was proposed by Tipping (2001). It has excellent generalization properties and has been successfully used in many real world applications including hydrology (Khalil et al. 2005, 2006). However, in this study RVM was selected because of the following two properties: (i) automatic relevance determination that selects the most relevant predictors for making forecasts, and (ii) sequential learning that allows model to progressively update itself as more and more data becomes available. In passing, it is worth mentioning that, to our knowledge, RVM has not been used in the context of dynamic feature selection.

Detailed mathematical formulation of RVM is available in Scholkopf and Smola (2001), Tipping (2001), and Bishop (2006). Here we provide a brief overview of RVM algorithm.

The PCs $\boldsymbol{Z} = [z_n], n = 1,\ldots,N$ extracted from PPCA form the input to the RVM. The n^{th} member of the input set $z_n \{ z_n \in \Re^q, z_n = [z_{n1}, z_{n2},\ldots, z_{nq}]^{\text{T}} \}$ constitutes the potential pool of predictors for making forecast at step n, that corresponds to time t. Further, the target value at step n is given by $y_n \{ y_n \in \Re; y = [y_1, y_2,\ldots, y_N]^T \}$, and it corresponds to standardized values of AISMR at time $t + \Delta t$, where Δt is the lead time of the forecast.

In linear RVM, the target value y_n is approximated as

$$y_n = \sum_{i=1}^{q} w_i z_{ni} + w_0 + \varepsilon_n \tag{8}$$

where, $\boldsymbol{w} = \left[w_0, w_1,\ldots, w_q \right]^{\text{T}}$ is a weight vector, and $\varepsilon = \left[\varepsilon_1,\ldots,\varepsilon_N \right]^{\text{T}}$ is independent zero mean Gaussian noise with variance σ^2. In this setting, likelihood of the observed data can be written as

$$p\left(\boldsymbol{y} \mid \boldsymbol{w}, \sigma^2\right) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N exp\left\{-\frac{1}{2\sigma^2}\left\|\boldsymbol{y} - \boldsymbol{\Phi w}\right\|^2\right\} \tag{9}$$

where $\boldsymbol{\Phi}$ is defined as

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & z_{11} & \cdots & z_{1q} \\ 1 & z_{21} & \cdots & z_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_{N1} & \cdots & z_{Nq} \end{bmatrix} \tag{10}$$

Under a Bayesian perspective, weights $\boldsymbol{w}$ can be estimated by first assigning prior distribution to it and then estimating its posterior distribution using likelihood of the observed data. In this study, following Tipping (2001) a zero mean Gaussian prior of the form given by Eq. 11 is used.

$$p\left(\boldsymbol{w} \mid \boldsymbol{\alpha}\right) = \prod_{i=0}^{q} \mathcal{N}\left(0, \alpha_i^{-1}\right) \tag{11}$$

In Eq. 11, $\boldsymbol{\alpha} = \left[\alpha_0, \ldots, \alpha_q\right]^T$ is a hyperparameter vector. The inverse of hyperparameter $\left(\alpha_i^{-1}\right)$ represents the importance of feature i in the model. It turns out that during the process of maximizing the likelihood of the observed data with respect to the hyperparameters, many of the hyperparameters go to infinity, and the corresponding features are removed from the model achieving sparsity in the model.

Having defined the likelihood (Eq. 9) and the prior (Eq. 11), the next step is to find the posterior distribution of the parameters. The posterior distribution of the weight vector $\boldsymbol{w}$ can be obtained analytically given hyperparameter values. However, there is no closed form equation for the posterior distribution of the hyperparameters. Nevertheless, it can be reasonably approximated by maximizing the log likelihood of the observed data with respect to hyperparameters (Tipping 2001) as given by Eq. 12

$$\mathcal{L}(\boldsymbol{\alpha}) = \ln\left(p\left(\boldsymbol{y} \mid \boldsymbol{\alpha}, \sigma^2\right)\right) = -\frac{1}{2}\left(N \ln 2\pi + \ln\left|\boldsymbol{R}\right| + \boldsymbol{y}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{y}\right) \tag{12}$$

where, $\boldsymbol{R} = \sigma^2 \boldsymbol{I}_N + \boldsymbol{\Phi A}^{-1}\boldsymbol{\Phi}^T$, and $A = diag(\alpha_0, \alpha_1, \ldots, \alpha_q)$.

The log likelihood function (Eq. 12) can be maximized by either using *type 2 maximum likelihood* (Berger 1985) or by using *sequential learning* (Li et al. 2002; Tipping and Faul 2003). Here the latter approach is used, wherein the log likelihood of observed data is maximized with respect to hyperparameter of each feature separately. The gradient of likelihood is given by

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha})}{\partial \alpha_i} = \frac{\alpha_i^{-1} S_i^{\ 2} - (Q_i^{\ 2} - S_i)}{2(\alpha_i + S_i)^2} \tag{13}$$

where Q_i^2 is known as the *quality* , and S_i is known as the *sparsity* of the i^{th} feature. They are calculated using Eq. 14 and Eq. 15, respectively.

$$Q_i = \boldsymbol{\varphi}_i^{\,T} \boldsymbol{R}_{-i}^{-1} \boldsymbol{y} \tag{14}$$

$$S_i = \boldsymbol{\varphi}_i^{\,T} \boldsymbol{R}_{-i}^{-1} \boldsymbol{\varphi}_i \tag{15}$$

In Eqs. 14 and 15, $\boldsymbol{\varphi}_i$ is the i^{th} column of $\boldsymbol{\Phi}$ (Eq. 10) and $\boldsymbol{R}_{-i}^{-1} = \boldsymbol{R}_i - \alpha_i^{-1} \boldsymbol{\varphi}_i \boldsymbol{\varphi}_i^{\,T}$. The quality term Q_i^2 measures the increment in log likelihood of the observed data $\boldsymbol{y}$ (Eq. 12) due to inclusion of feature *i* and it thus indicates the benefit of including that feature in the model. The sparsity term S_i, on the other hand, measures the decrease in log likelihood value due increase in the covariance term $\boldsymbol{R}$. It thus indicates the cost of including an irrelevant feature in the model.

In sequential learning algorithm, the model is initialized with the bias term. In each successive iteration, the 'quality' and the 'sparsity' of all the features are calculated. The feature which is not in the model and for which the value of Q_i^2 relative to the value of S_i is greatest, is included in the model. Similarly, the feature/s that are in the model but for which the value S_i is greater the value of Q_i^2 are removed from the model. The iteration terminates when no feature can be included or excluded from the model, or the improvement in log likelihood function (Eq. 12) is below a threshold value (~ machine precision).

At convergence, the algorithm yields a set of features (F) that are deemed most relevant for making predictions, along with the posterior distribution of the associated weight vectors w_i, $\forall i \in \mathsf{F}$. The algorithm also provides an estimate of hyperparameter $\alpha_i \left(\forall i \in \mathsf{F}\right)$, and the noise variance σ^2. The weight w_i and hyperparameter α_i, $\forall i \notin \mathsf{F}$, are notionally set to zero and infinity, respectively.

The distribution of predictand y^* for new set of predictors z^* is obtained as

$$p\left(y^* \mid z^*\right) = \mathcal{N}\left(\mu_{y^*}, \sigma_{y^*}^2\right) \tag{16}$$

where the mean and variance of the predicted value are, respectively,

$$\mu_{y^*} = \boldsymbol{\mu}_w^T \boldsymbol{\varphi}\left(z^*\right) \tag{17}$$

$$\sigma_{y^*}^2 = \hat{\sigma}^2 \boldsymbol{\varphi}\left(z^*\right)^{\mathrm{T}} \boldsymbol{\Sigma}_w \boldsymbol{\varphi}\left(z^*\right) \tag{18}$$

Here, vector $\boldsymbol{\varphi}\left(z^*\right) = \left[1, z^*\right]^{\mathrm{T}}$, $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ are the mean and covariance of the posterior weight distribution and $\hat{\sigma}^2$ is the estimated error variance.

3 Data Used in This Study

3.1 Predictors

It has been suggested in the literature that the seasonal climate predictability is mainly derived from slow varying *surface boundary conditions*, like SST, snowcover, vegetation, and soil moisture that influence global atmospheric circulation and thus global surface climate (Charney and Shukla 1981). Among surface boundary conditions, SST is the most popular predictor for seasonal forecast. It is the principal surface boundary condition that influences the atmospheric seasonal variability (Barnston et al. 2005), and it is the only variable for which long term consistent records are available.

SST has long been used as a predictor for AISMR. Sahai et al. (2003) and recently Pai and Rajeevan (2006) used only global SST data to develop long range forecasting model for AISMR. They reported high correlation ($\approx$ 0.8 to 0.9) between observed and predicted values during validation period. Further, there is a plethora of studies that have investigated the link between AISMR and SST in different regions of the world including (i) Pacific Ocean (Mooley and Munot 1997; Kumar et al. 2006; Krishnan and Sugi 2003), (ii) Atlantic Ocean (Srivastava et al. 2002; Goswami et al. 2006), (iii) Indian Ocean (Kucharski et al. 2006; Li et al. 2001; Clark et al. 2000), (iv) Arabian Sea (Rao and Goswami 1988; Kothawale et al. 2007), and (v) regions surrounding Australia and Indonesia (Nicholls 1983, 1995). These studies indicate that a substantial portion of the interannual variability in AISMR can be explained by SST alone and hence the potential predictors were derived only form global SST data in this study.

Monthly 1^0 resolution global SST data from 1870 onwards is available from the Hadley Centre Sea Ice and Sea Surface Temperature dataset (HadISST1) (Rayner et al. 2003). The dataset is based on interpolation of measured SST values compiled in International Comprehensive Ocean Atmosphere Data Set (ICOADS) database, and Met Office Marine Data Bank (MDB). The dataset is constructed using a reduced space optimal interpolation procedure. This dataset is updated every month and can be obtained from Hadley Centre's website *http://hadobs.metoffice.com/hadisst/*.

3.2 Predictand

The predictand used in the study is All India Summer Monsoon Rainfall (AISMR). The monthly area weighted summer monsoon rainfall data over India (Parthasarathy et al. 1994), which extends from 1871 to 2004, is extracted from Indian Institute of Tropical Meteorology, Pune, web site *http://www.tropmet.res.in*. Primary source of the data is India Meteorological Department.

4 Methodology

This section outlines the procedure involved in processing SST and rainfall data, identifying SST patterns that are good predictors for AISMR, and developing RVM model to forecast AISMR.

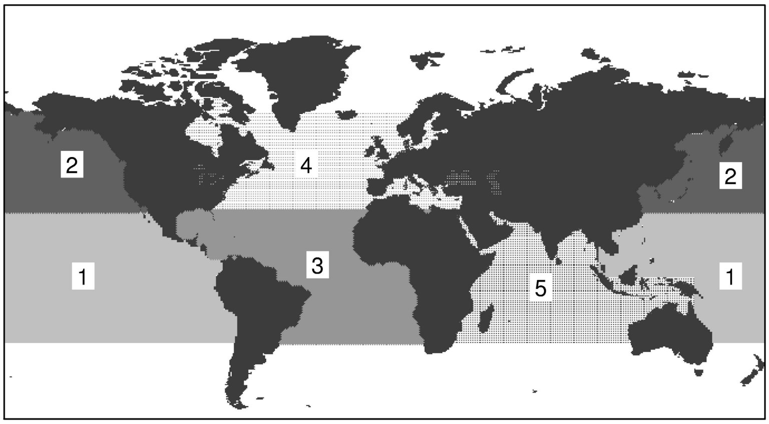


Fig. 1. Partitioning of HadISST1 data into five oceanic sectors: 1-Tropical Pacific sector (30^0S – 30^0N), 2-North Pacific sector (north of 30^0N), 3-Tropical Atlantic sector (30^0S – 30^0N), 4-North Atlantic sector (north of 30^0N), and 5-The Indian Ocean sector (north of 30^0S).

As the first step, the rainfall data were standardized (subtracted by the long term mean and divided by the standard deviation). Following Lau et al. (2002), the SST data were partitioned into five non-overlapping sectors- Tropical Pacific sector (30^0S–30^0N), North Pacific sector (north of 30^0N), Tropical Atlantic sector (30^0S–30^0N), North Atlantic sector (north of 30^0N), and Indian Ocean sector (north of 30^0S) as shown in Fig. 1. This was done because intrinsic ocean variability outside of the tropical Pacific Ocean is known to be frequently obscured by strong ENSO signal. Partitioning of data allows studying SST variability in all sectors separately.

The partitioned SST data was then used to calculate the SST *anomalies* (SSTa) and *tendencies of SST anomalies* (SSTt). SSTa at a grid point for a given season is defined as the deviation of SST value from its long term average. SSTt for a given season is defined as the change in SSTa from the previous season. SSTt values represent the evolution of SST data over time and are reported to have better predictive information than SSTa (Sahai et al. 2003). In this study, the time lag of SSTa and SSTt were varied from 1 to 12 months behind the start of monsoon month (June).

After preliminary exploratory data analysis (Tukey 1977), the rainfall, and the SST data were divided into a training set (1900-1950) and a test set (1951-2005). SST data prior to 1900 is less reliable (Smith and Reynolds 2003) and therefore not considered in the analysis. Following this, Pearson product-moment correlation coefficients between SST data (SSTa and SSTt) and rainfall data (AISMR) in training set were calculated. Potential predictors among SSTa and SSTt from different oceanic sectors were screened by imposing various thresholds (0.15 to 0.45) on the absolute value of correlation coefficients. The screening step is based on the assumption that the relevant features will exhibit some correlations on their own so that they can be segregated from the irrelevant features. Screening thus reduces the noise in the raw data, and is vital for the success of the following steps. The screened variables were then processed through probabilistic principal component analysis (PPCA) to extract principal components that preserve maximum variance in the screened data. The number of principal components to be retained for subsequent analysis was decided by Bayesian model selection method described in Sect. 2.2.

The principal components (PCs) obtained from the PPCA serve as inputs to the relevance vector machine (RVM). RVM uses automatic relevance determination to identify the most relevant features (PCs) for forecasting AISMR. It also builds a linear relationship between identified predictors and AISMR of the form given by Eq. 8. RVM, by virtue of its Bayesian formulation, progressively updates the predictor set, and its relationship with predictand. In the sequential learning algorithm adopted in this work (Sect. 2.3), the posterior distribution of model parameters at a given step become prior distribution of the parameters for the next step. Bayes' rule is then used to update the posterior distribution in light of new data.

5 Results and Discussion

Exploratory data analysis was done with a view to understand the relationship between SST data [SST anomalies (SSTa) and tendency of SST anomalies (SSTt)] and All India Summer Monsoon Rainfall (AISMR). To this end, a 30 year moving window was used to calculate the correlation between AISMR and principal components of SSTa and SSTt over different oceanic sectors for different time lags. Pearson product-moment correlation and Kendall's tau rank correlation coefficient were calculated. Typical results of the analysis are shown in Fig. 2. It is evident from the figure that the relationship between AISMR and SST is not static but changes with time. These results are in agreement with the previous studies discussed in Sect. 1. Furthermore, the results highlight the need for developing a dynamic feature selection and learning model for forecasting AISMR.

The SSTa and SSTt values in the *training data* were used to screen the set of potential predictors following the procedure outlined in Sect. 4. The screened variables were processed through probabilistic principal component analysis (PPCA). The number of principal components that were to be retained for the subsequent analysis was estimated by using Bayesian model selection method (Sect. 2.2). A typical result of this selection method is given in Fig. 3. The figure shows the marginal likelihood computed by varying the number of principal components. The optimum number corresponds to maximum value of the marginal likelihood.

The PCs obtained in the foregoing step serve as inputs to the RVM model, while standardized values of AISMR formed its output. The model was initially trained for the period 1901 to 1950. The trained model was then used to forecast the value of AISMR for 1951. After the forecast was made, the observed value of AISMR for year 1951 was used to update the RVM parameters. During an update operation, the predictors that have lost their relevance are removed form the model, new relevant predictors are added, and the relationship between existing predictors and AISMR is revised. The steps are repeated to sequentially generate forecast for the test period (1951 to 2005). The correlation between the forecasted and the observed AISMR during the test period was computed. The analysis was repeated number of times by varying the threshold for screening predictors in the range 0.15 to 0.4. The results obtained are presented as a solid line in the group of curves labeled 'C' in Fig. 4. The dashed line in the 'C-group' are the results obtained from the same analysis, but this time

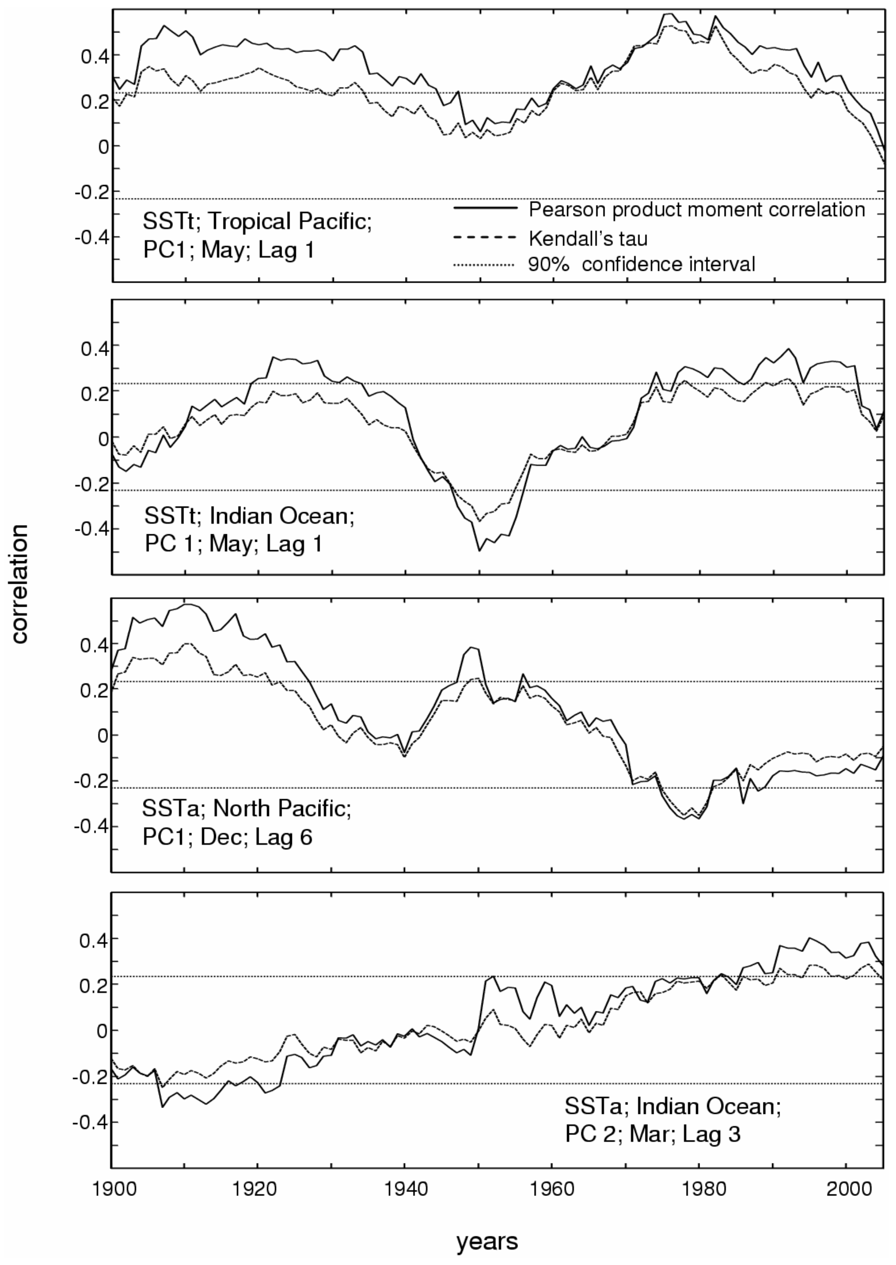


Fig. 2. Thirty year moving window correlation between AISMR and SST data. Text on each panel denotes the type of variable [SST anomaly (SSTa) / tendency of SST anomaly (SSTt)], oceanic sector, order of principal component, month of observation, and time lag (in months) from the start of monsoon season (June).

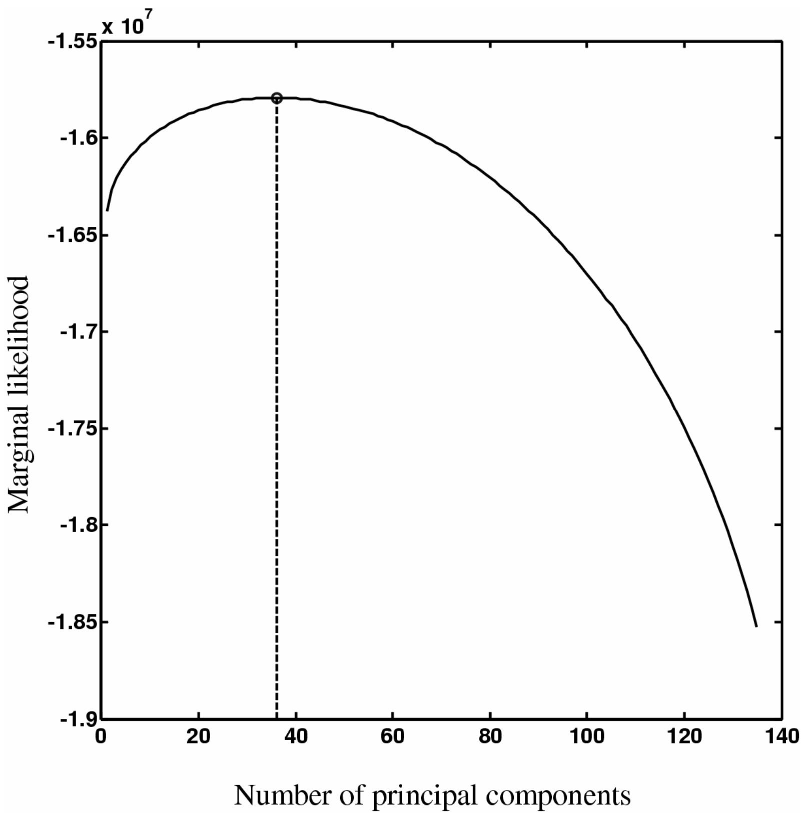


Fig. 3. Identification of number of principal components (PCs) using Bayesian model selection. The figure corresponds to the case for which the threshold on correlation for screening predictors was set to 0.15. The model selects 36 as the optimum number of PCs.

calculating PCs using standard principal component analysis (SPCA) instead of PPCA. Number of PCs was chosen to be same as the optimum number of PCs yielded by Bayesian model selection method for PPCA. It is evident for the figure that the model performed poorly in forecasting AISMR values. The results obtained are in contradiction with earlier findings that about 80% of inter annual variability in AISMR can be explained by global sea surface temperatures prior to monsoon months. It should be pointed out that qualitatively similar results were obtained for different choices of training and testing periods.

How can this miss-match be explained? Here is a possible explanation: In the literature, high correlation between forecasted and observed rainfall during independent test period has been reported without allowance for the *selection bias*. The selection bias is induced because the test data are used at the first instance to select the predictors, which are then used to develop the forecast model. But what can be the *magnitude of selection bias*? To answer this question, the following analysis was done. The *entire data* from 1901 to 2005 (i.e. both training data and test data) were used to screen the set of potential predictors. The screened variables were then processed

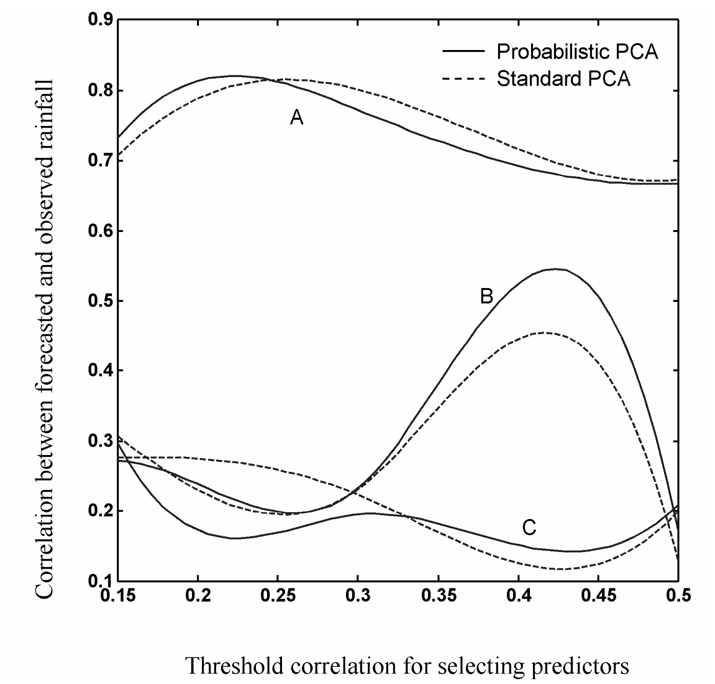


Fig. 4. Comparison of observed and forecasted rainfall values for testing period (1951 to 2005). The correlations were calculated for three strategies of screening potential predictors: (A) using entire data, (B) using training and alternate test data, and (C) using only test data.

through principal component analysis, the output of which forms the input to RVM model. The RVM model was trained in the same way as done before. The results obtained are shown as lines in the group of curves labeled ‘A’ in Fig. 4. The solid line corresponds to analysis using PPCA, whereas dashed line corresponds to SPCA. Clearly, a very high correlation can be obtained in this way. The results indicate that the selection bias can be very high and the results obtained without making allowance for it can be misleading. The selection bias, in general increases as the number of potential predictors increases. We further note that the concerns of getting deceptive results due to selection bias have been reported in other contexts. For example, Ambroise and McLachlan (2002) brought out the effects of selection bias in the context of cancer diagnosis and treatment.

Cross-validation and bootstrap estimates are the commonly used methods for assessing the performance of a model, when the number of data points is relatively small and the number of possible predictors large. However, these methods are not directly applicable to the dataset where the relationship between predictors and predictands is evolving over time. To address this problem a simple strategy is devised. Alternate years from the test data set are used along with the training dataset to screen the possible set of predictors. The performance of model is then assessed on forecasting the values of left over years in the test data set. The performance measured using this strategy is more realistic for an operational forecast model. The results obtained using this method are presented as group of curves labeled ‘B’ in Fig. 4. As expected

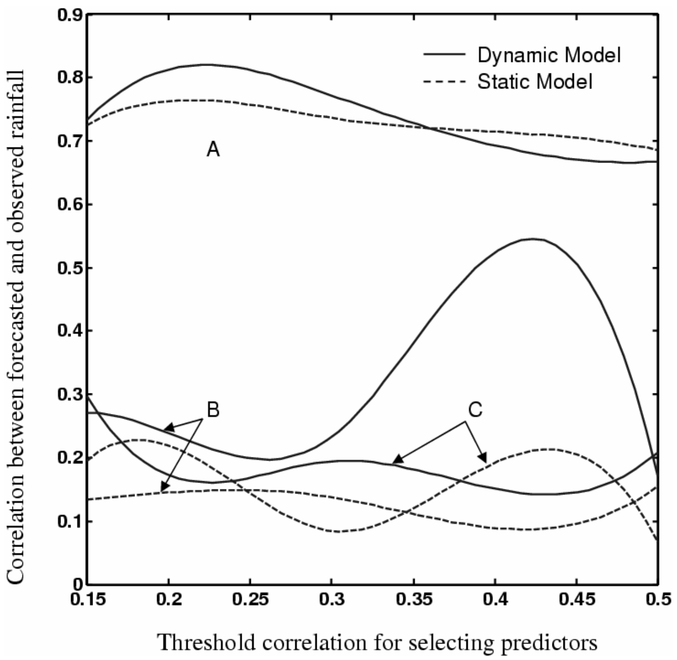


Fig. 5. Comparison of static and dynamic forecast models in forecasting rainfall values for testing period (1951 to 2005). The correlations were calculated for three strategies of screening potential predictors: (A) using entire data, (B) using training and alternate test data, and (C) using only training data.

the correlation between forecasted and observed rainfall values is in between two previous cases. Further, form Fig. 4 it can be inferred that the performance of forecasting model using PPCA or SPCA are similar. Nonetheless, PPCA is preferable because it provides an objective way of deciding the number of PCs.

The performance of a dynamic and a static model are also compared and the results are presented in Fig. 5. Both the static and the dynamic models were trained using data from 1901 to 1950. The dynamic model updates itself at each step as new data becomes available, while the static model remains same. The correlation between the forecasted and the observed values for the two models, for all three ways of screening potential predictors i.e. using: (A) entire data, (B) training and alternate test data, and (C) only training data are shown in Fig. 5. As expected the performance of dynamic model is better than static model. The advantages of dynamic model over static model are more pronounced for case B. The results corroborate the earlier findings that the relationship between SST and AISMR are continuously changing over time and that the dynamic forecast model is more suitable for forecasting AISMR.

6 Concluding Remarks

In this study, an attempt has been made to explore the links between All India Summer Monsoon Rainfall (AISMR) and global sea surface temperature (SST). The exploratory data analysis indicated that the relationship between AISMR and SST over different oceanic sectors is not static, but continuously changes with time. Not only that, the analysis revealed that the set of predictors for AISMR also changes with time. These findings indicate that a reliable statistical forecast of AISMR can be obtained only by a model that progressively updates itself by accounting for these changing relationships. Further, it was pointed out that any sort of ad-hoc or hand-crafted method of updating the model is unlikely to be optimal.

To address the above mentioned problem in a principled way, a Bayesian framework was adopted in this study. A methodology involving probabilistic principal component analysis, Bayesian model selection, and a sparse Bayesian learning algorithm (relevance vector machine) was introduced. The methodology automatically selects the best predictors in the global SST data for a forecasting model using principle of automatic relevance determination. Further, it progressively updates the predictor set and the relationship between predictors and predictand (AISMR) as new data becomes available.

The application of the proposed methodology to the forecasting of AISMR indicated that the strategy of constantly updating the model consistently provided better results than its static counterpart. However, in contrast to the results reported in the literature, the model developed in this work could only partially explain the interannual variability in the AISMR series. This discrepancy in the results indicated towards the problem of selection bias. It was found that, in the literature, high correlation between forecasted and observed rainfall during independent test period has been reported without allowance for the selection bias (i.e. test data are used to select the predictors for the model). To further understand the implications of selection bias in forecasting AISMR, its magnitude was estimated. The magnitude of selection bias was found to be strikingly high. The correlation between observed and forecasted value of AISMR jumped from ~ 0.25 to ~ 0.8 for unbiased to biased forecasts. These findings provide an inkling as to why there are frequent failures in forecasting AISMR even when model prediction error is small. It further points out the need to develop a strategy for estimating model prediction error for the systems that evolve over time. Towards this end, a simple strategy is recommended that considers alternate years in the test data for estimating model prediction error.

The methodology developed in this study is limited in several ways. Firstly, the methodology does not account for the measurement errors in SST data which is ubiquitous and significantly vary in space and time. Secondly, the sequential learning algorithm of relevance vector machine (RVM) is sensitive to the noise in the data. The problem is particularly pronounced for estimating noise term σ^2. Thirdly, the sequential learning algorithm for RVM tries to maximize the marginal likelihood of the data; however it does not guarantee a global optimum solution. The last two problems can be partly addressed by performing Monte Carlo simulations. Nevertheless, in spite of many limitations, the preliminary results obtained from the proposed methodology are promising. However, several avenues should be explored to further refine this attempt.

The true unbiased skill in forecasting AISMR using global SST data is very low. Even the use of sophisticated dynamic forecasting model can only marginally improve the forecasting performance. This brings to the fore the fact that the vagaries of Indian monsoon are difficulty to predict. The skillful forecasting of AISMR still remains a challenge. Perhaps more advanced soft-computing techniques in association with improved physical understanding of the monsoon system will improve the quality of forecasts.

References

Ambroise, C., McLachlan, G.J.: Selection bias in gene extraction on the basis of microarray gene-expression data. Proceedings of National Academy of Science USA 99(10), 6562–6566 (2002), doi:10.1073/pnas.102102699

Barnston, A.G., et al.: Improving seasonal prediction practices through attribution of climate variability. Bulletin of the American Meteorological Society 86(1), 59–72 (2005)

Berger, J.O.: Statistical decision theory and Bayesian analysis, 2nd edn. Springer, New York (1985)

Bhalme, H.N., Rahalkar, S.S., Sikder, A.B.: Tropical Quasi-Biennial Oscillation of the 10-mb wind and Indian monsoon rainfall-implications for forecasting. International Journal of Climatology 7(4), 345–353 (1987), doi:10.1002/joc.3370070403

Bishop, C.M.: Pattern Recognition and Machine Learning. In: Information Science and Statistics, 1st edn., Springer, New York, USA (2006)

Bjerknes, J.: Atmospheric teleconnections from the equitorial Pacific. Monthly Weather Review 97(3), 163–172 (1969)

Cannon, A.J., McKendry, I.G.: Forecasting all-India summer monsoon rainfall using regional circulation principal components: a comparison between neural network and multiple regression models. International Journal of Climatology 19(14), 1561–1578 (1999)

Charney, J.G., Shukla, J.: Predictability of monsoon. In: Lighthill, J., Pearce, R.P. (eds.) Monsoon Dynamics, pp. 99–108. Cambridge University Press, Cambridge (1981)

Clark, C.O., Cole, J.E., Webster, P.J.: Indian ocean SST and Indian summer rainfall: Predictive relationships and their decadal variability. Journal of Climate 13(14), 2503–2519 (2000)

Fasullo, J.: A stratified diagnosis of the Indian monsoon – Eurasian snow cover relationship. Journal of Climate 17(5), 1110–1122 (2004)

Gadgil, S., et al.: On forecasting the Indian summer monsoon: the intriguing season of 2002. Current Science 83(4), 394–403 (2002)

Gadgil, S. et al.: Extremes of the Indian summer monsoon rainfall, ENSO and equatorial Indian Ocean oscillation. Geophysical Research Letters 31, L2213 (2004), doi: 10.1029/2004GL019733

Goswami, B.N., et al.: A physical mechanism for North Atlantic SST influence on the Indian summer monsoon. Geophysical Research Letters 33, L02706 (2006), doi: 10.1029/2005GL024803

Horel, J.D.: A rotated principal component analysis of the interannual variability of the Northern Hemisphere 500 mb height field. Monthly Weather Review 109(10), 2080–2092 (1981)

Iyengar, R.N., Raghu, K.S.: Intrinsic mode functions and a strategy for forecasting Indian monsoon rainfall. Meteorology and Atmospheric Physics 90(1–2), 17–36 (2005)

Khalil, A., et al.: Sparse Bayesian learning machine for real-time management of reservoir releases. Water Resources Research 41, W11401 (2005), doi: 10.1029/2004WR003891

Khalil, A.F., et al.: Multiobjective analysis of chaotic dynamic systems with sparse learning machines. Advances in Water Resources 29(1), 72–88 (2006)

Kishtawal, C.M., et al.: Forecasting summer rainfall over India using genetic algorithm. Geophysical Research Letters 30(23), 2203 (2003), doi: 10.1029/2003GL018504

Kothawale, D.R., Munot, A.A., Borgaonkar, H.P.: Temperature variability over the Indian Ocean and its relationship with Indian summer monsoon rainfall. Theoretical and Applied Climatolog (in press, 2007), doi: 10.1007/s00704-006-0291-z

Kripalani, R.H., Kulkarni, A., Sabade, S.S.: Western Himalayan snow cover and Indian monsoon rainfall: A re-examination with INSAT and NCEP/NCAR data. Theoretical and Applied Climatology 74(1), 1–18 (2003)

Krishnan, R., Sugi, M.: Pacific decadal oscillation and variability of the Indian summer monsoon rainfall. Climate Dynamics 21(3), 233–242 (2003)

Kucharski, F., Molteni, F., Yoo, J.H.: SST forcing of decadal Indian monsoon rainfall variability. Geophysical Research Letters 33, L03709 (2006), doi: 10.1029/2005GL025371

Kumar, K.K., Rajagopalan, B., Cane, M.A.: On the weakening relationship between the Indian Monsoon and ENSO. Science 284(5423), 2156–2159 (1999)

Kumar, K.K., et al.: Unraveling the mystery of Indian monsoon failure during El-Niño. Science 314(5796), 115–119 (2006)

Lau, K.M., Kim, K.M., Shen, S.S.P.: Potential predictability of seasonal precipitation over the United States from canonical ensemble correlation predictions. Geophysical Research Letters 29(7), 1097 (2002), doi: 10.1029/2001GL014263

Li, T., et al.: On the relationship between Indian Ocean sea surface temperature and Asian summer monsoon. Geophysical Research Letters 28(14), 2843–2846 (2001)

Li, Y., Campbell, C., Tipping, M.E.: Bayesian automatic relevance determination algorithms for classifying gene expression data. Bioinformatics 18(10), 1332–1339 (2002)

Minka, T.P.: Automatic choice of dimensionality for PCA. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in neural information processing systems, vol. 13, pp. 598–604. MIT Press, Cambridge (2001)

Monahan, A.H.: Nonlinear principal component analysis: Tropical Indo-Pacific sea surface temperature and sea level pressure. Journal of Climate 14(2), 219–233 (2001)

Mooley, D.A., Munot, A.A.: Relationships between Indian summer monsoon and pacific SST/SOI tendency from winter to spring and their stability. Theoretical and Applied Climatology 56(3), 187–197 (1997)

Nicholls, N.: Predicting Indian monsoon rainfall from sea-surface temperature in the Indonesia-north Australia area. Nature 306(5943), 576–577 (1983)

Nicholls, N.: All-India summer monsoon rainfall and sea surface temperatures around Northern Australia and Indonesia. Journal of Climate 8(5), 1463–1467 (1995)

Pai, D.S., Rajeevan, M.: Empirical prediction of Indian summer monsoon rainfall with different lead periods based on global SST anomalies. Meteorology and Atmospheric Physics 92(1), 33–43 (2006)

Parthasarathy, B., Munot, A.A., Kothawale, D.R.: All-India monthly and seasonal rainfall series: 1871–1993. Theoretical and Applied Climatology 49(4), 217–224 (1994)

Prasad, K.D., Singh, S.V.: Possibility of predicting Indian monsoon rainfall on reduced spatial and temporal scales. Journal of Climate 5(11), 1357–1361 (1992)

Rajeevan, M.: Winter surface pressure anomalies over Eurasia and Indian summer monsoon. Geophysical Research Letters 29(10), 1454 (2002), doi: 10.1029/2001GL014363

Rajeevan, M., Guhathakurta, P., Thapliyal, V.: New models for long range forecasts of summer monsoon rainfall over North West and Peninsular India. Meteorology and Atmospheric Physics 73(3), 211–225 (2000)

Rajeevan, M., et al.: New statistical models for long-range forecasting of southwest monsoon rainfall over India. Climate Dynamics 28(7–8), 813–828 (2007)

Rao, K.G., Goswami, B.N.: Interannual variations of sea surface temperature over the Arabian Sea and the Indian monsoon: A new perspective. Monthly Weather Review 116(3), 558–568 (1988)

Rayner, N.A., et al.: Global analyses of sea surface temperature, sea ice, and night marine air temperature since the late nineteenth century. Journal of Geophysical Research 108(D14), 4407,1–37 (2003)

Robock, A., et al.: Land surface conditions over Eurasia and Indian summer monsoon rainfall. Journal of Geophysical Research 108(D4), 4131,1–17 (2003), doi: 10.1029/2002JD002286

Roweis, S.T.: EM algorithms for PCA and SPCA. In: Jordan, M.I., Kearns, M.J., Solla, S.A. (eds.) Advances in neural information processing systems, vol. 10, pp. 626–632. MIT Press, Cambridge (1998)

Sahai, A.K., et al.: Long-lead prediction of Indian summer monsoon rainfall from global SST evolution. Climate Dynamics 20(7), 855–863 (2003)

Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. In: Adaptive Computation and Machine Learning, 1st edn., MIT Press, Cambridge, MA, USA (2001)

Shabbar, A., Barnston, A.G.: Skill of seasonal climate forecasts in Canada using canonical correlation analysis. Monthly Weather Review 124(10), 2370–2385 (1996)

Smith, T.M., Reynolds, R.W.: Extended reconstruction of global sea surface temperatures based on COADS data. Journal of Climate 16(10), 1495–1510 (2003)

Srivastava, A.K., Rajeevan, M., Kulkarni, R.: Teleconnection of OLR and SST anomalies over Atlantic Ocean with Indian summer monsoon. Geophysical Research Letters 29(8), 1284 (2002), doi: 10.1029/2001GL013837

Tipping, M.E.: Sparse Bayesian learning and the Relevance vector machine. Journal of Machine Learning Research 1(3), 211–244 (2001)

Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B 61(3), 611–622 (1999)

Tipping, M.E., Faul, A.: Fast marginal likelihood maximization for sparse Bayesian models. In: Bishop, C.M., Frey, B.J. (eds.) Proceedings of the ninth international workshop on artificial intelligence and statistics, January 3–6, Key West, Florida, 8 pages (2003)

Tukey, J.W.: Exploratory data analysis. Addison-Wesley, Reading, MA, USA (1977)

van den Dool, H.M., Saha, S., Johansson, A.: Empirical orthogonal teleconnections. Journal of Climate 13, 1421–1435 (2000)

Vautard, R., Pires, C., Plaut, G.: Long-range atmospheric predictability using Space-Time principal components. Monthly Weather Review 124(2), 288–307 (1996)

Fault Diagnosis of Electronic Circuits Using Cellular Automata Based Pattern Classifier

Pradipta Maji[1] and P. Pal Chaudhuri[2]

[1] Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India
pmaji@isical.ac.in
[2] Cellular Automata Research Laboratory, Rajarhat Megacity, Kolkata, India
palchau@carltig.res.in

1 Introduction

This chapter formulates fault diagnosis in electronic circuits as a pattern classification problem. The proposed pattern classification scheme employs the computing model of a special class of sparse network referred to as cellular automata (CA). A particular class of CA referred to as multiple attractor CA (MACA) has been projected as a classifier of faulty response-pattern of a circuit. The genetic algorithm (GA) is employed to synthesize the desired CA required for diagnosis of a circuit under test (CUT). The CUT is assumed to have a network of large number of circuit components partitioned into a number of sub-circuits referred to as modules. Introduction of GA significantly reduces the design overhead of the MACA based classifier that supports:

1. low memory overhead for diagnosis - reduction of one to two order of magnitude of memory overhead has been achieved over that required for conventional fault dictionary based diagnosis scheme;
2. excellent diagnostic resolution and low diagnostic aliasing; and
3. low cost hardware of a generic fault diagnosis machine (FDM) with simple, regular, modular, and cascadable structure of CA that suits ideally for very large scale integration (VLSI) implementation.

Thus, the FDM can be viewed as a Watch-Dog Processor intended for high speed on-line diagnosis for critical application areas. The following scenario motivated us to undertake this research.

In order to improve the product quality, reduce time to market, and cut down production cost, the demand for fault diagnosis in an electronic circuit has greatly increased. The objective of fault diagnosis is to guide the test engineers to search the physical location of the defect on a circuit in the early production phase. Use of fault dictionaries is a probable solution for the diagnosis process, particularly when repeated diagnosis is required for different copies of the same circuit [1, 2, 3]. But this scheme becomes inefficient for a sufficiently large circuit due to the large volume of fault dictionary. Different schemes to compact the size of the dictionary are addressed in [4, 5, 6]. However, such compaction reduces diagnostic resolution since multiple response patterns for different faults get compressed to the identical signature. The best test environment for diagnostic purposes should differentiate between all faults that are distinguishable.

In the above background, we propose an efficient fault diagnosis scheme to identify a defect down to a set of candidate locations referred to as faulty module of an electronic circuit. Both analog and digital circuits can be handled by this scheme. The diagnosis scheme employs two class classifier designed with a special class of CA referred to as MACA. The MACA effectively provides an implicit storage mechanism of voluminous response data and thereby providing the solution to the problem of space and time associated with handling large fault dictionary. Preliminary concept of MACA application for diagnosis of digital circuits has been reported in [7, 8, 9]. However, no design details of MACA based classifier are provided in these papers.

In order to present the underlying principle of MACA based classifier design, this chapter starts with an introduction to CA preliminaries including MACA fundamentals in Section 2. The pattern classifier and its application for fault diagnosis is next presented in Section 3. The memory overhead of the proposed design and its comparison with that of conventional fault dictionary based scheme is also undertaken in this section. Formulation of the evolutionary program of GA for design of the classifier is next presented in Section 4. Section 5 reports the experimental results of fault diagnosis on different benchmark digital circuits and a few analog circuits synthesized with OTA network [10]. Finally, the hardware realization of the FDM is outlined in Section 6.

2 Cellular Automata

A CA [11] consists of a number of cells. The next state of a cell depends on the present states of its neighbors. In a 3-neighborhood dependency, the next state q_i^{t+1} of a cell is assumed to be dependent only on itself and on its two neighbors (left and right), and is denoted as $q_i^{t+1} = f(q_{i-1}^t, q_i^t, q_{i+1}^t)$, where q_i^t represents the state of the i^{th} cell at t^{th} instant of time. 'f' is the next state function and referred to as the rule of the automata. The decimal equivalent of the next state function, as introduced by Wolfram [12], is the rule number of the CA cell. For example, Rule 90: $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$ and Rule 150: $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$, where $\oplus$ (XOR function) denotes modulo-2

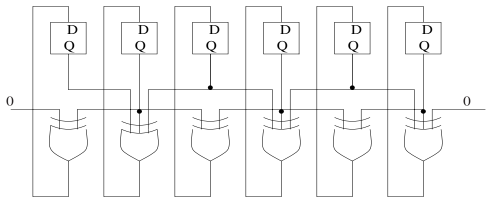


Fig. 1. A one dimensional 6 cell linear CA with rule vector $< 90, 150, 90, 150, 90, 150 >$

addition. Since 'f' is a function of 3 variables, there are 2^{2^3} i.e., 256 possible next state functions (rules) for a CA cell. Out of 256 rules there are only 7 rules with XOR logic. The CA employing XOR rule is referred to as linear CA. Fig. 1 represents a 6 cell linear CA. The fault-pattern classifier reported in this chapter employs this CA. The linear CA theory and its applications in diverse fields have been extensively dealt with in [13]. A brief overview follows for easy reference.

2.1 Characterization of Linear CA

The state $S_t(x)$ of an n-cell CA at t^{th} instant of time is a string of binary symbols. The next state $S_{t+1}(x)$ of the CA is given by: $S_{t+1}(x) = T.S_t(x)$. T is an $n \times n$ characteristic matrix [13], where

$$T_{ij} = \begin{cases} 1, & \text{if next state of } i^{th} \text{ cell depends on present state of } j^{th} \text{ cell}, \forall i, j = 1 \text{ to } n \\ 0, & \text{otherwise.} \end{cases}$$

If we restrict to 3-neighborhood dependency, then T_{ij} can have non-zero values only for $j = (i-1)$, i, and $(i+1)$. Thus, T becomes a tri-diagonal matrix. We next briefly state some of the important concepts concerning linear algebra. The GA formulation presented in Section 4 of this chapter effectively employs these concepts for synthesis of CA based pattern classifier.

Characteristic Polynomial: The polynomial $\phi(x)$ of which T is a root is referred to as the characteristic polynomial of the CA. The characteristic polynomial $\phi(x)$ is derived from T by calculating $\det(T + Ix)$.

Property 1: Multiple T's can have the same characteristic polynomial [13]. So the same polynomial represents multiple vector subspaces.
Elementary Divisors: The characteristic polynomial $\phi(x)$ comprises of invariant polynomials $\phi_i(x)^{n_i}$ where $\phi_i(x)$ is irreducible. The polynomials $\phi_i(x)^{n_i}$, invariant to the linear operator T, are referred to as elementary divisors.

Property 2: Elementary divisors arranged in different orders produce different vector subspaces.

Minimal Polynomial: Minimal polynomial $m(x)$ is the minimum degree factor of the characteristic polynomial that is annihilated by T - that is, $m(T)$=0.

If all the states in the state transition diagram of a CA lie in some cycles, it is a group CA, whereas in a non-group CA, the states form both inverted trees (transients) and cycles (Fig. 2). For a group CA $\det[T] \neq 0$, while for the non-group CA, $\det[T] = 0$. In this chapter we have employed a special class of non-group CA, referred to as MACA, for designing the classifier.

2.2 Multiple Attractor CA and Its Characterization

The state transition graph of an MACA consists of a number of cyclic and non-cyclic states. The set of non-cyclic states of an MACA form inverted trees rooted at the cyclic states. The cycles are referred to as attractors. Fig. 2 depicts the state transition diagram of a 5-cell MACA with four attractors

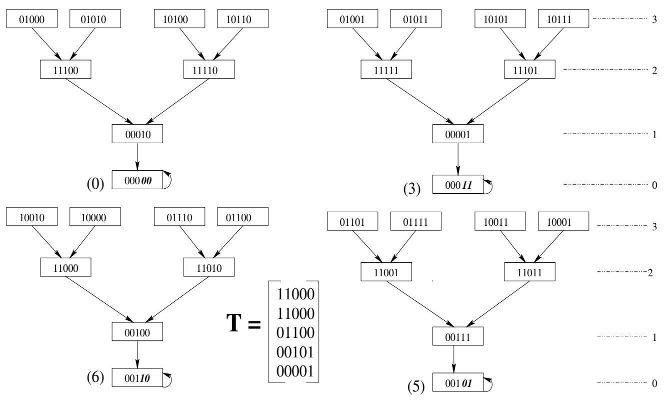


Fig. 2. State transition diagram of a 5-cell MACA

{00000,00011,00110,00101} having self loop. The states of a tree rooted at the attractor α form the α-basin. With reference to the state transition diagram of a MACA, the depth d of the CA is the number of edges between a non-reachable state and the attractor. The depth d of the 5-cell MACA of Fig. 2 is 3. The detailed characterization of MACA is available in [13]. A few fundamental results for an n-cell MACA having k number of attractors is next outlined.

1. Result I: The characteristic polynomial of the MACA is of the form $x^{n-m}(1+x)^m$, where $m = log_2(k)$.
2. Result II: The characteristic polynomial noted above can be also written in elementary divisor form as $(1+x)(1+x)\cdot\cdot m\ times\ \ x^{d_1}x^{d_2}\cdot\cdot x^{d_p}$ where $d_1 \geq d_2 \cdot\cdot \geq d_p$ and $d_1 + d_2 + \cdot\cdot + d_p = n - m$.
3. Result III: Minimal polynomial of MACA is $x^{d_1}(1+x)$, where depth $= d_1$.

   Definition 1. *An m-bit field of an n-bit pattern set is said to be pseudo-exhaustive if all possible* 2^m *patterns appear in the set.*

4. **Theorem 1.** *[13] In an n cell MACA with* $k = 2^m$ *attractor basins, there exists m-bit positions at which the attractors generate pseudo-exhaustive* 2^m *patterns.*
5. **Theorem 2.** *[13] The modulo-2 sum of two states is the non-zero predecessor of 0-state (pattern with all 0's) if and only if the two states lie in the same MACA basin.*

Example 1. *The example MACA of Fig. 2 is used to illustrate the above results. It is a* 5*-cell MACA having 4 attractors and with depth equal to 3.*

1. *Result I: The characteristic polynomial is* $x^3 \cdot (1+x)^2$*. Therefore,* $m = \log_2(k) = 2$*. This is consistent with the result in Fig. 2 where number of attractors* $k = 4$*.*
2. *Result II: The characteristic polynomial in elementary divisor form is* $x^3 \cdot (1+x)\cdot(1+x)$*.*

3. *Result III: The minimal polynomial is $x^3 \cdot (1+x)$.*
4. *In Fig. 2, two least significant bit positions of the set of attractors constitute the pseudo exhaustive field (PEF).*
5. *We take an attractor {00011} and any two states {11111, 11101} of the attractor basin. The modulo-2 sum of these two patterns is {00010} which is a state in 0-basin. By contrast, if we take two states {00001} and {11000} belonging to two different attractor basins {00001} and {11000} respectively, their modulo-2 sum is {11011} which is a state in a non-zero attractor ({00101}) basin.*

3 MACA: A Pattern Classifier

An n-bit MACA with k-attractor basins can be viewed as a natural classifier [13]. It classifies a given set of patterns into k-distinct classes, each class containing the set of states in the basin. In the process an MACA acts as an implicit memory. To enhance the classification accuracy of the machine, most of the works [7, 8, 9, 14] have employed MACA to classify patterns into two classes (say I and II) where Class I is represented by one set of attractor basins (say [I]= {00011, 00110 and 00101} in Fig. 2) while Class II is represented by the rest of the basins (say [II] = {00000}). All the patterns in the attractor basins [I] belong to Class I while rest of the patterns belong to Class II. When a CA is loaded with an input pattern and is allowed to run in autonomous mode for a number of cycles equal to the depth of CA, it travels through a number of states and ultimately reaches an attractor state. As per Theorem 1, the pseudo-exhaustive fields (PEF) will identify the class of the patterns uniquely. The PEF yield the memory address of the class to which the pattern belongs to. Therefore, Class I yields memory address $\{01, 10, 11\}$ while Class II will be identified by the address $\{00\}$ (Fig. 2). Following two issues deserve attention at this stage, which set our design perspective.

1. The two class classifier identifies the class of an element in a constant time d, where d is the depth of MACA. If we didn't have the proposed MACA based classifier and yet wanted a constant time identification, we would have needed a hash table of size 2^n, n being the width of the pattern. This is impractical even for a moderate size of n.
2. For an ideal classifier, to distinguish between two classes, we would need one bit to store class information. The classifier of Fig. 2 requires two bits to store pseudo-exhaustive field of a CA attractor basins. A k-attractor two class classifier needs $log_2(k)$ bits. This involves a memory overhead compared to an ideal classifier.

3.1 Design Goal

Our design will strive to find an MACA that would have least number of attractor basins. That is, to classify pattern set into two classes, we should ideally find an MACA with two attractor basins - each basin having the members of the specific class. Even if this ideal situation is not attained, the algorithm should design an

MACA based classifier having minimum number of attractor basins denoted as 2^m, m being the number of bits in the PEF of an attractor. While one subset of basins accommodates the elements of one class, the remaining subset houses the elements of the other class. For example, three attractor basins of the example MACA of Fig. 2 cover elements of Class I, while fourth one constitutes Class II.

3.2 MACA Based Two Class Classifier

The design of the MACA based classifier for two pattern sets P_1 and P_2 should ensure that elements of one class (say P_1) are covered by a set of attractor basins that do not include any member from the class P_2. Any two patterns $a \in P_1$ and $b \in P_2$ should fall in different attractor basins. According to Theorem 2, the pattern derived out of modulo-2 sum of a and b ($a \oplus b$) should lie in a non-zero attractor basin. Let X be a set formed from modulo-2 sum of each member of P_1 with each of P_2 that is, $X = \{x_l \mid x_l = (a_i \in P_1) \oplus (b_j \in P_2) \forall_{i,j}\}$. Therefore, all members of X should fall in non-zero basin. This implies that the following set of equations should be satisfied.

$$T^d \cdot X \neq 0 \tag{1}$$

where T is a valid MACA to be employed for designing two class classifier with d as the depth (Result III) of the MACA.

3.3 Circuit Diagnosis - A Pattern Classification Problem

A circuit has been assumed to consist of a number of sub-circuits or modules. The diagnosis framework has been designed to diagnose the faulty module in the circuit. Let, for a circuit with K number of modules - $M_1, M_2, \cdots, M_K$, $\{S_{11}, S_{12}, \cdots\}$, $\{S_{21}, S_{22}, \cdots\}$, $\cdots$, $\{S_{K1}, S_{K2}, \cdots\}$ be the signature sets of K faulty modules - S_{ij} refers to the signature generated due to j^{th} fault in the i^{th} module M_i. The aim is to design a classifier which has to classify the signature sets into K classes. The following simple example illustrates the formulation of circuit diagnosis as a pattern classification problem.

Let us consider the Example CUT 'EC' of Fig. 3 with 5 primary outputs (POs). It has two partitions - Module 1 and Module 2. The faulty signature set S_1 for Module 1 is computed through fault simulation by injecting each of the faults of this module. Similarly, the set S_2 is also computed for Module 2. The signature analyzer is a 5 bit maximum length CA [13].

Let, the faulty signature sets generated for two modules be $S_1 = \{0, 2, 8, 10, 30\}$ and $S_2 = \{1, 3, 11, 31\}$. The MACA (T) of Fig. 2 satisfies the Relation 1, and represents the desired pattern classifier for the sets S_1 and S_2, where S_1 is covered by the zero attractor basin and S_2 by the non-zero attractor basins. The memory locations addressed by the PEF (Theorem 1) of attractors store the faulty module number. At the testing phase, a faulty chip realizing 'EC' generates any one (say) signature 10 (01010). If we run the CA (T) of Fig. 2 with 10 as seed for three cycles, it will reach to the attractor 00000 (Fig. 2). By observing PEF (00) of attractor, the faulty module (Module 1), can be identified.

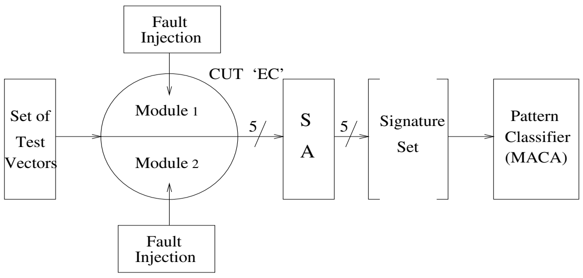


Fig. 3. Diagnosis of an example CUT 'EC'

From the above example, it can be observed that the problem of diagnosis in electronic circuits maps to the design of MACA based classifier. The large fault dictionary gets replaced with the memory to store the MACA (T) and the PEF symbols of its attractors. In effect, the MACA acts as an implicit memory.

3.4 Design of Multi-class Classifier

A two class classifier can be viewed as a single stage classifier. For designing a multi-class classifier, this scheme of single stage classification will be repeatedly employed leading to a multi-stage classifier consisting of multiple MACA, each MACA corresponds to a single stage (Fig. 4). Hence, in the rest of the chapter, we concentrate on designing an efficient MACA based two class classifier.

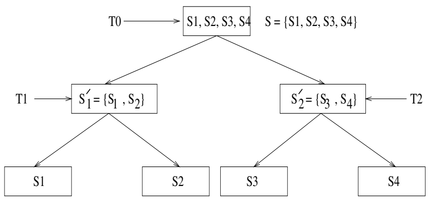


Fig. 4. MACA based multi-class classifier to handle 4 classes: A leaf node represents a class in the input set $S = \{S_1,S_2,S_3,S_4\}$

3.5 An Example Fault-Pattern Classifier

This subsection illustrates the design of MACA based multi-class classifier for diagnosis of a faulty module in a circuit that has been identified as faulty with a conventional testing scheme. Let, a 4 module circuit with n PO lines ($n = 5$) generate 5 bit faulty signatures S_{ij} ($i = 1, 2, 3, 4; j = 1, 2, \cdots$) on introduction of faults in each module, where $S_1 = \{0, 6, 10, 12, 16, 22, 26, 28\}$; $S_2 = \{2, 4, 8, 14, 18, 20, 24, 30\}$; $S_3 = \{5, 7, 13, 15, 21, 23, 29, 31\}$; and $S_4 = \{1, 3, 9, 11, 17, 19, 25, 27\}$. The MACA based classifier is designed to output the class S_i for a given input pattern S_{ij}. To solve this problem, we employ multiple two

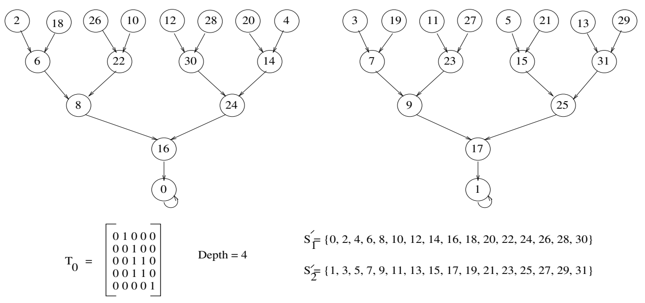

T0 = [0 1 0 0 0; 0 0 1 0 0; 0 0 1 1 0; 0 0 1 1 0; 0 0 0 0 1]
Depth = 4
Ś1 = {0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30}
Ś2 = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31}

Fig. 5. First level of classification: Set S is divided into $\acute{S}_1$ and $\acute{S}_2$ and obtain T_0

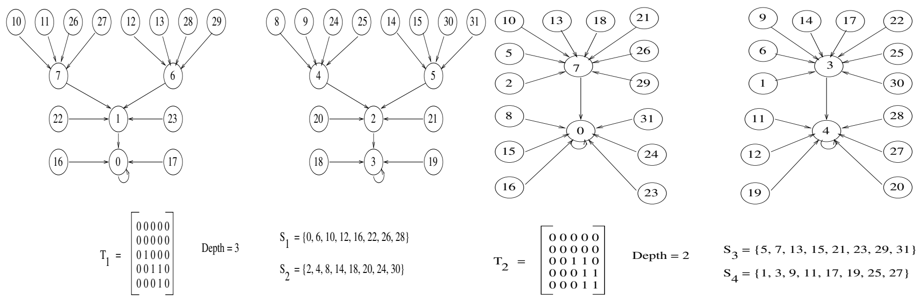

T1 = [0 0 0 0 0; 0 0 0 0 0; 0 1 0 0 0; 0 0 1 1 0; 0 0 0 1 0]
Depth = 3
S1 = {0, 6, 10, 12, 16, 22, 26, 28}
S2 = {2, 4, 8, 14, 18, 20, 24, 30}
T2 = [0 0 0 0 0; 0 0 0 0 0; 0 0 1 1 0; 0 0 0 1 1; 0 0 0 1 1]
Depth = 2
S3 = {5, 7, 13, 15, 21, 23, 29, 31}
S4 = {1, 3, 9, 11, 17, 19, 25, 27}

Fig. 6. Second level of classification: Set $\acute{S}_1$ is divided into S_1 and S_2 with T_1; and Set $\acute{S}_2$ is divided into S_3 and S_4 with T_2

class classifiers. Fig. 5 illustrates the design of 3 MACA based classifiers at two levels to diagnose one of the four faulty modules. At first level, we divide whole pattern set S (Fig. 4) into two classes - $\acute{S}_1$ and $\acute{S}_2$, where $\acute{S}_1 = \{S_1, S_2\}$ and $\acute{S}_2 = \{S_3, S_4\}$. Next we synthesize an MACA (T_0) which can classify two distinct classes $\acute{S}_1$ and $\acute{S}_2$. Fig. 5 represents two classes $\acute{S}_1$ and $\acute{S}_2$ along with the MACA (T_0). The same process is next repeated for $\acute{S}_1$ and $\acute{S}_2$ to isolate S_1, S_2 and S_3, S_4 respectively generating two MACA - T_1 and T_2 (Fig. 6). At the diagnosis phase, a pattern (say) $S_{24} = 14$ be given as the input and its class is to be identified. At the first stage, the classifier designed with the MACA (T_0) of Fig. 5 is loaded with 14. The PEF (Theorem 1) of 0-attractor of the MACA returns the desired class $\acute{S}_1$ (at that level) after 3 time steps. Finally, the classifier of Fig. 6 (T_1) is loaded with 14 and it returns its desired class S_2 after 3 time steps.

The above formulation employs MACA as an implicit memory to store the faulty signature patterns rather than explicit storage of fault dictionary. It calls for an analysis of memory overhead to store MACA structures.

3.6 Memory Overhead

This subsection analyzes the memory overhead of the MACA based scheme for fault diagnosis of electronic circuits and compares it with that of the conventional

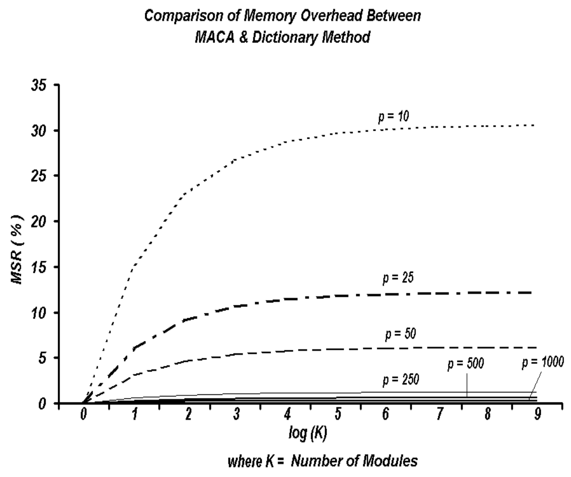


Fig. 7. Comparison of memory overhead between MACA and dictionary method for $n = 100$ and different values of average number of unique faulty signatures p

dictionary method. The CUT is assumed to have n PO lines and consequently it generates n-bit signature. The following theorem specifies memory overhead.

Theorem 3. *The memory overhead of MACA based classifier is given by*

$$\text{Memory(MACA)} = \sum_{i=0}^{\log_2 K-1} 2^i \cdot [(3n-1) + 2^m + \log_2(d)] \tag{2}$$

where K is the number of modules, m and d are the average values of the depth and the number of bits in the PEF of each MACA (T).

Proof. The memory overhead of the proposed classifier can be computed as the memory required to store the n-bit MACA based multi-class classifier (T's) for the CUT, the depth and the attractors corresponding to each MACA. For a circuit with K number of modules, total number of levels required to design multi-class classifier realized with multi-stage two class classifier is $\log_2 K$. Now, as we restrict to 3-neighborhood dependency, T becomes a tri-diagonal matrix. So, the number of bits required to store each n-bit MACA (T) is equal to $(3n-2)$. Let, d and m be the average values of the depth and the number of bits in the PEF (Theorem 1) of each MACA (T). Then, the memory required to design multi-class classifier in terms of bits is given by

$$\text{Memory(MACA)} = \sum_{i=0}^{\log_2 K-1} 2^i \cdot [(3n-1) + 2^m + \log_2(d)] \tag{3}$$

since, total number of nodes in the tree of multi-class classifier (Fig. 4) is $[\sum_{i=0}^{\log_2 K-1} 2^i]$ and the memory overhead to store the relevant information of each node is $[(3n-1) + 2^m + \log_2(d)]$.

In case of conventional dictionary method, the memory overhead is equal to the memory size required to store the number of unique faulty signatures in the CUT. Let, p be the average number of unique faulty signatures of a module in the CUT. Then, the total memory required in dictionary method is equal to

$$\text{Memory(Dictionary)} = n \cdot K \cdot p \tag{4}$$

The ratio of memory overhead as computed in Relations 3 and 4 noted as memory size ratio (MSR) is given by,

$$\text{MSR} = \frac{\text{Memory(MACA)}}{\text{Memory(Dictionary)}} = \frac{\sum_{i=0}^{\log_2 K-1} 2^i \cdot [(3n-1) + 2^m + \log_2(d)]}{n \cdot K \cdot p} \tag{5}$$

As the values of m and d are very small compared to n, Equation 5 reduces to

$$\text{MSR} = \frac{3 \cdot (K-1)}{K \cdot p} \tag{6}$$

Theorem 4. *Memory requirement for MACA based multi-class classifier is less than that required for dictionary based scheme if* $p > [\frac{3 \cdot (K-1)}{K \cdot p}]$.

Proof. The proposed MACA based scheme will be superior than the conventional dictionary based method if MSR < 1. This will be satisfied if the minimum value of p maintains the relation

$$p_{min} > 3 \cdot (1 - \frac{1}{K}). \text{ Hence, the proof.} \tag{7}$$

Fig. 7 represents the MSR percentage (ratio of memory required to implement the proposed scheme with respect to conventional dictionary method) at different values of number of modules (K). It is seen that the percentage of memory increases with the number of modules (K) of the CUT, but at a very slow gradient for $K > 8$. Whereas, for a fixed value of K, as the value of p (the average number of unique faulty signatures) increases, the percentage of memory required to implement MACA based diagnosis scheme decreases. The results shown in Fig. 7 establishes the following facts:

1. For a given value of K (the number of modules in the CUT), MSR value goes down with increase in the number of unique signatures p. That is, the memory requirement of MACA based classifier goes down compared to that required for dictionary based scheme if the number of unique signatures (p) increases; which leads to lesser diagnostic aliasing.
2. For a given value of p, the MSR value marginally increases with value of $K > 8$.
3. Combining above two facts, with $p \geq 25$ the MSR approaches a constant value irrespective of the value of K.

Low percentage of MSR of the proposed MACA based classifier is the result of use of MACA as an implicit memory. The above observations have been validated through extensive experimentation reported in Section 5 for diagnosis of benchmark digital circuits and a few analog designs reported in [10].

Design of an algorithm to synthesize an MACA based two class classifier for any given pattern set, is a hard problem. So, we fall back on the evolutionary scheme of GA to solve this problem. Next section presents an efficient formulation of GA for evolution of MACA based two class classifier while realizing the design goals reported in Section 3.

4 GA for Evolution of MACA

This section describes the GA based solution to evolve the MACA with the desired functionality. The aim of this evolution is to arrive at the desired MACA that can perform the task of two class classification with minimum attractors.

The basic structure of GA revolves around the concept of encoding a solution in bit string format referred to as chromosome and evolving the successive solutions according to its fitness. The three major functions of GA - random generation of initial population, crossover, mutation, and finally the fitness function, as developed in the current formulation, are next discussed.

4.1 Random Generation of Initial Population with Desired Format

To form the population, it must be ensured that each solution randomly generated is an n bit MACA with 2^m number of attractor basins where m can assume any value from 1 to n. From Result II of Section 2.2, the elementary divisor form of the characteristic polynomial of MACA is $(1+x)(1+x)\cdot\cdot m$ times $x^{d_1}x^{d_2}\cdot\cdot x^{d_p}$, where $d_1 + d_2 \cdots + d_p = (n - m)$, and the number of $(1 + x)$ determines the number of attractor basins.

As per the Property 2 of Section 2.1, the elementary divisors if arranged in different order, produces MACA with different attractor basins. For MACA synthesis, the elementary divisors are first randomly distributed among itself forming a sequence. Fig. 8 shows one such sequence for the characteristic polynomial $x^7(1 + x)^3$: $x^2 \cdot x^2 \cdot (1 + x) \cdot x \cdot (1 + x) \cdot (1 + x) \cdot x^2$. This format, referred to as pseudo-chromosome format, has been detailed in Section 4.2.

A CA (T matrix), as reported in [13], can be generated out of each elementary divisor. A tri-diagonal T matrix with characteristic polynomial $\phi_i(x)$ is accordingly synthesized. Let T_i matrices correspond to elementary divisors x^{d_i} (it will be a $d_i \times d_i$ matrix) and T_j matrices correspond to each of the elementary divisors $(1 + x)$. If T_is and T_js are randomly arranged in block diagonal form, the characteristic polynomial of the resultant T is $x^{n-m} \cdot (1 + x)^m$ and the minimal polynomial is $x^{d_p} \cdot (1 + x)$ and it generates the MACA [13]. The arrangement of block diagonal form, as shown in Fig. 9, has three different options that are

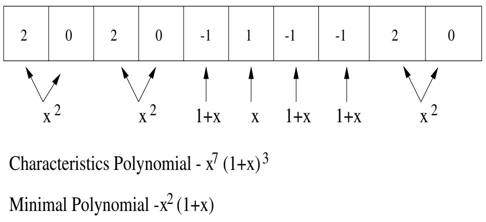

2 0 2 0 -1 1 -1 -1 2 0
x 2
x 2
1+x
x
1+x
1+x
x 2
Characteristics Polynomial - x7 (1+x)3
Minimal Polynomial -x2 (1+x)

Fig. 8. MACA in diagonal representation form

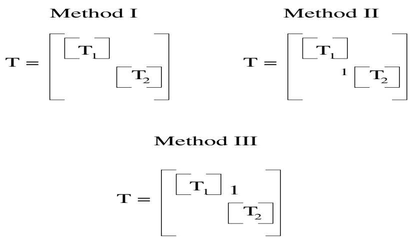


Fig. 9. Different methods to arrange two matrices $[T_1]$ and $[T_2]$ in block diagonal form. Method II (III) has '1' on lower (upper) diagonal position and so T_2 (T_1) has dependency on T_1 (T_2)

incorporated in designing the algorithm. The algorithm to randomly generate an MACA is next outlined.

Algorithm 1. *Synthesis Algorithm*
Input: n - the size of MACA, m - the number of bits in PEF (Theorem 1) and consequently 2^m is the number of attractor basins.
Output: T matrix representing an MACA.

1. *Randomly divide $(n - m)$ into $d_1, d_2 \cdots d_p$ to form the corresponding d_i's.*
2. *Synthesize T_i matrix from each elementary divisors x^{d_i}.*
3. *Synthesize T_j matrix from each elementary divisors $(1 + x)$.*
4. *Randomly arrange sequence of T_i and T_js, forming the pseudo-chromosome.*
5. *Use Method 1 of Fig. 9 to arrange two consecutive $T_i's$ or $T_j's$ in Block Diagonal Form.*
6. *Select a Method from Method I, II, and III to arrange a T_i and T_j in block diagonal form.*

4.2 Pseudo-chromosome Format

It is a method of representing an MACA with respect to the sequence in which its $x^{d_i}{'s}$ and $(1 + x)'s$ are arranged. It gives a semblance of chromosome and hence termed as pseudo-chromosome format. It is a string of n bits where

- d_i positions occupied by a x^{d_i} is represented by d_i followed by $(d_i - 1)$ zeros. For example, $x^3 = [300]$, and
- $(1 + x)$ is represented by -1.

The pseudo-chromosome format of the MACA is illustrated in Fig. 8. The MACA in Fig. 8 has the characteristic polynomial $x^7(1 + x)^3$. The sequence of $(1+x)'s$ and $x^{d_i}{}'s$ is $x^2 \cdot x^2 \cdot (1+x) \cdot x \cdot (1+x) \cdot (1+x) \cdot x^2$. Hence the representation in pseudo-chromosome format is $[2\ 0\ 2\ 0 - 1\ 1 - 1 - 1\ 2\ 0]$. Rather than the bit string of a conventional GA framework, the proposed scheme employs a symbol string of numerical digits with specified ordering. This ordering ensures that the corresponding CA is an MACA. A departure from the conventional form demands associated modifications for the crossover and mutation process.

4.3 Crossover Algorithm

The crossover algorithm implemented is similar in nature to the conventional one normally used for GA framework with minor modifications as illustrated below. The algorithm takes two MACAs from the present population (PP) and forms the resultant MACA. Like a single point crossover, it sets a crossover point and each half about the crossover point is selected from the two respective MACAs.

Fig. 10 shows an example of the crossover process. Two MACAs in pseudo-chromosome format are shown in Fig. 10(a) and Fig. 10(b). The first 5 symbols are taken from MACA$_1$, while the rest 6 symbols are taken from MACA$_2$. But due to this crossover, the resulting CA, as explained below violates the pseudo-chromosome format in positions 4 and 5. (encircled in Fig. 10(c)).

The pseudo-chromosome format has x^{d_i} represented by the digit d_i followed by $(d_i - 1)$ zeros. But in the case of Fig. 10(c), we have 3 followed by a single zero. This is a violation since the property of MACA is not maintained. So we take out those two symbols and form a CA of elementary divisor x^2 and locally adjust it as per the method suggested in Steps 4 and 5 of Algorithm 1. The resultant MACA after adjustment is shown in Fig. 10(d). The algorithm noted below formalizes the above discussions.

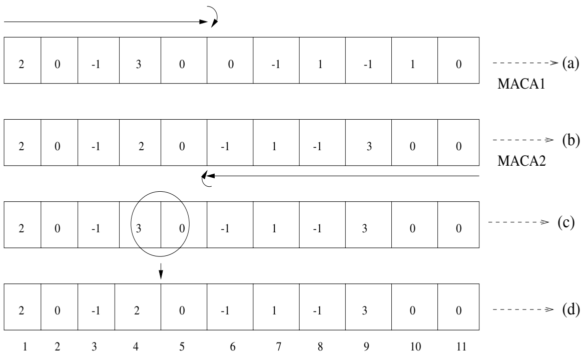


Fig. 10. An example of crossover technique

Algorithm 2. *Crossover Algorithm*
Input: Randomly selected two MACA ($MACA_1$ and $MACA_2$) from PP.
Output: New MACA.

```
1. Randomly generate a number $q$ between 1 and $n$.
2. Take the first $q$ bits from $MACA_1$ and the last $n - q$ bit from $MACA_2$.
3. Make the necessary adjustments in the partition points to conform pseudo-
   chromosome format for the resulting MACA.
4. Stop.
```

4.4 Mutation Algorithm

The mutation algorithm emulates the normal mutation scheme. It makes some minimal change in the existing MACA of PP to a new MACA for next population (NP). The MACA is mutated at a single point. In mutation algorithm, an $(1+x)$'s position is altered. In case some anomaly crops up due to such alteration, it is resolved to ensure that after mutation the new CA is also an MACA. The inconsistent format, as shown in Fig. 11(b) is the mutated version of Fig. 11(a). The inconsistency of the pseudo-chromosome format of Fig. 11(b) can be resolved locally to generate the consistent format of Fig. 11(c). The algorithm to resolve the inconsistency is elaborated below.

Algorithm 3. *Mutation Algorithm*
Input: Randomly select one MACA of PP.
Output: New MACA for NP.

```
1. Take the pseudo-chromosome format of the MACA.
2. Randomly select a $(1+x)'s$ position ($p_1$) and put it in a non-$(1+x)$ position
   ($p_2$).
3. Make necessary adjustment to conform to the pseudo-chromosome format of
   the resulting MACA.
4. Stop.
```

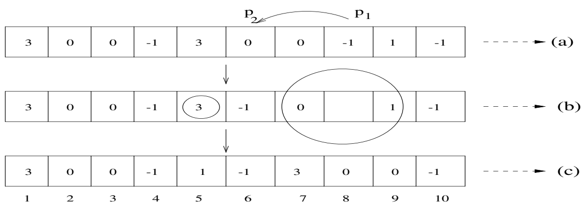


Fig. 11. An example of mutation technique

4.5 Fitness Function

The fitness of evolved MACA reflects two major goals of synthesis of MACA: F_1 - satisfiability of Relation 1 (Section 3.2): $T^d \cdot X \neq 0$; and F_2 - realization of design goal to reduce the number of attractors. The fitness $\mathcal{F}$ of an MACA in a population is determined by weighted mean of two factors - F_1 and F_2.

Determination of F_1. The power of classification is determined non- deterministically. As already mentioned in Section 2, X is the set formed from modulo-2 summation of each pattern of P_1 with each pattern of P_2. With 30 randomly chosen patterns from X, we test fraction of X satisfying the Relation 1 (Section 3.2): $T^d X \neq 0$. The fitness criteria F_1 of the MACA is determined by the percentage of patterns satisfying the Relation 1. For example, if 15 out of 30 chosen patterns from X satisfies Relation 1, $F_1 = 0.5$.

Determination of F_2. If the number of attractor basins required to classify the pattern set is less, then the fitness F_2 will be better. So F_2 has been defined as -

$$F_2 = 1 - [(m-1)/n]^l \tag{8}$$

where 2^m denotes the number of attractor basins for the n cell CA, and l is equal to 1·8. The value of l is set empirically.

Subsequent to extensive experimentation, we have fixed up relative weightage of F_1 and F_2 to arrive at the following empirical relation for fitness function

$$\mathcal{F} = 0.8 \cdot F_1 + 0.2 \cdot F_2 \tag{9}$$

The experimental results reported next confirm that this relation, although evolved empirically, provides the desired direction of GA evolution to arrive at the MACA for a two class classifier.

4.6 Performance Analysis

To analyze the performance of MACA based classifier, we perform extensive experiments based on randomly generated pattern set for different values of n (number of bits in a pattern). For each value of n, 2000 patterns are taken for each class and 10 different pairs of pattern sets are built. The algorithm has been coded in C language on a SUN with Solaris 5.6, 350 MHz clock. For a given data set of two classes, the MACA based two class classifier, as reported in Table 1, correctly classifies the data set.

Table 1. Performance of MACA based Classifier

Value of n	Value of m	Accuracy (%)	Max^m Gene^n	Evolution Time (min)
10	1	99.97	15	2.34
20	2	99.64	22	12.54
30	3	98.96	35	27.09
40	3	97.64	52	35.34
50	3	98.71	67	42.37
60	4	98.08	80	59.11
70	4	97.43	85	77.32
80	5	98.07	105	89.07
90	5	96.61	120	102.28
100	5	97.13	138	128.03

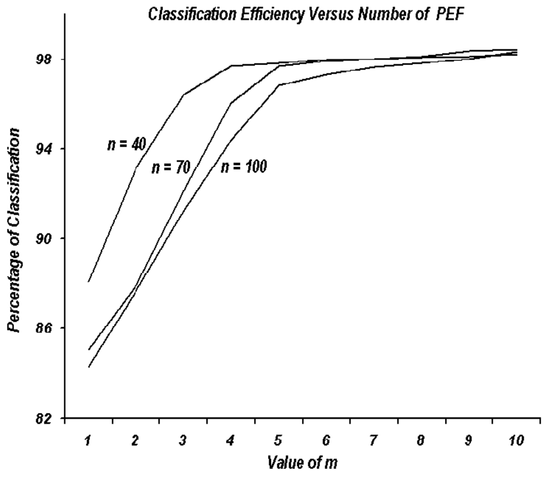


Fig. 12. Classification accuracy at different values of m for n= 40, 70, and 100

The GA starts with various values of m (the number of bits in PEF). The value of m, as noted in Theorem 1, dictates the number of attractor basins 2^m in the selected MACA. The design goal, as noted in Section 3, is to reduce the number of attractor basins while satisfying the Relation 1 for two class classifier. The GA parameters, as reported in Section 4.5, are formulated for fast convergence to certain region of m values. In each case, 80% of the population in the final solution assumes a particular value of m.

Columns I and II of Table 1 represent the number of bits (n) and the value of m (the number of bits in PEF). The classification accuracy of a data set is reported in Column III. Classification accuracy of a data set is the percentage of data which can be correctly classified in different attractor basins. Columns IV and V provides the maximum number of generation and the evolution time required to arrive at the desired MACA through GA evolution. The classification accuracy is above 97% in most of the cases which validates the theoretical foundation that MACA evolved through the proposed GA formulation acts as an excellent pattern classifier of two classes. This shows that our scheme of GA is able to hit the correct zone of classification, and the value of m where classification efficiency and design overhead is optimum. We also confirm that even if we increase the value of m than the one reported in Column II of Table 1, the classification accuracy improves at a very slow gradient. This has been depicted in Fig. 12. Further, as noted in Column V, the growth of GA evolution time to converge to the desired MACA with larger value of n is linear in nature.

5 Fault Diagnosis of Electronic Circuits

In this section, we perform fault diagnosis of electronic circuits with MACA based pattern classifier reported in Section 4. Fault diagnosis in both digital and analog circuits have been reported. While Section 5.1 reports the diagnosis results for digital circuits, same for analog circuits are covered by Section 5.2.

5.1 Digital Circuit Fault Diagnosis

The fault diagnosis experimentation for digital circuit has been carried out on a large number of benchmarks (ISCAS) for single stuck-at faults. A CUT is simulated with the fault simulator Hope [15]. The PO cones generated with Hope provide the basis of partitioning of the CUT. Hope extracts the PO cones to construct the fault list of a CUT. The fault list constructed by Hope reflects the spatial locality of wires in the CUT stuck at 1 or 0. The maximum length CA is used to generate the signature of a fault in the CUT. The experimental setup has been simulated in SUN with Solaris 5.6, 350 MHz clock.

Table 2 depicts the test environment and the circuit description. The first two columns (I and II) specify the benchmarks along with the number of internal lines, and primary input (PI), output (PO) lines. Columns III and IV display the number of collapsed faults and the number of unique faulty signatures of the CUT. The fault coverage figures noted in Column V resulted by applying a specific number of test patterns generated from a pseudo random pattern generator. The fault diagnosis results are presented in Table 3. The percentage of diagnosis achieved by the proposed scheme and with the fault dictionary method are noted in Column III. The percentage fault diagnosis is computed as

Table 2. Test Results of Diagnosis in Dictionary Method

CUT	# PI/PO	# Collapsed Faults	# Unique Signatures	Fault Coverage
c880	60/26	942	896	98.62
c1908	33/25	1879	1801	98.99
c3540	50/22	3428	2838	96.00
c1355m	41/32	2278	1664	92.32
c1355	41/32	1574	912	98.48
c6288	32/32	7744	7648	99.50
c499m	41/32	902	776	96.01
c499	41/32	758	699	97.62
c2670	233/140	2747	2332	84.27
c7552	207/108	7550	7053	94.71
s967	16/23	1066	993	98.22
s991	65/17	910	776	95.27
s713	35/23	581	388	81.24
s3384	43/26	3380	2810	91.48
s4863	49/16	4764	4123	93.32
s3271	26/14	3270	2585	99.39
s5378	35/49	4603	2416	66.43
s6669	83/55	6684	6358	100
s641	35/24	467	359	85.22

Table 3. Test Results of MACA based Fault Diagnosis

CUT (n, p)	# Partition	MACA	Dictionary	MSR (%)
c880 (26, 896)	6	98.71	95.62	1.8243
c1908 (25, 1801)	6	98.83	96.03	0.9106
c3540 (22, 2838)	8	97.73	86.52	0.8184
c1355m (32, 1664)	6	98.10	78.94	0.9672
c1355 (32, 912)	6	97.75	58.84	1.7646
	8	96.71		2.4705
	10	93.55		3.1764
c6288 (32, 7648)	6	99.72	99.33	0.2104
c499m (32, 776)	10	99.31	91.72	3.7331
c499 (32, 699)	6	97.09	93.20	2.3024
c2670 (140, 2332)	10	100	99.19	1.2046
c7552 (108, 7053)	6	98.96	98.81	0.2238
	8	98.57		0.3134
	10	97.67		0.4029
s967 (23, 993)	4	98.28	94.08	0.9983
s991 (17, 776)	4	97.54	89.93	1.3189
s713 (23, 388)	4	93.62	81.85	2.5549
s3384 (26, 2810)	10	94.74	81.64	1.0471
s4863 (16, 4123)	10	92.43	89.63	0.7504
s3271 (14, 2585)	6	96.71	79.54	0.6648
	8	97.05		0.9308
	10	94.13		1.1968
s5378 (49, 2416)	4	95.03	76.21	0.3902
	8	98.06		0.9106
	10	92.35		1.1707
s6669 (55, 6358)	10	99.94	95.12	0.4426
s641 (24, 359)	6	95.86	88.86	4.5845

$$\%\text{ fault diagnosis} = \frac{\text{number of faults diagnosed}}{\text{number of detected faults}} \tag{10}$$

For the sake of illustration of diagnosis scheme, each of the circuits has been partitioned into a number of modules noted in Column II of Table 3. Diagnosis is done to the level of faulty module in the CUT. We have reported results for 4, 6, 8, 10 partitions for a CUT. Each partition represents a module. In Column IV, the memory requirement to implement the MACA based scheme is reported as a ratio of that required for diagnosis based on fault dictionary (Relation 5, Section 3.6). For majority of the CUTs, the memory overhead is only a small percentage of the memory consumed by fault dictionary based scheme. The diagnosis quality (Column III) and memory overhead figures (Column IV) establish the superiority of the proposed scheme.

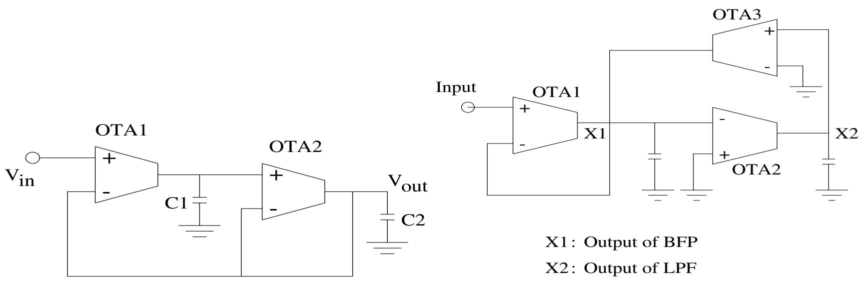


Fig. 13. OTA realization of low pass filter and band pass filter

5.2 Analog Circuit Fault Detection

In [10] a scheme is reported for synthesis of operational transconductance amplifier (OTA) based analog circuits to realize different classes of Filters. We have done experiment for diagnosis of faulty OTA and capacitor of such circuits.

The practical inability of testing analog circuits with desired precision demands the definition of a tolerance band within which the circuit functions correctly. This is particularly true when a parameter of the analog circuit (for example, transconductance of an OTA) cannot be tunned precisely as per the design requirement. As a result, test and diagnosis of an analog circuits has become a challenging problem. In general, test and diagnosis of an analog system demands analysis of a large volume of a test response data within the tolerance band [16]. We report the experimental setup for generation of faulty and fault free signatures for an analog circuit.

Generation of Signatures. The set of signatures are derived out of the following data sets: $F_{good[i]}$ - data set of the good circuit for the i^{th} performance parameter, $i = 1, 2, \cdots, p$, where p denotes the maximum number of performance parameters considered; and $F_{fault[ij]}$ - data set of the circuit by introducing fault at j^{th} device for the i^{th} performance parameter. $j = 1, 2, \cdots, K$, where K is the number of components (OTA and capacitor) in the circuit. The discrete set of data - $F_{good[i]}$ and $F_{fault[ij]}$ has very large volume. It is compressed to a set of signatures $S = [S_0, S_1, \cdots, S_K]$, where S_0 represents the signature set for non-faulty circuit and S_x (for $x = 1, 2, \cdots, K$) denotes the signature set generated by introducing fault at x^{th} component, K being number of components. To generate the signature for the tolerance band (non-faulty region) and faulty region, a device parameter is divided into q number of equal steps. The corresponding signature is given as $S_x = [S_{(x)1}, S_{(x)2}, \cdots, S_{(x)q}]$, where $S_{(x)j}$ $(j = 1, 2, \cdots, q)$ denotes the signature generated at the j^{th} step of x^{th} component. The complete sequential steps for the signature generation are noted in [16].

The signature sets generated by the above procedure are partitioned into two classes - faulty and non-faulty. Faulty class is next divided into K number of sub-classes, each sub-class corresponds to signature set generated by introducing fault

Table 4. Fault Diagnosis Analysis of LPF and BPF of Fig. 13

Components	# Samples	Detected	Not Detected	Success Rate
LPF/OTA1	8970	8962	8	99.91
LPF/OTA2	5430	5424	6	99.88
LPF/C1	5430	5421	9	99.83
LPF/C2	5610	5603	7	99.87
Components	# Samples	Detected	Not Detected	Success Rate
BPF/OTA1	7201	7193	8	99.89
BPF/OTA2	4321	4306	15	99.65
BPF/OTA3	4441	4436	5	99.88
BPF/C1	4297	4181	116	97.30
BPF/C2	4321	4219	102	97.64

in a single component. MACA based two class classifiers have been synthesized at each level of diagnosis (as shown in Fig. 4 - MACA based multi-class classifier).

Experimental Results. In this subsection, we demonstrate the efficiency of the diagnosis procedure with two example circuits. Fig. 13 shows the OTA realization of a low pass filter (LPF) and a band pass filter (BPF) respectively. The fault diagnosis results are shown in Table 4. Column I shows the circuit components - OTAs and capacitors. Column II specifies the number of sample values picked up from the output signal of the analog circuit. Digital signature is generated out of these digitized sample values. Faults detected under this experimental set up is shown in Column III. The overall success rate of diagnosis of faulty component (OTA) is more than 99%. In both cases, the values of MSR are 0.0372% and 0.0468% respectively according to Relation 5 of Section 3.6.

6 A Generic Fault Diagnosis Machine

A two class classifier is basically an MACA evolved through GA evolution. A multi-class classifier, as noted in Fig. 4, is a set of MACA. The design of the set of MACA based classifier for a circuit is an one time cost, which is incurred due to the computational overhead of GA evolution reported in Section 4.6. The growth of this overhead is linear in nature.

The circuit shown in Fig. 14 depicts a programmable CA (PCA) cell. A PCA n programmable cells can be configured/programmed to realize any n-cell linear CA corresponding to a specified rule vector. A PCA with associated control structure (Fig. 14) can in effect behave as a generic FDM. For a typical case of Fig. 4, the FDM needing 3 MACA can be realized from the structure shown in Fig. 14. Next we briefly elaborate the functions of each block of Fig. 14.

- Block 1 - Store the input pattern.
- Block 2 - Store all the rules (T) and corresponding depths.
- Block 3 - Store rule and depth to be feed in.
- Block 4 - Store the attractors and class information for all rules.

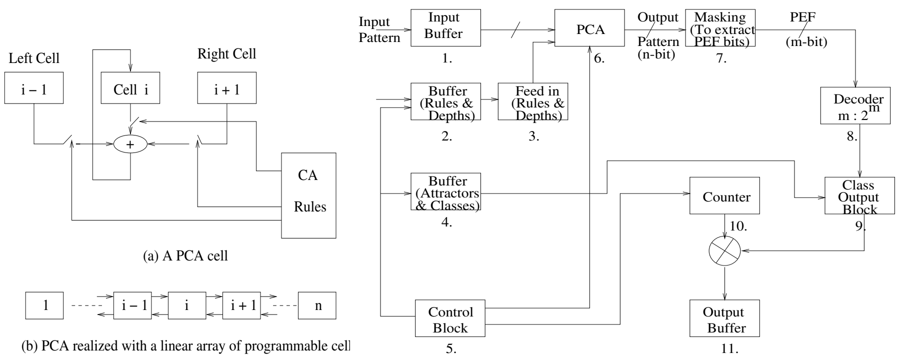


Fig. 14. Structure of a PCA cell and a generic fault diagnosis machine (FDM)

- Block 5 - Generate the control signal.
- Block 6 - PCA generates outputs corresponding to rules.
- Block 7 - Generate pseudo-exhaustive bits (PEF).
- Block 8 - Select a particular attractor based on the PEF bits.
- Block 9 - Give the class information based on attractors.
- Block 10 - Counter for K class classifier.
- Block 11 - Store the final output class.

7 Conclusion

This chapter presents an elegant method to classify a large volume of data set into distinct classes. The proposed classifier, build around a special class of CA referred to as MACA, ensures fast identification of faulty signatures generated from electronic circuit. The MACA effectively provides an implicit storage mechanism of voluminous response data and thereby providing the solution to the problem of handling large data set. The excellent diagnostic resolution and the memory overhead figures prove the superiority of the MACA based design over that of existing scheme with conventional fault dictionary.

References

Abramovici, M., Breuer, M., Friedman, A.: Digital Systems Testing and Testable Design. IEEE Press, Los Alamitos (1990)

Agrawal, V.D., Mercer, M.R.: Testability Measures - What Do They Tell Us? In: Proc. International Test Conference, pp. 391–396 (1982)

Bardell, P., McAnney, W., Savir, J.: Built In Test for VLSI: Pseudorandom Techniques. Wiley Interscience, Chichester (1987)

Pomeranz, I., Reddy, S.M.: On The Generation of Small Dictionaries for Fault Location. In: ICCAD, November 1992, pp. 272–279 (1992)

Ryan, P.G., Fuchs, W.K., Pomeranz, I.: Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits. In: ICCAD, November 1993, pp. 508–511 (1993)

Boppana, V., Fuchs, W.K.: Fault Dictionary Compaction by Output Sequence Removal. In: ICCAD, November 1994, pp. 576–579 (1994)

Sikdar, B.K., et al.: Design of Multiple Attractor $GF(2^p)$ Cellular Automata for Diagnosis of VLSI Circuits. In: Proceedings of 14^{th} International Conference on VLSI Design, India, pp. 454–459 (2001)

Paul, K.: Theory and Application of Multiple Attractor Cellular Automata for Fault Diagnosis. In: Proceedings of Asian Test Symposium (December 1998)

Sikdar, B.K., et al.: Multiple Attractor Cellular Automata for Hierarchical Diagnosis of VLSI Circuits. In: Proceedings of 10^{th} Asian Test Symposium, Japan (2001)

Roy, B.N., Chaudhuri, P.P., Nandi, P.K.: Efficient Synthesis of OTA Network for Linear Analog Functions. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 21(5), 517–533 (2002)

Neumann, J.V.: The Theory of Self-Reproducing Automata. In: Burks, A.W. (ed.), University of Illinois Press, Urbana, London (1966)

Wolfram, S.: Theory and Application of Cellular Automata. World Scientific (1986)

Chaudhuri, P.P., et al.: Additive Cellular Automata, Theory and Applications, vol. 1. IEEE Computer Society Press, Los Alamitos, California (1997)

Chattopadhyay, S., et al.: Highly Regular, Modular, and Cascadable Design of Cellular Automata-Based Pattern Classifier. IEEE Transaction on VLSI Systems 8(6), 724–735 (2000)

Lee, H.K.: Hope: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits. In: Proceedings of 29^{th} Design Automation Conference, pp. 336–340 (1992)

Roy, B.N., Chaudhuri, P.P., Nandi, P.K.: Test Solution for OTA Based Analog Circuits. In: Proceedings of 7^{th} Asia South Pacific Design Automation Conference and 15^{th} International Conference on VLSI Design, January 2002, Bangalore, India, pp. 773–778 (2002)

A Survey of Artificial Neural Network-Based Modeling in Agroecology

Jiménez Daniel[1,3,4], Pérez-Uribe Andrés[3], Satizábal Héctor[2,3,4], Barreto Miguel[2,3,4], Van Damme Patrick[1], and Tomassini Marco[2]

[1] Ghent University, Faculty of BioScience Engineering: Agricultural Science
[2] Université de Lausanne, Hautes Etudes Commerciales (HEC), Institut des Systèmes d'Information (ISI)
[3] REDS Institute, University of Applied Sciences of Western Switzerland (HEIG-VD)
[4] BIOTEC. Precision Agriculture and the Construction of Field-Crop Models for Tropical Fruit Species

1 Introduction

Agroecological systems are difficult to model because of their high complexity and their nonlinear dynamic behavior. The evolution of such systems depends on a large number of ill-defined processes that vary in time, and whose relationships are often highly non-linear and very often unknown. According to Schultz et al. (2000), there are two major problems when dealing with modeling agroecological processes. On the one hand, there is an absence of equipment able to capture information in an accurate way, and on the other hand there is a lack of knowledge about such systems. Researchers are thus required to build-up models in rich and poor-data situations, by integrating different sources of data, even if this data is noisy, incomplete, and imprecise.

In order to model an agroecological system, we can proceed by considering the modeling problem as a regression or a classification problem. For instance, we deal with a regression problem when modeling natural processes such as crop yield, climate and physiological variables, vegetation dynamics, greenhouse conditions, severity of a given pest and/or disease, etc., given that the dependent variables are continuous. On the other hand, when we deal with a classification problem when we want to model phenomena such as environmental variability, yield quality and quantity, genetic variation, soil properties, land cover, etc., given that the dependant variables of the system are categories, and that the main idea consists on assigning the same class to individuals with similar features (i.e., by forming groups).

Artificial neural networks have proved to be a successful approach for modeling agricultural systems by considering them either as regression or classification problems (Hashimoto, 1997, Schultz and Wieland, 1997, Schultz et al., 2000). Thus, these techniques can be regarded as an alternative modeling approach to traditional statistics, in particular when dealing with the highly variable, noisy, incomplete, imprecise and qualitative nature of agroecological information. Such techniques have shown to be capable of "learning" nonlinear models including qualitative and quantitative information, and in general, they have shown better pattern recognition capabilities than traditional linear approaches (Murase, 2000, Schultz et al., 2000, Noble and Tribou, 2007). During the last twenty years or so, researchers have acquired a lot of experience using artificial neural network network–based models. This chapter presents a survey of artificial neural network modeling applications in agroecology in an attempt to guide future researchers and practitioners involved in this domain.

We start illustrating the artificial neural network–based modeling approach by describing two examples of our own work on the domain of sugarcane yield modeling, and finally we present an extensive survey of modeling agroecological systems by means of artificial neural networks.

2 Application Examples of Artificial Neural Network-Based Modeling

An artificial neural network (Bishop, 1995) is a computational structure where many simple computational elements, called artificial neurons, perform a nonlinear function of their inputs. Such computational units are massively interconnected and are able to model a system by means of a training algorithm. This algorithm attempts to minimize an error measure that is computed in different ways depending on the specific technique used to adjust the connections (i.e., the learning algorithm). There are two major approaches to train an artificial neural network (i.e., to adapt its parameters): supervised and unsupervised learning. In the *supervised learning* approach, specific examples of a target concept are given, and the goal is to learn how to recognize members of the class or to build a regression model using the description attributes. In this case, the synaptic weights among neurons are adjusted in order to minimize the error between the known desired outputs and the actual output given by the neural network during the learning process. In the *unsupervised learning* approach, the set of examples is provided without any prior classification, and the goal is to discover underlying regularities and patterns, most often by identifying clusters or subsets of similar examples. Training in this case consists on looking for a compressed representation of the collected examples (original data) and the error is the difference between this representation of our original data and the original data (Bishop, 1995).

In this section, we describe two example applications of artificial neural network–based modeling in agriculture. The particular problems we are presenting are sugarcane yield prediction and sugarcane yield data exploration. These example applications are intended to help the researcher from the agroecological domain, to better realize the capabilities and the potential of the artificial neural network techniques, both for modeling and for providing a new way to explore and analyze the data.

2.1 Modeling Sugarcane Yield: A Supervised Approach

In order to increase crop yield, it is important to know the environmental conditions that lead to a higher productivity. This is a very important matter for the sugar industry that nowadays extends its interests towards organic fuels. In the experiments described in this section we attempt to relate crop yield to a group of climatic variables that we assessed as being relevant for the plant development. To achieve this goal, we used a feed-forward multilayer Perceptron that we trained using the Backpropagation algorithm (Bishop, 1995) in order to make a non-linear regression.

The training set we used is a database provided by a sugarcane research center in Colombia (CENICAÑA) that gathers information from different sugar refineries in the south-western region of the country. This database contains information of one year (2005) about the crop yield and climate conditions. These variables were *Temperature* (maximum, minimum, average and range), *Relative Humidity* (maximum, minimum, average and range), *Radiation* (average) and *Precipitation* (accumulated). *Plant age* and *Water balance* are also available for each zone of production. The growing period of sugarcane is about thirteen months. During this time, the plant is exposed to different climatic conditions, and all these stimuli probably affect crop yield. Expert knowledge converges to assert that the most relevant periods are the beginning and the end of plant development. The first months are essential for vegetative structure formation whereas during the last months the plant accumulates the major part of saccharose. In our modeling framework, we considered as inputs, the weekly average of the climate information of the last four months before harvest and the first four months after the preceding harvest (Jiménez et al., 2007, Satizábal et al., 2007). The output target value of the model was the cumulative yield in terms of tons of sugarcane per hectare. The best neural network configuration was found after varying the number of hidden units and finally, it was a multilayer Perceptron with a single hidden layer of ten units that gave the best results. In order to obtain a general model, a group of 100 neural networks was trained using the Backpropagation algorithm, and in every case the relevance of each input was assessed (Jiménez et al., 2007). The variables ordered by relevance are shown in figure 1.

The analysis of input relevance shows that plant age, water balance and relative humidity of the different weeks are the most important variables for the construction of the model.

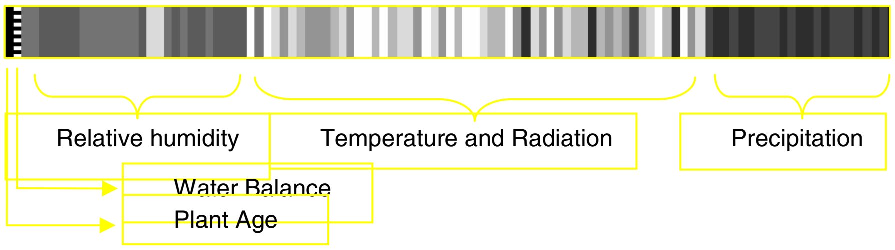


Fig. 1. Input variables of the neural network-based model ordered by relevance (the most relevant variables are shown on the left)

Figure 1 also shows that the precipitation is the less relevant agroecological variable, perhaps, because of artificial irrigation.

Figures 2a and 2b show two profile graphics plotted using the two most relevant variables. Figure 2a shows that sugarcane becomes heavier when the plant age increases, which is quite logic if we consider that sugarcane is a plant that continuously grow until reaching a maximal height. Conversely, figure 2b shows a decrease on

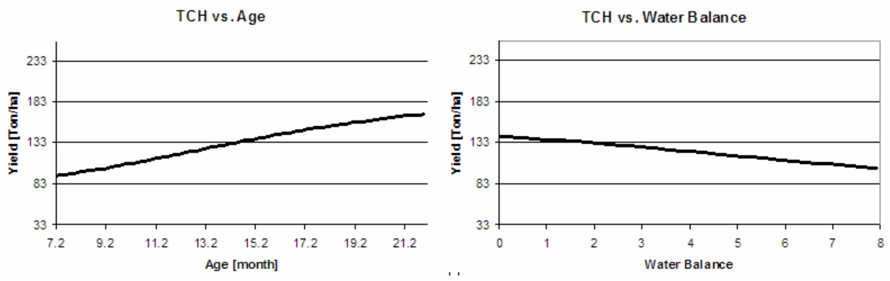


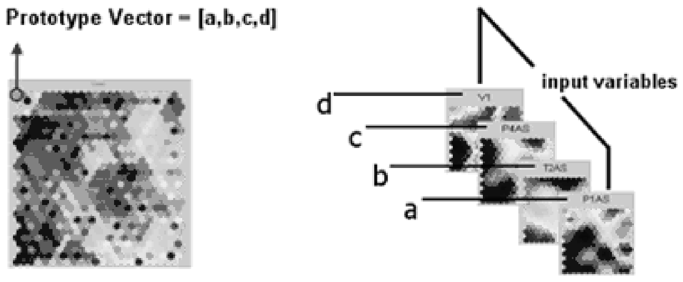
Fig. 2. a) Input profile for the variable *plant age* using a multilayer Perceptron. **b)** Input profile for the variable *water balance* using a multilayer Perceptron.

weight when there is more water availability. This relationship cannot be easily explained, as in the first case.

2.2 Modeling Sugarcane Yield: An Unsupervised Approach

This second case study illustrates the use of an unsupervised learning approach to modeling in agriculture. The main objective is to obtain new insights about the common factors that determine productivity similarities in the sugarcane culture of the southwest region of Colombia. To achieve this, we used a 6-year (1999 to 2005) sugar cane database containing 1328 observations, each relating 54 agroecological variables to the crop yield, in order to train a Self-Organizing Map (SOM). A SOM (Kohonen, 1997) is an unsupervised clustering technique that is able to map high dimensional data to a low dimensional space (reduction of dimensionality). When using a SOM, the data points that are close in the high-dimensional space remain close in the low-dimensional space representation, thus providing an original technique for visualizing high-dimensional data.

SOMs have proved to be effective for the exploratory analysis of agro-ecological data and have become a powerful technique widely used in ecological modeling (Liu et al., 2006). They are very useful when it is essential to extract features from a



(a) (b)

Fig. 3. a) Self-organizing map. Each neuron i represents an n-dimensional prototype, where n is equal to the dimension of the input space, in this case n = 4. **b)** Component planes. Each component plane represents one input variable.

complex data set (Chon et al., 1996, Giraudel and Lek, 2001) and produce easily comprehensible low dimensional maps that improve the visualization of the data, and facilitate data interpretation. Last but not least, they are able to work with incomplete data sets (Samad and Harp, 1992), noisy data, and appear to be less sensible to outliers than other clustering methodologies (Vesanto and Alhoniemi, 2000).

More important is the possibility of slicing the Self-organizing maps in order to visualize their so-called component planes (see Figure 3) in order to improve the analysis of the relationships between variables and/or their influence on the outputs of the system. Those component planes show the relative distribution of each input variable (Kohonen, 1997). Unlike the traditional method where the number of pair wise scatter plots increases quadratically with the number of variables, using such a Self-organizing map component planes-based visualization, the number of sub-plots grows linearly with the number of variables (Himberg, 1998) (see Figure 3). In addition, it is possible to cluster variables with a similar pattern. After plotting all component planes, the relationship between variables can be easily observed. The task of organizing similar components planes in order to find correlating components is called correlation hunting (Vesanto, 1999). However, when the number of components is large it is difficult to determine which planes are similar to each other. Different techniques can be used to reorganize the component planes in order to perform the correlation hunting. The main idea is to place correlated components close to each other (Barreto and Pérez-Uribe 2007).

3 Survey of Artificial Neural Network-Based Modeling Applications in Agriculture

In this section, we present an extensive survey of research and application articles reporting the use of artificial neural networks in agriculture. To ease the access to this information, we have classified the literature by application in the domain of agriculture. The applications are about grading fruits, climate forecast, weed control, pest and disease management, yield prediction, natural resources management, irrigation and fertilization, physiology, greenhouse control and soils. We start by describing the application domain and then we give some details of the contribution using tables, see table 1.

Table 1. Explanation of table content

Prob.	The particular problem that has been worked out
Input	The data they have used as inputs to the models
Alg.	The techniques and algorithms they have used
Res.	The main results of the contribution
Ref.	The reference of the work

3.1 Grading Fruits

Grading fruits is an important operation after harvesting. Fruit identification is commonly a manual task carried out by workers based on their experience and empirical

Table 2. Applications of artificial neural networks in grading fruits

Prob.	To grade apple color
Input	Image data
Alg.	Improved Backpropagation
Res.	Successful classification of the quality
Ref.	(Nakano, 1997)
Prob.	To grade oranges (*Miyauchi iyokan*) according to their acid or sugar content
Input	Color, shape, roughness of skin surface, and weight
Alg.	The Kalman filter algorithm
Res.	The sweet (and not the acid) oranges were those that were reddish, low height, glossy and medium sized
Ref.	(Kondo et al., 2000)
Prob.	To classify strawberry varieties
Input	Different chemical signatures associated to growing year, the place of production and the fresh and frozen state of samples
Alg.	Self-Organizing Map (SOM)
Res.	Growing year was one of the most relevant factors for inducing variability
Ref.	(Boishebert et al., 2006)
Prob.	To detect defects on sweet cherries
Input	Spectral information of different tissues of cherries
Alg.	EGAN (feed-forward artificial neural network)
Res.	Accuracy of 73% in classifying sweet cherry defects
Ref.	(Guyer and Yang, 2000)

knowledge. However, these methodologies have some drawbacks such as inconsistency, cost, subjectivity and tediousness. Artificial neural network approaches have been used in order to overcome these drawbacks. In table 2, we show some examples of grading fruits according to external or internal features.

3.2 Climate in Agriculture

Climate is one of the most important components of agricultural systems. This component influences many agroecological processes and variables, affecting soil properties, pest and disease dynamics, agricultural practices, production, yield, etc. The variability in the yield in rain-fed agricultural production systems can be attributed to weather differences up to 90% (Hoogenboom et al., 2000). Therefore, it is desirable to develop accurate climate models capable of estimating climatic variability and its impact on agricultural systems or the environment, in an attempt to support the decision making process in agriculture. The following table provides some examples of climate modeling in agricultural systems.

Table 3. Applications of artificial neural networks in climate domain

Prediction of climate variables	
Prob.	To forecast solar radiation
Input	Daily values of extraterrestrial radiation, daily temperature range, precipitation, cloudiness and relative sunshine duration
Alg.	Backpropagation
Res.	The modeling process by means of neural networks successfully estimated solar radiation
Ref.	(Bocco et al., 2006)
Prob.	To estimate daily maximum and minimum air temperature and total solar radiation
Input	Maximum air temperature, daily minimum air temperature, daily total solar radiation, difference in elevation, difference in directions and the straight line between the target and input station location
Alg.	Backpropagation, traditional spatial analysis
Res.	Neural network models were more accurate on estimating maximum and minimum temperatures and solar radiation for a single location
Ref.	(Li, 2002)
Prob.	To analyze trends in rainfall
Input	Rainfall data corresponding to a period from 1893 to 1933
Alg.	Backpropagation - Adaptive basis function network (ABFNN)
Res.	ABFNN performed better than Backpropagation in predicting long–term rainfall behavior
Ref.	(Philip and Joseph, 2003)
Prob.	To estimate ozone concentrations
Input	
Alg.	Self-Organizing Map (SOM), two-stage neural networks, multiple linear regression, two-level clustering, multilayer Perceptron neural networks
Res.	The two-stage neural network had the best performance, explaining at least 60% of ozone concentration variance
Ref.	(Chung Lu et al., 2006)
Forecasting temperature and critical duration period, which affect plant growth	
Prob.	Frost forecast in peaches during critical periods of growth
Input	Temperature, relative humidity, rainfall, wind speed and solar activity
Alg.	Feed-forward neural network with different activation functions, trained with Backpropagation
Res.	The best frost prediction was achieved using relative humidity, solar activity and wind speed from 2 to 6 hours before the frost event
Ref.	(Jain, 2003)

Table 3. (*continued*)

Studying the impact of climate change on the potential distribution of vegetation	
Prob.	To model the distribution of vegetation in past, present and future climates, in tropical Forests
Input	Seven climate variables, nine soil parent material classes and seven terrain classes
Alg.	Backpropagation
Res.	Certain locations were occupied by a forest class in some climates while others occupied the same class despite changes in local climate
Ref.	(Hilbert and Ostendorf, 2001)
Prob.	To Predict functional traits of the ecosystem
Input	Data regarding six functional traits derived from the normalized difference vegetation index (NDVI)
Alg.	Backpropagation, regression models
Res.	The correlation between predicted and observed values for each functional trait was higher for the model developed with the artificial neural network than when using a regression model
Ref.	(Paruelo and Tomasel, 1997)

3.3 Weed Control

Weed control is an essential procedure in agriculture; weeds compete with commercial crops in terms of space, nutrients, water, and light. Weed control generally involves spraying herbicides which is an undesirable practice due to its negative environmental impact, and because of the cost. Recently, farmers and researchers have been interested in minimizing environmental impacts and reducing the costs provided by weed control. Neural network classifiers have been shown to be a useful tool for discriminating between different kinds of weeds exploiting a database of pictures. Some works have used this approach to develop a site-specific weed management system which helps farmers to spray herbicides in a selective way, thereby decreasing the amount of these substances in the crops and decreasing costs. The following table shows several articles dealing with weed control; most of them focused on recognizing weeds in field; however a particular case, where weed seeds were recognized in a crop seeds selection process, is also shown.

Table 4. Applications of artificial neural networks in weed control

Identification of weeds in field	
Prob.	To classify weed species in crop fields
Input	Image data set of texture features
Alg.	Backpropagation, Counterpropagation, Radial Basis Function networks
Res.	The feed-forward neural network trained with the Backpropagation algorithm had the best classification performance
Ref.	(Burks et al., 2005)

Table 4. (*continued*)

Prob.	To differentiate sunflower from weed and soil
Input	Images taken between two and three weeks after planting the sunflower
Alg.	Backpropagation
Res.	The maximum number of correct differentiation of weeds from sunflower plants was 71 (out of 86), 82 in separating sunflower from bare soil and 74 in distinguishing weed images from bare soil
Ref.	(Kavdır, 2004)
Prob.	Site-specific herbicide applications using automatic weed localization
Input	Digital images of more than 30 common weeds present in the area
Alg.	neural networks and fuzzy logic
Res.	With this approach, site-specific weed densities could be determined and thus herbicide applications could be managed, monitored
Ref.	(Yang et al., 2003)
Prob.	To recognize weeds and seedlings of carrot (*Daucus carota* L.)
Input	Measures of leaf shape,Digital images
Alg.	Self-Organizing Map (SOM)
Res.	Neural networks approach discriminated species without predefined plant descriptions, however this method required a previous image processing
Ref.	(Aitkenhead et al., 2003)
Prob.	To identify weed species from corn (*Zea mays* L.) and sugar beet (*Beta vulgaris* L) crops
Input	Spectral proprieties
Alg.	Backpropagation, probabilistic neural network , learning vector quantization, Self-Organizing Maps, local linear mappings
Res.	With Local Linear Mapping methodology, it was obtained an identification accuracy of 90%
Ref.	(Moshou et al., 2001, 2002)
Recognition of weed seeds in a crop seeds selection process	
Prob.	To discriminate weed species present in the commercial seed production industry in Argentina
Input	Image dataset of morphological, color and textural features
Alg.	Backpropagation
Res.	Morphology was the most important feature for identifying weed seeds
Ref.	(Granitto et al., 2000)

3.4 Pest and Disease Management

Pest and disease management is an essential component of agricultural system management. Both, pest and disease damages to crops are known to be important factors in yield losses. These yield losses are commonly tackled with strategies which involve

Table 5. Applications of artificial neural networks in pest and disease management

	Diseases
Prob.	To predict the severity of gray leaf spot of maize (*Cercospora zeae-maydis*) in corn (*Zea mays* L.)
Input	Environmental, cultural, and location-specific data, temperature, relative humidity, surface wetness, cumulative hours of surface wetness and of daily temperature, cumulative hours of nightly relative humidity
Alg.	Backpropagation
Res.	The best variables predicting severity were hours of daily temperatures and hours of nightly relative humidity
Ref.	(Paul and Munkvold, 2005)
Prob.	To forecast diseases in wheat (*Triticum* spp)
Input	Environmental and aerobiota data
Alg.	Backpropagation, logistic regression and multivariate linear discrimination
Res.	Neural network and statistical models showed similar performance
Ref.	(Francl, 2004)
Prob.	To detect the development of "yellow rust" (*Puccinia striiformis* f. sp. *Tritici*,) in wheat (*Triticum* spp cv. Madrigal). To compare a hybrid neural network + spectral refection method with a fluorescence remote sensing system
Input	Spectral images (wavebands)
Alg.	Self-Organizing Map (SOM)
Res.	The waveband centered at 861 nm was the variable which best discriminated healthy from diseased leaves. The hybrid approach showed the best performance
Ref.	(Moshou et al., 2004, 2005)
Prob.	To detect and classify Phalaenopsis (*Phalaenopsis* spp.) seedling diseases:bacterial soft rot (*Erwinia chrysanthem),* bacterial brown spot (*Burkholderia cattleyae)* Phytophthora black rot (*Phytophthora parasitica* and *Phytophthora palmivora)*
Input	Images of 18 texture features of the lesion area and the RGB color features
Alg.	Backpropagation
Res.	Phalaenopsis seedling diseases were efficiently detected and classified
Ref.	(Huang, 2007)
	Pests
Prob.	To predict pest pod borer (*Helicoverpa armigera*) attack on chickpea (*Cicer arietinum* L.)
Input	Data of climate, location and pest incidence: date, standard week, minimum and maximum temperature, humidity, rainfall, larvae/plant, eggs/plant, light and pheromone trap catchs, zone, location, season, area surveyed, plant protection type
Alg.	Bayesian Regularization + Levenberg-Marquardt algorithm
Res.	Pest attack activities were successfully predicted for one week in advance, by using weather and pest surveillance data
Ref.	(Gupta et al., 2003)

Table 5. (*continued*)

Prob.	To predict the presence or absence of greater flamingo (*Phoenicoterus rubber roseus*) damages in rice fields in the Mediterranean
Input	Landscape features of rice paddies
Alg.	Backpropagation
Res.	The neural networks successfully predicted flamingos' incursions from a reduced set of ecological variables
Ref.	(Tourenq et al., 1999)

the use of a range of pesticides, thereby increasing production costs and increasing the danger of toxic residual levels on agricultural products. Correct pest management requires an accurate identification of pests and diseases; it requires knowledge about their cycles, as well as the effect of environmental variables on pest and disease development and population dynamics. Some models based on artificial neural networks have been developed to support this management, in an attempt to support decision makers in agriculture to avoid unnecessary pesticide applications, and to identify various phenomena leading to yield losses. These models are based on the relationship between environmental conditions, and pest and disease impacts on the crops. In the following table, we show some examples of predicting pests and diseases in agricultural systems, by using different sources of data (environmental, images, cultural, etc.).

3.5 Crop Yield Prediction and Other Estimations

It has long been understood that farmer incomes increase or decrease depending on crop yields. In an attempt to support farmers in their decision-making processes, it is important to understand the existing relationships between the factors resulting in crop yield. These factors are extremely complex in time and space, and the success in management decisions lies in understanding the influence of soil, landscape, climate, genotype, water availability on crop yield, as well as identifying the moments or variables that could be modified by farmers in pursuit of improving their crop yields. The search for techniques capable of recognizing these influences is an essential first step into understanding and identifying these processes.

It has been shown that artificial neural networks are a powerful approach to tackle these problems. This technique has been used in predicting yield crops such as: corn, sugar beet, soybean and winter wheat, using databases of environmental, plant features, and hyperspectral images. Artificial neural networks have not only been employed in crop yield prediction but also in predicting the volume of pine barks, exploring the contribution of weather and other variables to some properties in winter cereal, as well as estimating the concentrations of pollutants in grass plant species. The following table presents a more detailed description of these works.

Table 6. Applications of artificial neural networks in predictions and estimations

	Corn (*Zea mays* L) and Soybean (*Glycine max*) yield prediction
Prob.	To predict corn and soybean yields
Input	Data from different locations, kind of soil and multiple combinations of monthly or weekly precipitation
Alg.	Backpropagation, multiple linear regression models
Res.	The neural network model showed the best yield prediction for both crops using data of weekly precipitation
Ref.	(Kaul et al., 2005)
	Corn (*Zea mays* L) yield prediction
Prob.	To study multiple site-years corn crop yields
Input	Topographic, climatological and soil properties data
Alg.	Several variations of Backpropagation (standard, batch, momentum, weight decay, quickprop, and resilient Backpropagation), forward search stepwise multiple linear regression, and progression pursuit regression
Res.	The performance of neural networks and linear methods was similar
Ref.	(Drummond et al., 2000)
Prob.	To identify the most important factors for corn yield and quality
Input	Data of soil (electrical conductivity, organic matter, pH, chemical elements availability), landscape (slope, elevation) and genetic (seed hybrid factors)
Alg.	Backpropagation
Res.	Genetic characteristics (seed hybrid) was the factor which best explained the variability of corn quality and yield variability
Ref.	(Miao et al., 2006)
Prob.	To predict maize yield under varying soil and land management conditions
Input	Land management practices and physical and chemical soil properties
Alg.	Linear models, artificial neural networks regression trees
Res.	Regression trees were the best prediction model of maize yield
Ref.	(Park et al., 2005)
Prob.	To predict corn yield
Input	Hyperspectral images, vegetation indexes
Alg.	Backpropagation, stepwise multiple linear regression
Res.	The neural networks and the stepwise multiple linear regression performed better than the models based only on vegetation indexes
Ref.	(Uno et al., 2005)
Prob.	To predict spatial corn yield variability
Input	Soil and landscape characteristics (fertility, elevation, electrical conductive-ity and satellite imagery)
Res.	Neural networks successfully predicted spatial yield variability using fertil-ity, elevation, electrical conductivity and spectral satellite image features
Ref.	(Shearer et al., 2000)

Table 6. (*continued*)

	Sugar beet (*Beta vulgaris* 'saccharifera') yield prediction
Prob.	To predict sugar beet crop yield
Input	Physical and chemical characteristics of plants
Alg.	Predictive modular network (PRENOM)
Res.	The PRENOM network performed better than previous prediction approaches on estimating sugar beet crop yield
Ref.	(Kehagias et al., 1998)
	Wheat (*Triticum aestivum* L.) yield prediction
Prob.	To predict dryland winter wheat (*Triticum aestivum* L.) grain yield by using topographic attributes
Input	Landscape topographic attributes
Alg.	Spatial analysis neural network algorithm, whereby a kernel function accounts for the influence of neighboring points in the input space
Res.	The spatial analysis neural network algorithm successfully estimated spatial patterns of dryland crop yield
Ref.	(Green et al., 2007)
	Predicting the volume of pine (*Pinus brutia*) barks
Prob.	To estimate volume of pine barks
Input	Measures of: outside-bark diameter at breast height, inside-bark diameter at breast height, outside-bark volume of the tree, and bark correction factor
Alg.	Kalman's learning algorithm for training and Cascade Correlation, nonlinear regression approaches (logistic mode, Gompertz, Metcherlich, Morgan, Mercer, Florin, Verhulst)
Res.	The neural networks using the mentioned algorithms successfully estimated pine bark volume
Ref.	(Diamantopoulou, 2005)
	Exploring the contribution of weather and other variables over some properties in winter cereal
Prob.	To predict the phenological development, the matter growth and the course of soil moisture in winter cereal stands (wheat, rye and barley)
Input	Meteorological data, crop observations and measures
Alg.	Backpropagation
Res.	Neural networks successfully predicted phenological development
Ref.	(Schultz and Wieland, 1997)
	Estimating the concentrations of pollutants in grass plant specie
Prob.	To estimate the deposition and accumulation of pollutants (lead) in a grass plant specie (*Cynodon dactylon*)
Input	The vegetation height, wind velocity, height of building, distance of adjacent street, traffic volume

Table 6. (*continued*)

Alg.	Backpropagation, multiple linear regression and stepwise multiple linear regression
Res.	The distance of adjacent streets, density, and height of buildings were the most important variables influencing the lead concentration. The artificial neural network approach was more accurate than regression models
Ref.	(Dimopoulos et al., 1999)

3.6 Management of Natural Resources

When studying the management of natural resources, it is important to know the different interactions that take place between the organisms, their environment and the changes occurred in the ecosystem. For instance, it is particularly useful to assess the changes of the organisms' distribution, ordination, and temporal change abundance or composition, to enable us to plan strategies, which take into account a more rational use of natural resources when dealing with agricultural systems. Therefore, there is a necessity to develop approaches that support the decision-making process in natural resources management. In the following table, there is a summary of some works carried out in order to model organisms' changes in ecosystems. It is also shown a case of soybean variety identification in the area of the preservation of genetic resources.

Table 7. Applications of artificial neural networks in management of natural resources

Prob.	To classify and ordinate a community of vegetation
Input	10 types of vegetation expressed as presence or absence
Alg.	Self-Organizing Map (SOM)
Res.	SOM was a feasible tool in classifying (grouping similar samples) ordering (arranging samples in an ordered manner) in ecology
Ref.	(Foody, 1999)
Prob.	To predict grassland vegetation community changes
Input	Grassland composition changes, and precipitation, evaporation, temperature, and wind velocity
Alg.	Backpropagation
Res.	
Ref.	(Tan and Smeins, 1996)
Prob.	To predict the apparition of a honeysuckle (*Lonicera morrowi*)
Input	Physical site characteristics, and soils data
Alg.	Backpropagation
Res.	The predictions of the neural networks were closely with field observations
Ref.	(Deadman and Gimblett, 1997)
Prob.	To predict richness and species composition in a tropical forest
Input	Satellite remote sensing data
Alg.	Backpropagation, radial basis function (RBF), generalized regression neural networks, Kohonen Self-Organizing Maps

Table 7. (*continued*)

Res.	It was possible to estimate both richness and species composition by using remotely sensed data
Ref.	(Foody and Cutler, 2006)
Prob.	To evaluate intra- and inter-specific variations using leaflet of soybean (*Glycine max*)
Input	Leaf shape images from 38 varieties
Alg.	Hopfield model, simple Perceptron
Res.	The model was a suitable tool in the varietal discriminitation of soybean leaflets
Ref.	(Oide and Ninomiya, 2000)

3.7 Irrigation and Fertilization

Water and plant nutrients are very relevant for crop management, because they are limiting factors in plant growth. While irrigation supplies the water required by crops according to their hydric requirements, when any other water source is limited, fertilization provides plants the necessary elements for their healthy growth. According to Raju et al. (2006) the water resources are becoming scarce due to factors resulting from human activities that have reduced water availability for irrigation. This problem is more evident in developing countries, where the population is growing faster, and where there is more contamination of water resources. In pursuit of a solution to this problem, a more accurate irrigation plan is needed in order to optimize the use of water. According to Broner and Comstock (1997) traditional knowledge-based crop management expert systems, include fertilization and irrigation and knowledge from field experts and growers. This knowledge is commonly acquired from different regions, which may differ in climate, soils and culture. There is a necessity to

Table 8. Applications of artificial neural networks in irrigation and fertilization

	Fertilization
Prob.	To provide nitrogen recommendations of fertilizer applications for malting barley
Input	Data of phosphorous and nitrogen recommendations generated by an expert system based on an interactive program
Alg.	Backpropagation
Res.	Analysis of the influence of data preprocessing
Ref.	(Broner and Comstock, 1997)
	Irrigation
Prob.	To select among available alternatives for irrigation planning
Input	Labour employment, agricultural production, economic benefits
Alg.	Self-Organizing Map (SOM)
Res.	The SOM-based integrative approach, was a successful tool for modeling a multiobjective irrigation planning
Ref.	(Raju et al., 2006)

fine-tune these systems. Artificial neural networks are able to tackle this problem by building models by exploiting training sets containing site-specific examples. The following table summarizes two studies dealing with irrigation and fertilization modeling in agricultural systems.

3.8 Physiology

Plant physiology is the study of the function, or physiology, of plants. It comprises the study of processes such as plant water relations and plant responses to different stimuli. On the one hand, evapotranspiration and transpiration constitute a part of

Table 9. Applications of artificial neural networks in physiology

	Evaporation
Prob.	To estimate evapotranspiration
Input	Data of meteorological variables and physical basis involved in the evapotranspiration process and the estimates provided by empirical models
Alg.	Quickprop, empirical models (Hargreaves-Samani and Blaney-Criddle)
Res.	Neural networks showed their potential for modeling evapotranspiration
Ref.	(Arca et al., 2001)
	Emulating plant growth
Prob.	To model plant growth by means of characterizing plant processes. Neural networks used to model transpiration process
Input	Air, canopy temperature, relative humidity, and plant type
Alg.	Backpropagation
Res.	The neural network model successfully modeled transpiration. But, it was necessary to identify certain plant physiology processes such as, assimilation, allocation, and nutrient update for modeling plant growth
Ref.	(Zee and Bubenheim, 1997)
Prob.	To analyze the lettuces (*Lactuca sativa* L.) growth characteristics under reduced gravity
Input	The biomass, chlorophyll content, plant width, and height
Alg.	Backpropagation
Res.	The neural networks were a feasible technique for modeling plant growth under reduced gravity
Ref.	(Zaidi et al., 1999)
Prob.	To estimate the biomass growth in winter cereals
Input	Site information, real observations and measurements, and weather information
Alg.	Backpropagation
Res.	The best network generating the desired biomass estimations used as inputs information of: temperature since sowing, field capacity of site, sum of precipitation since sowing, sum of global radiation since sowing and soil moisture in soil layer
Ref.	(Schultz et al., 2000)

water relation process. Evapotranspiration include: evaporation of water from soil and transpiration (evaporation of water by plants trough stomata and its leaves). This process is an important component of the hydrologic cycle and decision makers in agriculture shall be able to estimate a more accurately irrigation water requirements with information of evapotranspiration and other water balance factors. On the other hand, plant growth is a process that can be influenced by many physiological factors, as well as by many environmental stimuli. By modeling plant growth, decision makers should be able to fully profit from plant growth. The following table presents a series of works using artificial neural networks to estimate evaporation and transpiration, as well as three studies of neural networks emulating plant growth.

3.9 Greenhouse

Growing in greenhouses is an agronomical practice where plants are grown under more controlled conditions than conventional agriculture. This practice has some advantages such as an increase in crop yield, "indoor climate" factors can be controlled, pests and diseases are drastically reduced, plants are healthier, and it is also an ecological choice because herbicides are not required given that there are no weeds, and because it uses less water than growing plant outdoors. To generate the most suitable environment for plants growth is a challenge for growing greenhouses. To address this problem it has thus become necessary to use models able to simulate and predict greenhouse environment. These models should be able to identify the indoor conditions to modify, resulting in the most suitable environment for plant growth. In this manner, decision makers would improve their crop yields. The following table

Table 10. Applications of artificial neural networks in greenhouse control

Prob.	To study the compensation of external disturbances on the basis of input-output linearization and decoupling, in the operation of ventilation and moisturizing of greenhouses
Input	Combination of biological and physical models
Alg.	Feedback-feedforward linearization
Res.	The artificial neural network approach achieves input-output linearization and decoupling in the moisturizing and cooling of greenhouses
Ref.	(Pasgianos et al., 2003)
Prob.	To model greenhouse climate control
Input	Outside weather conditions, physical constants and control fluxes
Alg.	Bottleneck neural network in input reduction
Res.	Artificial neural networks were useful to control greenhouses
Ref.	(Seginer, 1997)
Prob.	To optimize the cultivation and the storage production process in tomato
Input	Cultivation process: the nutrient concentration of the solution. Storage optimization process: storage temperature
Alg.	Backpropagation, genetic algorithm
Res.	The expert system inferred practically the same strategy of cultivation and storage provided by a skilled grower
Ref.	(Morimoto and Hashimoto, 2000)

Table 10. (*continued*)

Prob.	To simulate and predict the inside greenhouse environment
Input	Inside air temperature, humidity and carbon dioxide concentration
Alg.	physical models and black-box linear parametric models
Res.	The neural network was not a feasible technique to predict the inside climate. The physical models had the best performance
Ref.	(Boaventura, 2003)
Prob.	To identify lettuce (*Lactuca sativa* L.) growth and greenhouse temperature
Input	Values of daily averaged CO2 concentration
Alg.	NUFZY (A hybrid neurofuzzy approach), OLS (orthogonal least squares)
Res.	The model proposed NUFZY coupled with the fast orthogonal least squares training algorithm had good performance in predicting both lettuce growth and greenhouse temperature
Ref.	(Tien and van Straten, 1998)

summarizes some examples of the prediction and simulation of greenhouse environment using artificial neural networks.

3.10 Soils

Soils provide plants with the support and the essential elements for growing. In soils many historical interactions and many processes take place, which make them more appropriate for a given use (agriculture, cattle raising, etc). The chemical and physical characteristics of soils are the result of these processes and interactions, and depending on these characteristics, a given soil could be suitable for the healthy growth of plants or

Table 11. Applications of artificial neural networks in soils

	Classification of physical properties
Prob.	To classify soil texture
Input	Combinations of different classifications of soil particles according to its size, and other soil parameters
Alg.	Backpropagation
Res.	The neural networks using as inputs soil properties such as: silt, clay, and organic carbon, had a soil structure classification an accuracy of around 79% in the training and validation phases
Ref.	(Levine et al., 1996)
Prob.	To classify soil texture
Input	Data from satellite; aerial remote sensing and soil structure
Alg.	Backpropagation
Res.	The spectral radiance and contextual information of the data sources were the most relevant variables in soil texture classification
Ref.	(Zhai et al., 2006)

Table 11. (*continued*)

Classification of land cover	
Prob.	To outperform a traditional land cover classification, known as National LandCover Data (NLCD)
Input	The visible bands (blue, green, and red) of the satellite image, textural information (set of texture measures based on the brightness value spatial dependency gray-level co-occurrence matrices)
Alg.	Backpropagation, decision-trees
Res.	The overall accuracy of the proposed method had better classification results than the National Land Cover Data classification
Ref.	(Arellano, 2004)

not. The understanding of phenomena associated to soil properties is particularly useful in making decisions about the adequate management of environmental resources and the improvement of productivity. Models simulating soil processes will help the understanding of important processes and will help to clarify problems related to agricultural activities. Diverse studies using neural networks have been done studying soils. The following table provides some studies related to model soil processes such as: rainfall-runoff, soil temperatures, soil water retention, pesticide concentrations, to predict chemical properties and to classify soil physical conditions as well.

4 Conclusions

Artificial neural networks have been successfully employed in the solution of diverse modeling problems of either regression or classification type in many agricultural domains. In most of the modeling applications presented in this survey, artificial neural network models showed better performance than traditional approaches. However, in several case studies, it is the hybrid approaches that present more accurate predictions, and only in one particular case, the traditional methods obtained better results. As far as the data is concerned, it has been shown that many sources of information can be used to train artificial neural networks in order to build an agroecological model. For instance, satellite images, pictures, climate and soil data, morphological descriptions, landscape characteristics, land management information, etc. are some examples of data sources exploited in the articles we have surveyed. More important is the ability of neural network models to incorporate data of diverse nature and from multiple sources.

The most frequently used modeling approach in our survey is based on simple feed-forward multi-layer Perceptrons, trained with the Backpropagation algorithm or with a variant of such an algorithm.

Given that the resulting models are based on a relatively high number of parameters (the interconnection weights between the neurons), they have a major drawback: knowledge extraction is quite difficult. Thus, we often refer to these models as “black boxes”. These models can perform quite well either on classification or regression tasks, but we do not know very well how they are using the input data to come-up with the outputs. Therefore, we cannot easily identify the most relevant input variables of a model or even if there are irrelevant input variables that have been used in the modeling process. Thus, a lot of research work has been devoted to this problem and several techniques have been proposed, attempting to alleviate this drawback and providing certain knowledge extraction capabilities from such models.

Another widely used neural network approach is the Self-Organizing Map, which is an unsupervised learning tool that has been mainly used as a clustering technique, but that can also be used as a powerful visualization tool, thanks to its capabilities of dimensionality reduction and topology preservation. Last but not least, it can be used as a very powerful tool for data exploration, by computing their so-called component planes.

References

Aitkenhead, M.J., et al.: Weed and crop discrimination using image analysis and artificial intelligence methods. Computers and Electronics in Agriculture 39, 15–171 (2003)

Arca, B., Benincasa, F., Vincenzi, M.: Evaluation of neural network techniques for estimating evapotranspira. In: Engineering Application of Neural Networks Conference, Cagliari, pp. 62–69 (2001)

Arellano, O.: An Improved Methodology for Land-Cover Classification Using Artificial Neural Networks and a Decision Tree Classifier. Ph.D. thesis, University of Cincinnati (2004)

Barreto, M., Pérez-Uribe, A.: Improving the correlation hunting in a large quantity of SOM component planes. In: de Sá, J.M., et al. (eds.) ICANN 2007. LNCS, vol. 4668, Springer, Heidelberg (2007)

Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)

Boaventura, J.: Greenhouse Climate Models: An Overview. In: EFITA 2003, Debrecen, Hungary (2003)

Bocco, M., Ovando, G., Sayago, S.: Development and evaluation of neural network models to estimate daily solar radiation at Córdoba, Argentina. Pesquisa Agropecuária Brasileira 41, 179–184 (2006)

Boishebert, d., Giraudel, J.L., Montury, M.: Characterization of strawberry varieties by SPME–GC–MS and Kohonen self-organizing map. Chemometrics and Intelligent Laboratory Systems 80, 13–23 (2006)

Broner, I., Comstock, C.R.: Combining expert systems and neural networks for learning site-specific conditions. Computers and Electronics in Agriculture 19, 37–53 (1997)

Burks, T.F., et al.: Evaluation of Neural-network Classifiers for Weed Species Discrimination. Biosystems Eng. 91, 293–304 (2005)

Chon, T.S., et al.: Patternizing communities by using an artificial neural network. Ecological Modelling 90, 69–78 (1996)

Chung, L.H., Hsieh, J.H., Chang, T.S.: Prediction of daily maximum ozone concentrations from meteorological conditions using a two-stage neural network. Atmospheric Research 81, 124–139 (2006)

Deadman, P., Gimblett, H.R.: An Application of Neural Net Based Techniques and GIS for Vegetation Management and Restoration. AI Applications (1997)

Diamantopoulou, M.J.: Artificial neural networks as an alternative tool in pine bark volume estimation. Computers and Electronics in Agriculture 48, 235–244 (2005)

Dimopoulos, I., et al.: Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in Athens city. Ecological Modelling 120, 157–165 (1999)

Drummond, S.T., Sudduth, K.A., Joshi, A.: Predictive Ability Of Neural Networks For Site-Specific Yield Estimation. The Second International Geospatial Information in Agriculture and Forestry, Lake Buena Vista, Florida (2000)

Foody, G.M.: Applications of the self-organising feature map neural network in community data analysis. Ecological Modelling 97–107 (1999)

Foody, G.M., Cutler, M.E.J.: Mapping the species richness and composition of tropical forests from remotely sensed data with neural networks. Ecological Modelling 195, 37–42 (2006)

Francl, L.J.: Squeezing the turnip with artificial neural nets. Phytopathology 94, 1007–1012 (2004)

Giraudel, J.L., Lek, S.: A comparison of self-organizing map algorithm and some conventional statistical methods for ecological community ordination. Ecological Modelling 146, 329–339 (2001)

Granitto, P.M., et al.: Automatic Identification Of Weed Seeds By Color Image Processing. VI Argentine Congress on Computer Science Ushuaia, Argentina (2000)

Green, T.R., et al.: Relating crop yield to topographic attributes using Spatial Analysis Neural Networks and regression. Geoderma (Article in press) (2007)

Gupta, R., et al. (eds.): Understanding Helicoverpa armigera Pest Population Dynamics related to Chickpea Crop Using Neural Networks Third International Conference on Data Mining, Florida, USA. IEEE Computer Society Press, Los Alamitos (2003)

Guyer, D.E., Yang, X.: Use of genetic artificial neural networks and spectral imaging for defect detection on cherries. Computers and Electronics in Agriculture 29, 179–194 (2000)

Hashimoto, Y.: Special issue:Applications of artificial neural networks and genetic algorithms to agricultural systems. Computers and Electronics in Agriculture 18, 71–72 (1997)

Hilbert, D.W., Ostendorf, B.: The utility of artificial neural networks for modelling the distribution of vegetation in past, present and future climates. Ecological Modelling 146, 311–327 (2001)

Himberg, J.: Enhancing the SOM-based Data Visualization by Linking Different Data Projections. In: Proceedings of 1st International Symposium IDEAL 1998, Intelligent Data Engineering and Learning–Perspectives on Financial Engineering and Data Mining, pp. 427–434 (1998)

Hoogenboom, G., Georgiev, G., Gresham, D.: Development of weather based products for agricultural and environmental applications. In: Preprints of the 24th Conf. On Agricultural and Forest Meteorology, pp. 66–67. American Meteorological Society, Boston, Mass (2000)

Huang, K.-Y.: Application of artificial neural network for detecting Phalaenopsis seedling diseases using color and texture features. Computers and Electronics in Agriculture 57, 3–11 (2007)

Jain, A.: Predicting Air Temperature For Frost Warning Using Artificial Neural Networks. Msc thesis. The University of Georgia (2003)

Jiménez, D.R., Satizabal, H.F., Pérez-Uribe, A.: Modelling Sugar Cane Yield Using Artificial Neural Networks The 6th European Conference on Ecological Modelling, Trieste, Italy, November 27-30 (to appear, 2007)

Kaul, M., Hill, R.L., Walthall, C.: Artificial neural networks for corn and soybean yield prediction. Agricultural Systems 85, 1–18 (2005)

Kavdır, I.: Discrimination of sunflower, weed and soil by artificial neural networks. Computers and Electronics in Agriculture 44, 153–160 (2004)

Kehagias, A., et al.: Predictive Modular Neural Networks Methods for Prediction of Sugar Beet Crop Yield. In: IFAC Conference on Control Applications and Ergonomics in Agriculture, Athens, Greece (1998)

Koekkoek, E.J.W., Booltink, H.: Neural network models to predict soil water retention. European Journal of Soil Science 50, 489–495 (1999)

Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1997)

Kondo, N., et al.: Machine vision based quality evaluation of Iyokan orange fruit using neural networks. Computers and Electronics in Agriculture 29, 135–147 (2000)

Levine, E.R., Kimes, D.S., Sigillito, V.G.: Classifying soil structure using neural networks. Ecological Modelling 92, 101–108 (1996)

Li.: Spatial Interpolation Of Weather Variables Using Artificial Neural Networks. Msc thesis, University of Georgia, Athens (2002)

Liu, Y., Weisberg, H., He, R.: Sea surface temperature patterns on the West Florida Shelf using Growing Hierarchical Self-Organizing Maps. Journal of Atmospheric and Oceanic Technology 23, 325–338 (2006)

Miao, Y., Mulla, D.J., Robert, P.C.: Identifying important factors influencing corn yield and grain quality variability using artificial neural networks. Precision Agriculture 7, 117–135 (2006)

Morimoto, T., Hashimoto, Y.: AI approaches to identification and control of total plant production systems. Control Engineering Practice 8, 555–567 (2000)

Moshou, D., et al.: Plant disease detection based on data fusion of hyper-spectral and multi-spectral fluorescence imaging using Kohonen maps. Real-Time Imaging 11, 75–83 (2005)

Moshou, D., et al.: Automatic detection of 'yellow rust' in wheat using reflectance measurements and neural networks. Computers and Electronics in Agriculture 44, 173–188 (2004)

Moshou, D., Ramon, H., Baerdemaeker, d.: A Weed Species Spectral Detector Based on Neural Networks. Precision Agriculture 3, 209–223 (2002)

Moshou, D., et al.: A neural network–based plant classifier. Computers and Electronics in Agriculture 31, 5–16 (2001)

Murase, H.: Special issue:artificial intelligence in agriculture. Computers and Electronics in Agriculture 29, 1–2 (2000)

Nakano, K.: Application of neural networks to the color grading of apples. Computers and Electronics in Agriculture 18, 105–116 (1997)

Noble, P.A., Tribou, E.H.: Neuroet: An easy-to-use artificial neural network for ecological and biological modeling. Ecological Modelling 203, 87–98 (2007)

Oide, M., Ninomiya, S.: Discrimination of soybean leaflet shape by neural networks with image input. Computers and Electronics in Agriculture 29, 59–72 (2000)

Park, S.J., Hwang, C.S., Vlek, P.L.G.: Comparison of adaptive techniques to predict crop yield response under varying soil and land management conditions. Agricultural Systems 85, 59–81 (2005)

Paruelo, J.M., Tomasel, F.: Prediction of functional characteristics of ecosystems: a comparison of artificial neural networks and regression models. Ecological Modelling 98, 173–186 (1997)

Pasgianos, G.D., et al.: A nonlinear feedback technique for greenhouse environmental control. Computers and Electronics in Agriculture 40, 153–177 (2003)

Paul, P.A., Munkvold, G.P.: Regression and Artificial Neural Network Modeling for the Prediction of Gray Leaf Spot of Maize. Phytopathology 95, 388–396 (2005)

Philip, N.S., Joseph, K.B.: A neural network tool for analyzing trends in rainfall. Computers & Geosciences 29, 215–223 (2003)

Raju, K.S., Kumar, D.N., Ducksteinc, L.: Artificial neural networks and multicriterion analysis for sustainable irrigation planning. Computers & Operations Research 33, 1138–1153 (2006)

Satizábal, H.F., Jiménez, D.R., Pérez-Uribe, A.: Consequences of Data Uncertainty and Data Precision in Artificial Neural Network Sugar Cane Yield Prediction. In: Sandoval, F., et al. (eds.) IWANN 2007. LNCS, vol. 4507, Springer, Heidelberg (2007)

Satizábal, H.F., Pérez-Uribe, A.: Relevance Metrics to Reduce Input Dimensions. ICANN 2007 International Conference on Artificial Neural Networks, Porto, Portugal, September 9 – 13 (to appear, 2007)

Samad, T., Harp, S.A.: Self-organization with partial data. Network: Computation in Neural Systems 3, 205–212 (1992)

Schultz, A., Wieland, R.: The use of neural networks in agroecological modelling. Computers and Electronics in Agriculture 18, 73–90 (1997)

Schultz, A., Wieland, R., Lutze, G.: Neural networks in agroecological modelling- stylish application or helpful tool? Computers and Electronics in Agriculture 29, 73–97 (2000)

Seginer, I.: Some artificial neural network applications to greenhouse environmental control. Computers and Electronics in Agriculture 18, 167–186 (1997)

Shamseldin, A.Y.: Application of a neural network technique to rainfall-runoff modelling. Journal of Hydrology 199, 272–294 (1997)

Shearer, J.R., et al.: Yield Prediction Using A Neural Network Classifier Trained Using Soil Landscape Features and Soil Fertility Data Annual International Meeting, Midwest Express Center. ASAE Paper No. 001084, Milwaukee, Wisconsin (2000)

Tien, B.T., van Straten, G.: A NeuroFuzzy Approach to Identify Lettuce Growth and Greenhouse Climate. Artificial Intelligence Review 12, 71–93 (1998)

Timm, L.C., et al.: Neural network and state-space models for studying relationships among soil properties. Scientia Agricola (Piracicaba, Braz.) 63, 386–395 (2006)

Tourenq, C., et al.: Use of artificial neural networks for predicting rice crop damage by greater flamingos in the Camargue, France. Ecological Modelling 120, 349–358 (1999)

Uno, Y., et al.: Artificial neural networks to predict corn yield from Compact Airborne Spectrographic Imager data. Computers and Electronics in Agriculture 47, 149–161 (2005)

Veronez, M.R., et al.: Artificial Neural Networks applied in the determination of Soil Surface Temperature – SST. In: 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences, Lisboa (2006)

Vesanto, J.: SOM-based data visualization methods. Intelligent Data Analysis 3, 111–126 (1999)

Vesanto, J., Alhoniemi, E.: Clustering of the Self-Organizing Map. IEEE Transactions on neural networks 11, 568–600 (2000)

Yang, C.C., et al.: Development of a herbicide application map using artificial neural networks and fuzzy logic. Agricultural Systems 76, 561–574 (2003)

Yang, C.C., et al.: Application Of Artificial Neural Networks For Simulation Of Soil Temperature. Trans. of the ASAE 40, 649–656 (1997a)

Yang, C.-C., et al.: An artificial neural network model for simulating pesticide concentrations in soil. Transactions of the ASAE 40, 1285–1294 (1997b)

Zaidi, M.A., Murase, H., Honami, N.: Neural Network Model for the Evaluation of Lettuce Plant Growth. Journal of Agricultural Engineering Research 74, 237–242 (1999)

Zee, F., Bubenheim, D.: Plant Growth Model Using Artificial Neural Networks (1997)

Zhai, Y., et al.: Soil texture classification with artificial neural networks operating on remote sensing data. Computers and Electronics in Agriculture 54, 53–68 (2006)

Software Development Knowledge Management Using Case-Based Reasoning

Paulo Gomes, Joel Cordeiro, Pedro Gandola, and Nuno Seco

CISUC, University of Coimbra, Portugal

1 Introduction

The knowledge generated in the software development process is a precious resource for companies. Usually this knowledge is not stored, which does not enable its reuse in future projects. Knowledge reuse in the domain of software development has several advantages: it improves productivity and quality of software systems (Coulange 1997, Gamma et al. 1995, Jacobson et al. 1997, Meyer 1987, Prieto-Diaz 1993); it minimizes the loss of know-how when a member of the development team abandons the company; and it enables knowledge sharing across different development teams or projects.

Usually software development comprises several phases (Boehm 1988): analysis, design, implementation, testing and integration. This work focuses on the design phase, which defines the structure and behavior of the system being specified. Design is a complex an ill-defined task (Tong et al. 1992), which comprises synthesis and analysis subtasks, making the modeling and automation of the design processes difficult. A software engineer often reuses knowledge from previous projects during the design of new projects. In this work, we are interested in studying the management of design knowledge in a software development company, involving several software designers.

Most of the decisions concerning software design are made using the designers' experience. The more experience a designer has, the better s/he can perform her/his job. Reasoning based on experience is a basic mechanism for designers, enabling them to reuse previous design solutions in well known problems or even in new projects. In artificial intelligence there is a sub area called Case-Based Reasoning (CBR, see (Aamodt et al. 1994, Kolodner 1993)) that uses experiences, in the form of cases, to perform reasoning. CBR can be viewed as a methodology for developing knowledge-based systems that uses experience for reasoning about problems (Althoff 2001). We think that CBR is a well suited methodology for building a design system that can act like an intelligent design assistant.

We developed a computational system – REBUILDER UML – that perform three tasks: store, manage and reuse of software design knowledge. To achieve these goals, we propose a system based on CBR. This reasoning framework is flexible enough to comply with different knowledge types and reasoning mechanisms, enabling the software designer to use whatever design assistant s/he wants to use. We also integrated an ontology, which enables several semantic operations like indexing software objects and computing semantic distances between software objects. In

REBUILDER UML there is a module designated – REBUILDER TextToDiagram – that translates natural language text into class diagrams. This module has as input project requirements or specifications and transforms it, into an UML class diagram. This module is also described in this chapter with some detail.

The next section describes the CBR methodology. We then present the predecessor of our system: REBUILDER I. The following section describes our approach and the architecture of our system, after which the following sections describe the main parts: the knowledge base structure and content; the knowledge base manager used to control and modify the knowledge base; the CBR engine; and the TextToDiagram module. The next section describes an example of use, providing more information about its functioning, followed by a section that describes experimental work. To conclude this chapter, we present other software reuse systems and make some final remarks and future directions.

2 ReBuilder I

REBUILDER I (Gomes 2004, Gomes et al. 2004, Gomes et al. 2002, Seco et al. 2004) was developed with two main goals: create a corporative memory of software designs, and provide the software designer with a design environment capable of promoting software design reuse. This is achieved in our approach using CBR as the main reasoning process, and with cases as the main knowledge building blocks. REBUILDER comprises four different modules: Knowledge Base (KB), UML Editor, Knowledge Base Manager and CBR engine (see Fig. 1).

The UML editor is the front-end of REBUILDER I and the environment where the software designer works. Apart from the usual editor commands to manipulate UML objects, the editor integrates new commands capable of reusing design knowledge. These commands are directly related with the CBR engine capabilities and are divided into two main categories:

- **Knowledge Base actions:** such as connect to KB and disconnect from KB.
- **Cognitive actions:** such as retrieve design, adapt design using analogy or design composition, verify design, evaluate design, and actions related with object classification using WordNet (Miller et al. 1990).

The Knowledge Base Manager module is used by the administrator to manage the KB, keeping it consistent and updated. This module comprises all the functionalities

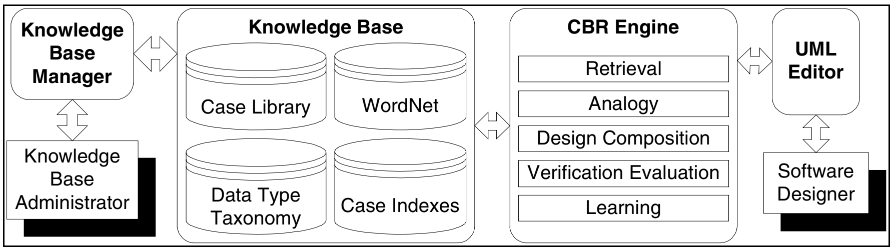


Fig. 1. REBUILDER's architecture

of the UML editor, and it adds case base management functions to REBUILDER I. These are used by the knowledge base administrator to update and modify the KB. The list of available functions is:

- **KB Operations:** Creates, opens or closes a KB.
- **Case Library Manager:** Opens the Case Library Manager, which comprises functions to manipulate the cases in the case library, like adding new cases, removing cases, or changing the status of a case.
- **Activate Learning:** Gives the knowledge base administrator an analysis about the contents of the case library. REBUILDER I uses several case base maintenance techniques to determine which cases should be added or removed from the case library.
- **Settings:** Adds extra configuration settings which are not present in the normal UML Editor version used by software designers. It also enables the knowledge base administrator to fine tune the reasoning mechanisms.

The KB comprises four different parts: case library, which stores the cases of previous software projects; an index memory used for efficient case retrieval; data type taxonomy, which is an ontology of the data types used by the system; and WordNet, which is a general purpose ontology.

The CBR Engine is the reasoning module of REBUILDER I. As the name indicates, it uses the CBR paradigm to establish a reasoning framework. This module comprises five different parts: Retrieval, Design Composition, Analogy, Verification, and Learning.

REBUILDER UML addresses the limitations of REBUILDER I, especially the ontology performance. REBUILDER I has two main limitations due to the use of WordNet. Performance issues, for instance, finding the semantic distance between two concepts can take several seconds, which is unacceptable for the system usage. The second limitation is the shallowness of WordNet that is not adequate for specific domains like computing and software engineering. Specific concepts used in most of the software systems being modeled, do not exist in WordNet. Another aspect of REBUILDER I that was modified in REBUILDER UML is the application philosophy, instead of having a client server architecture, REBUILDER UML is a plug in for a CASE tool, which can easily be used within a development team, or by a single software engineer. This change has made the system more flexible regarding usability.

3 ReBuilder UML

REBUILDER UML is implemented as a plug in for Enterprise Architect (EA www.sparxsystems.com.au), a commercial CASE tool for UML modeling, and it comprises three main modules (see Fig. 2): the knowledge base (KB), the CBR engine, and the KB manager. The KB is the repository of knowledge that is going to be reused by the CBR engine. The main goal of the system is to reuse UML class diagrams, which are stored as cases in the case library and reused by the CBR engine. The knowledge base manager enables all the knowledge stored in the system to be maintained.

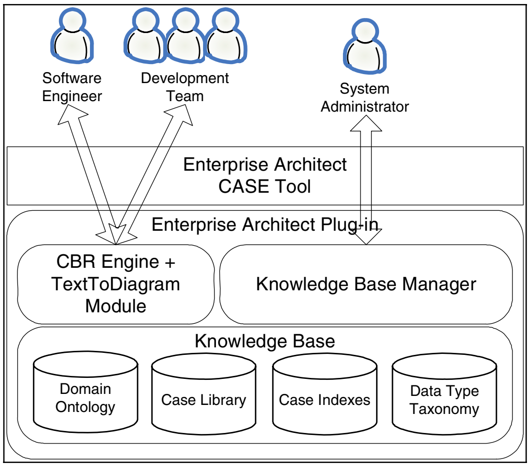


Fig. 2. The architecture of REBUILDER UML, based as a plug-in for Enterprise Architect

There are two types of users in REBUILDER UML, software engineers and the system administrator. A software engineer uses the CASE tool to model a software system in development, and s/he can use REBUILDER UML actions to reuse old diagrams. These diagrams can come from previous systems, or from the development team in which the software engineer is integrated. The other user type is the system administrator, which has the aim of keeping the KB fine tuned and updated. Since each software engineer has a copy of the central KB, the system administrator is responsible for making new releases of the KB and installing it in the systems of the development team (or teams). Thus, the role of the administrator is very important for REBUILDER UML to be used properly by several users, enabling the sharing of knowledge among them. Despite this, the system can also be used in a stand alone fashion, acting as an intelligent knowledge repository for a single user. In this setup, the user is at the same time playing both roles, reusing knowledge and maintaining it.

The integration with EA is made by a plug in, enabling REBUILDER UML to have access to the data model of EA, and also to its model repository. Visually the user interacts with REBUILDER UML through the main menu of EA. The user has access to the specific commands of REBUILDER UML, enabling search, browse, retrieval, reuse and maintenance operations. The next sections describe the main parts of REBUILDER UML.

4 Knowledge Base

The KB comprises four different parts: the domain ontology, which represents the concepts, relations between concepts, attributes and axioms of the domain being modeled; the case library that stores all the UML class diagrams, called cases; the case indexes, which are associations between class diagram objects and ontology concepts; and the data type taxonomy, which is a simple taxonomy of programming data types used for semantic comparison.

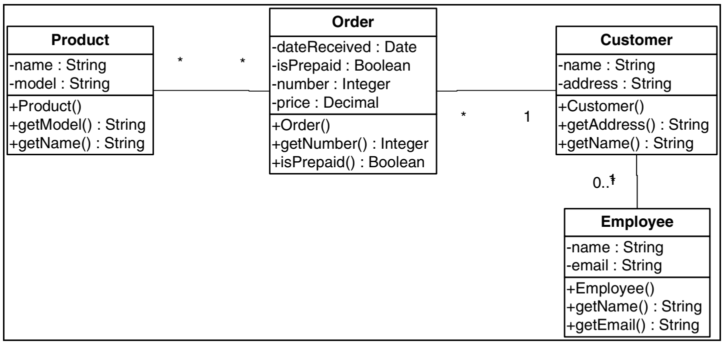


Fig. 3. An example of an UML class diagram

A case in REBUILDER UML represents a specific UML class diagram (see Fig. 3 for an example of a class diagram). Conceptually a case comprises: a name used to identify the case within the case library; the main package, which is an object that comprises all the objects that describe the main class diagram; and the file name where the case is stored. Cases are stored using XML/XMI since it is a widely used format for data exchange.

UML class diagram objects considered are: packages, classes, interfaces and relations. A package is an UML object used to group other objects, and it is defined by: a name, a concept in the ontology, and a list of other UML objects. A class describes an entity and it corresponds to a concept described by attributes at a structural level, and by methods at a behavioral level. A class is described by: a name, a concept in the ontology, an attribute list, and a method list. The interface describes a protocol of communication for a specific class. An interface can have one or more implementation, and is described by: a name, a concept in the ontology, and a list of methods. A relation describes a relationship between two UML objects, and it is characterized by several attributes, which are: a name, the source object, the destination object, the relation type (association, generalization, dependency, or realization), cardinality, and aggregation. An attribute refers to a class and is characterized by a name that identifies the attribute within the class it belongs; the attribute's scope in relation to the external objects: public, private, or protected; the attribute's data type; and the attribute's default value. A method describes a request or message that can be submitted to a class, and is described by: a name that identifies the method within the class to which it belongs; the method's scope in relation to the external objects: public, private, or protected; the list of the input parameters; and the list of output parameters. A parameter can be a reference or a value that is used or generated by a class method, and is described by: a name identifying the parameter within the method to which it belongs, and the parameter's data type.

The domain ontology defines concepts, which are represented by a set of words. Words that can be used to represent the same concept are called synonyms. A word associated with more than one concept is called a polysemous word. For instance, the word mouse has two meanings, it can denote a rat, or it can express a computer mouse. Besides the list of words, a concept has a list of semantic relations with other concepts in the ontology. These relations are categorized in four main types: *is-a*,

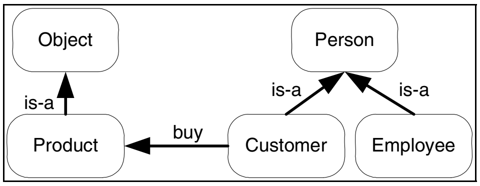


Fig. 4. An example of part of a domain ontology with concepts and relations

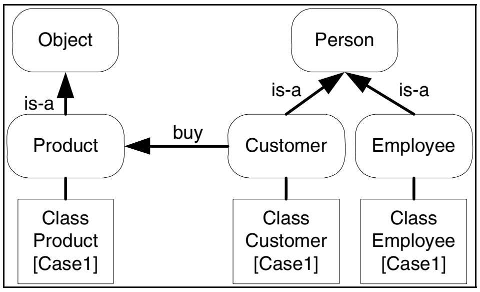


Fig. 5. An example of case indexing considering the diagram of Fig. 3 as *Case1*

part-of, *substance-of* and *member-of*, but the administrator can specify other types of relations. An example of part of an ontology is presented in Fig. 4.

The ontology is used for computing the semantic distance between two concepts. Another purpose of the ontology is to index cases, and for this task, REBUILDER UML associates a concept to each diagram object. This link is then used as an index to the ontology structure, which can be used as a semantic network for case or object retrieval. Considering the diagram of Fig. 3 as *Case1*, Fig. 5 represents part of the case indexing, with objects *Product*, *Customer* and *Employee* indexed in the ontology.

As cases can be large, they are stored in files, which make case access slower then if they were in main memory. To solve this problem we use the case indexes. These provide a way to access the relevant case parts for retrieval without having to read all the case files from disk. Each object in a case is used as an index. REBUILDER UML uses the concept associated to each object to index the case in the ontology. This way, the system can retrieve a complete case, using the case package, or it can retrieve only a subset of case objects, using the objects' indexes. This allows the user with the possibility to retrieve not only packages, but also classes and interfaces.

5 Knowledge Base Manager

REBUILDER UML stores and manages design knowledge gathered from the software designer's activity. This knowledge is stored in a central repository, which is managed by the administrator. The basic responsibilities of the administrator are to configure the system and to decide which cases should be in the case library. Another task that s/he has to perform is to revise new diagrams submitted by the software designers.

Deciding the contents of the case library is not an easy task, especially if the case base has a large number of cases. In this situation the KB manager provides a tool which enables the administrator to retrieve the most similar cases to the one being evaluated. If the new case is different enough, then it is added to the case library, otherwise the administrator stores the new case in a list of obsolete cases.

When a diagram is submitted by a software designer as a candidate to a new case to be added to the case library, the administrator has to check some items in the diagram. First the diagram must have concepts associated to the classes, interfaces and packages. This is essential for the diagram to be transformed into a case, and to be indexed and reused by the system. Diagram consistency and coherence must also be checked.

The KB Manager module is used by the administrator to manage the KB, keeping it consistent and updated. This module comprises all the functionalities of the UML editor, and it adds case base management functions to REBUILDER UML. These are used by the KB administrator to update and modify the KB. The available functions are:

- **KB Operations** – create, open or close a KB;
- **Case Library Manager** – opens the case library manager, which comprises functions to manipulate the cases in the case library, like adding new cases, removing cases, or changing the status of a case;
- **Ontology Manager** – provides the user with an editor to modify the ontology, enabling the creation and manipulation of concepts, which are used by the system to reason;
- **Settings** – adds extra configuration settings which are not present in the normal UML Editor version used by the software designers. It also enables the KB administrator to configure the reasoning mechanisms.

6 CBR Engine

The system provides two types of retrieval: object retrieval and case retrieval. The retrieval mechanism searches the ontology structure looking for similar objects/cases and then ranks the objects/cases found, presenting them to the designer.

Retrieval comprises two phases: retrieval of a set of relevant objects/cases from the case library, and assessment of the similarity between the target problem and the retrieved objects/cases. The retrieval phase is based on the ontology structure, which is used as an indexing structure. The retrieval algorithm uses the classifications of the target object as the initial search probe in the ontology.

The query for retrieval comprises the objects selected by the user. If there is more than one object, then the systems retrieves complete diagrams (cases). If only one object is selected at the moment of the retrieval command, then the system retrieves objects from the case library. The retrieval algorithm uses the concepts of the selected objects as the initial search probe. Then the algorithm checks if there are any object indexes associated with the ontology nodes of those concepts. If there are enough indexes, the algorithm stops and returns them. Otherwise, it explores the concept nodes adjacent to the initial ones, searching for object indexes until the number of found indexes reaches the number of objects that the user wants to be retrieved.

The second step of retrieval is ranking the retrieved objects/cases by similarity with the target object(s). We have defined two similarity metrics, according to two situations: retrieval of cases (case similarity metric) or retrieval of objects (object similarity metric).

- **Case Similarity Metric:** this metric is based on three different aspects: structural similarity between objects and relations in the query and the case; and once query objects are matched with case objects, it assesses the semantic similarity between matched objects. Basically this metric assesses structure similarity and semantic similarity of cases and its objects.
- **Object Similarity Metric:** the object similarity metric is based on three items: concept similarity of objects being compared, inter-object similarity comprising the assessment of relation similarity between objects, and intra-object similarity that evaluates the similarity between objects' attributes and methods.

After the ranked objects/cases are presented to the user, s/he can select three different actions: copy, replace or merge. The copy action copies the selected object/case to the current diagram, where it can then be reused. The replace operation replaces the query with the selected object/case. The merge action merges the selected object/case with the query. Even if the software engineer does not use any of the retrieved objects/cases, s/he can explore the design space using the retrieved knowledge. This enables a more efficient way of designing systems and increases productivity, enabling novice engineers to get a first solution, from which they can iteratively build a better solution.

7 TextToDiagram Module

REBUILDER TextToDiagram comprises a morphological analyzer that outputs tagged text, a syntactic analyzer that outputs syntactic chunks and a semantic analyzer that outputs UML class diagrams, as can be seen in Fig. 6. It uses the OpenNLP tool (Project 2005) to do the morphological and syntactic analysis. The knowledge layer comprises three parts: a domain ontology used to compute semantic distances between concepts; case indexes, which are links between cases and the ontology; and the case base, comprising all the cases needed for the system to reason. The ontology comprises concepts and relations between concepts. Cases comprise a text, a diagram and mappings between the text and the diagram objects and relations. A case index is an association between a word in a case and an ontology concept.

7.1 Morphological Analyzer

The Morphological analyzer receives a text where the user expresses system requirements. It has essentially three phases: splitting, tokenization and POS tagging. During the splitting phase, it identifies all the phrases of input text (see Fig. 7).

Then, the tokenizer breaks the identified phrases into tokens (see Fig. 8). Finally, the POS tagger identifies all the grammatical categories (using tags for

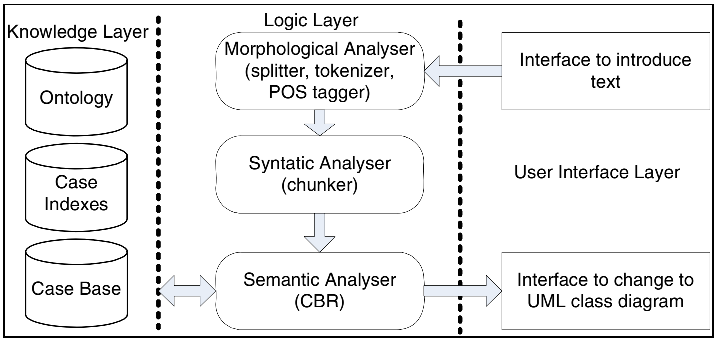

Knowledge Layer
Ontology
Case Indexes
Case Base
Logic Layer
Morphological Analyser (splitter, tokenizer, POS tagger)
Syntatic Analyser (chunker)
Semantic Analyser (CBR)
Interface to introduce text
User Interface Layer
Interface to change to UML class diagram

Fig. 6. REBUILDER TextToDiagram architecture

> Vendors may be sales employees or companies.
> Sales employees receive a basic wage and a commission, whereas companies only receive a commission.

Fig. 7. Splitting of the input text

> Vendors | may | be | sales | employees | or | companies | .
> Sales | employees | receive | a | basic | wage | and | a | commission | , | whereas | companies | only | receive | a | commission | .

Fig. 8. Text tokenization

> Vendors/NNS may/MD be/VB sales/JJ employees/NNS or/CC companies/NNS ./.
> Sales/NNS employees/NNS receive/VBP a/DT basic/JJ wage/NN and/CC a/DT commission/NN ,/, whereas/IN companies/NNS only/RB receive/VBP a/DT commission/NN ./.

Fig. 9. Text POS tagging

example: NN for noun, NNS for plural nouns, VB for verbs, and so on) of the tokens identified as can be seen in Fig. 9.

7.2 Syntactic Analyzer

The Syntactic analyzer receives all tags from the previous module and defines all the chunks. These chunks are syntactic units useful when looking for units of meaning

> [NP Vendors/NNS] [VP may/MD be/VB] [NP sales/JJ employees/NNS or/CC companies/NNS] ./.
> [NP Sales/NNS employees/NNS] [VP receive/VBP] [NP a/DT basic/JJ wage/NN] and/CC [NP a/DT commission/NN] ,/, [PP whereas/IN] [NP companies/NNS] [ADVP only/RB] [VP receive/VBP] [NP a/DT commission/NN] ./.

Fig. 10. Text Chunking

larger than the individual words, identified in the text using shallow parsing. Fig. 10 presents the obtained chunks. The chunks produced by the syntactic analyzer are then passed to the semantic analyzer to produce the corresponding class diagram.

7.3 Semantic Analyzer

The Semantic analyzer receives the chunks and produces the corresponding UML class diagram using Case Based Reasoning (CBR). Each case (see Fig. 11 for an example) has a problem description (chunks and a text ID), the solution description (the UML class diagram) and the mappings between the problem terms and the solution objects (from nouns to classes, verbs to relations or methods and so on). A case is stored in a XMI file (XMI stands for XML Metadata Interchange, which is a proposed use of XML intended to provide a standard way for programmers and other users to exchange information about metadata).

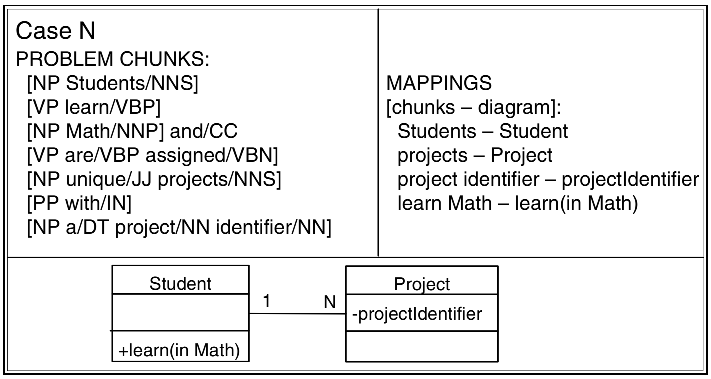


Fig. 11. Example of a case

There are two structures used to store cases indexes: an ontology and a syntactic tree. The ontology reflects the semantic similarity (and relatedness) using relations of specialization and generalization between concepts. There are indexes in the ontology that establish a relation between concepts and cases. The indexes are links between words in the case problem description and the corresponding ontology concept. This association is straightforward since the domain ontology is intended not to have ambiguities. There are also relations between chunks and cases in the syntactic tree.

The syntactic tree (see Fig. 12) reflects the syntactic similarity, storing cases in specific nodes of chunks. For instance the phrase "*A bank has many clients with several accounts*" would have the chunks [**NP** A/DT bank/NN] [**VP** has/VBZ] [**NP** many/JJ customers/NNS] [**PP** with/IN] [**NP** several/JJ accounts/NNS], so the most similar case with this one stored in the tree is the case 1.

The semantic analyzer architecture was developed according to the main steps of CBR, as we can see in the Fig. 13. The first step is case retrieval, which uses the index structures to select a group of similar cases. Then, from these cases, the system selects the best case using similarity metrics (**case selection**). With the best case selected, the system reuses it using heuristic rules (**case adaptation**) by mapping the

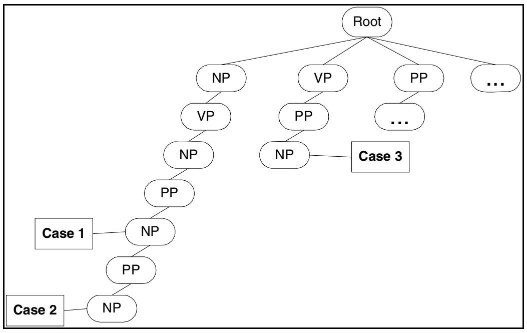


Fig. 12. Example of the syntactic tree

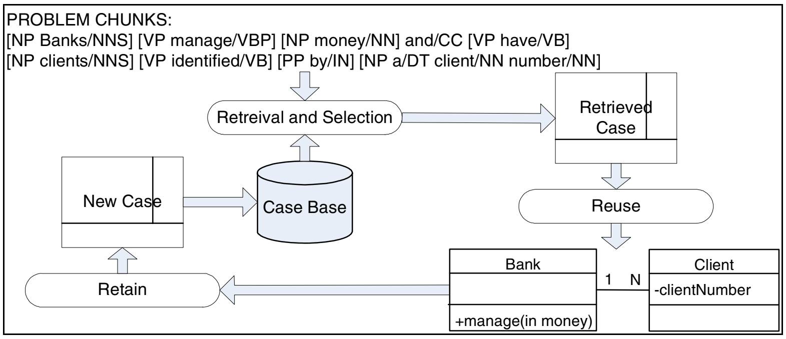

PROBLEM CHUNKS:
[NP Banks/NNS] [VP manage/VBP] [NP money/NN] and/CC [VP have/VB]
[NP clients/NNS] [VP identified/VB] [PP by/IN] [NP a/DT client/NN number/NN]
Retreival and Selection
Retrieved Case
New Case
Case Base
Reuse
Bank
+manage(in money)
1
N
Client
-clientNumber
Retain

Fig. 13. Semantic analyzer architecture

chunks of the solution with chunks of the problem. The last process stores the process obtained in the case base, if the similarity is beyond a specified threshold.

Case Retrieval

The retrieval process comprises four algorithms that use the index structures: Syntactic, Semantic Filter, Semantic and Conjunctive. All algorithms start by finding a list of entry points into the index structures. Verbs and names, which are associated with ontology concepts, are used as the entry points in the ontology. In the syntactic index tree, the entry point is the node with the same chunks as our target sentence. If it does not exist, the algorithm creates a list of entry points (above a specified threshold) containing nodes that represents the generalization and specialization of the target sentence. The core of the retrieval algorithms is presented in Fig. 14.

The **syntactic retrieval** algorithm uses only the syntactic index tree. The **semantic filter retrieval** algorithm uses the syntactic index tree and the ontology to constrain the search to a specific semantic distance threshold. The **semantic filter** algorithm uses the ontology to obtain concept references to names of the target sentence and uses these concepts as the entry points. The **conjunctive retrieval** algorithm uses both the syntactic index tree (as in the syntactic retrieval) and the ontology (as in the semantic filter) to retrieve indexes and in the end intersects the indices returned from each structure.

```
 indices syntacticRetrieval(syntacticNodes)
 for each element in syntacticNodes
frontier += initialFrontier
 end for
 while length(frontier) > 0
for each element in frontier
   indices += obtainSyntacticIndeces(element)
   auxFrontier += specializations(element) + generalizations(element)
end for
 end while
 return indices
```

Fig. 14. Core of the retrieval algorithm

Case Selection

The selection process outputs the most similar case using seven algorithms with different similarity metrics: syntactic, semantic, discourse, contextual, morphological, Levenshtein distance and n-gram similarity. We use the chunks, tags and tokens as comparison units in these algorithms.

The **syntactic similarity** metric verifies the number of common nodes between problem chunks and solution chunks. This metric is computed using the following formula (where *T* – target; *S* – Source, *I* – number of intersection nodes, *nodes* – number of nodes):

$$\frac{1}{2}\times\left(\frac{I(T,S)}{nodes(T)}+\frac{I(T,S)}{nodes(S)}\right) \tag{1}$$

The **semantic similarity** metric verifies the semantic distance between the verbs and nouns of the problem and solution with this formula (*MSL* – Maximum Search Length allowed by the algorithm, constraining the search space):

$$\frac{1}{2*MSL}\times\left(\frac{dist(T,S)}{nodes(T)}+\frac{I(T,S)}{nodes(S)}\right) \tag{2}$$

The **contextual similarity** metric verifies the number of times that specific cases from a specific text were used. If the case was already used the algorithm returns 1 else it returns 0.

The **discourse similarity** verifies the position of each sentence of both the problem and the solution in the original text:

$$\frac{1}{\text{sentenceNumber(T) - sentenceNumber(S)}^2+1} \tag{3}$$

The **morphological similarity** metric verifies the number of common tags between the phrase of the problem and the phrase of the solution, the formula is:

$$\frac{I(T,S)}{\max(nodes(T),nodes(S)} \tag{4}$$

The **Levenshtein distance** metric (Day 1984) is used between two phrases (chunks or tags) and is given by the minimum number of operations needed to transform one phrase into the other, where an operation is an insertion, deletion, or substitution of a single string.

The **n-gram similarity** metric (Kondrak 2005) is used to compare n-grams of two phrases (chunks, tags or tokens):

$$\frac{I(nGramsT, nGramsS)}{nGrams(T)} + \frac{I(nGramsT, nGramsS)}{nGrams(S)} \tag{5}$$

Each previous equation is normalized to the interval [0...1]. Then the system computes a weighted sum to evaluate the overall similarity between the problem and the solution. That sum is given by:

$$\begin{aligned}&\text{similarity(source, target)} = \text{w1} * \text{syntactic similarity} + \text{w2} * \text{semantic similarity}\\&+ \text{w3} * \text{contextual similarity} + \text{w4} * \text{discursive similarity} + \text{w5} * \text{morphological similarity}\\&+ \text{w6} * \text{Levenshtein similarity} + \text{w7} * \text{n - grams similarity}\end{aligned} \tag{6}$$

The weights w1 through w7 are associated with each metric, the sum of the weights is equal to 1. In order to define a weight configuration, we are going to use genetic algorithms. An individual is a set of weights and the evaluation function is a set of test cases which are presented to the system as problems. We then compare the generated diagram with the pre defined diagram (established by software engineers). The idea of weight optimization is to minimize the distance between diagrams.

Case Adaptation

The adaptation process uses two algorithms: one to adapt the solution chosen for each phase and other to join all diagrams into one (remember that each diagram represents only one phrase of the original text). After the process of mapping all sentences of the target text with similar source sentences, we load the specific diagram. The selected index and names of the source diagram are replaced with the nouns encountered in the target sentence. Other heuristics are used to perform the adaptation, for example: after a prepositional phrase usually comes an attribute of the previous noun phrase; the first name of the phrase is always an entity; emergence and action verbs usually represent entities methods; state verbs represent binary attributes; attributes can be identified by values expressed in attributive adjectives among others. An example of this process can be seen in the Fig. 15.

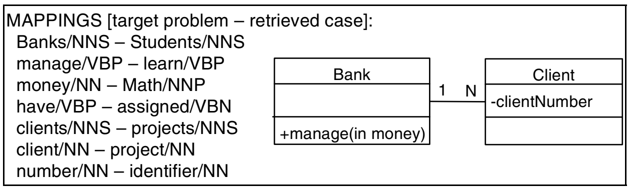


Fig. 15. Mappings of the adaptation process

8 Example of Use

This section presents an example of how REBUILDER UML can be used by a software designer. This example shows how design knowledge can be retrieved and stored for reuse.

Suppose that a designer is starting the design of an information system for a high school. S/he has already the system's analysis done, in the form of use cases (an UML diagram type used for describing system requirements). From these use cases, some initial entities are extracted by the designer and drawn in the system. Fig. 16 shows the initial class diagram, representing one of the system's modules. This module is responsible for handling the information data about teachers, classes and rooms timetables.

One of the tools available to the designer is the retrieval of similar designs from the case library. The designer can retrieve three types of objects: packages (a complete class diagram, what we call a case), classes or interfaces. Imagine that s/he selects the package object and clicks on the retrieval command. REBUILDER UML retrieves the number of diagrams defined by the designer (in this situation three cases). Fig. 17 presents one of the retrieved cases (the most similar one).

After the new diagram is completed, the designer can submit the diagram to the KB administrator (see Fig. 18). This implies that the designer considers the diagram correct and ready for being stored in the KB, for later reuse. This new diagram goes into a list of unconfirmed cases of the case library. The KB administrator has the task of examining these diagrams more carefully, deciding which are going to be

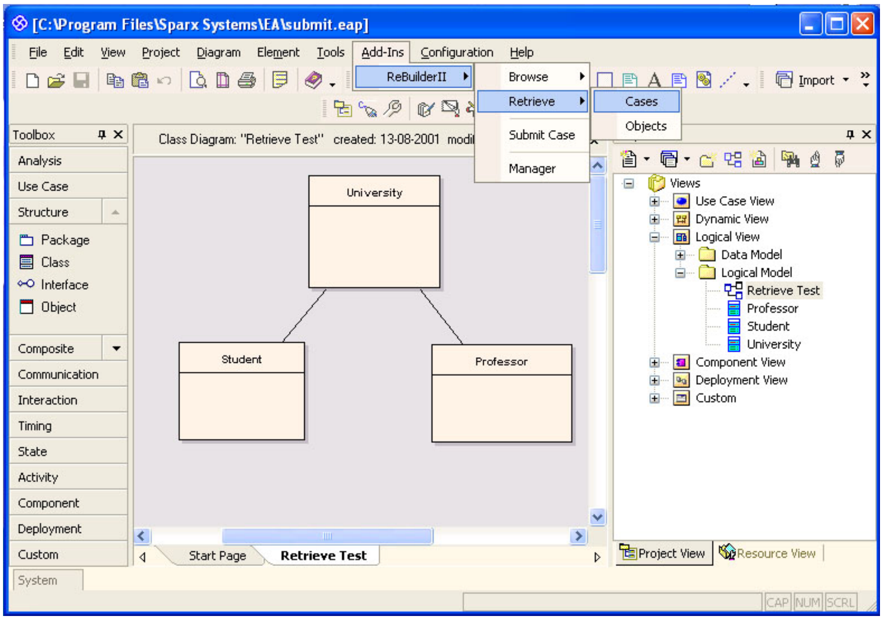


Fig. 16. The diagram used as query in the retrieval example

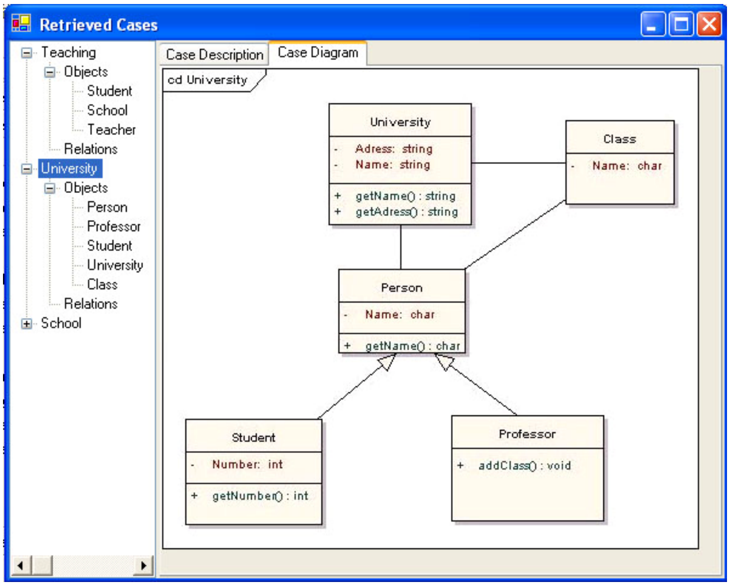


Fig. 17. One of the retrieved cases using the diagram in Fig. 16 as the query

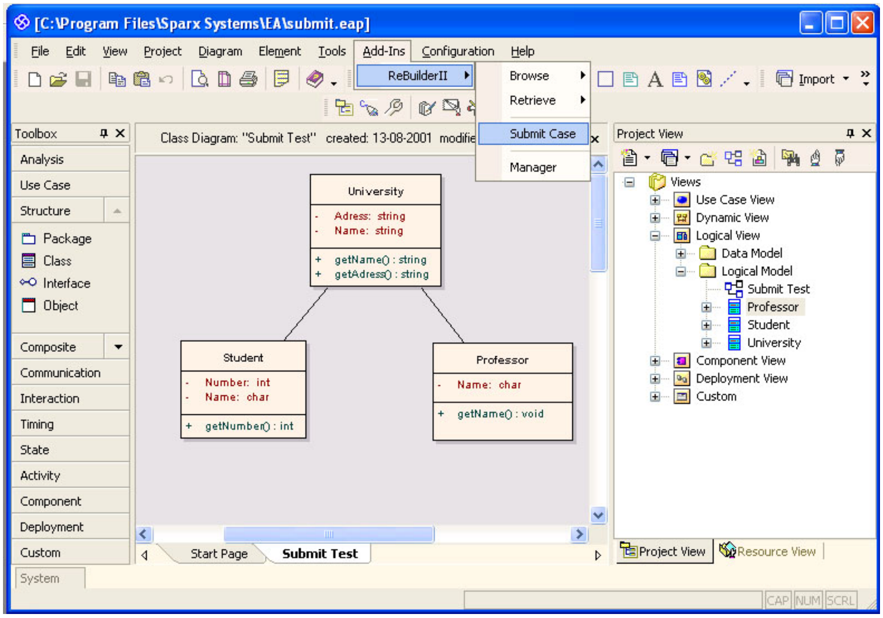


Fig. 18. The submission of a completed diagram in REBUILDER UML

transformed into cases, going to the list of confirmed cases (ready to be reused), and which are going to the list of obsolete cases not being used by the system.

9 Experimental Work

The experimental work developed in REBUILDER UML comprises three analysis axis: recall and precision results, retrieval time performance, and user experiments. The case base used has 29 cases, each case with an average number of 10 classes. Each class has 4 to 7 attributes, and 5 to 14 methods. In the first type of analysis 16 queries were defined, comprising only classes and relations (with an average of 4 classes and no attributes nor methods). A set of reference cases from the case base was defined for each query. The queries were then run in REBUILDER UML varying the size of the retrieval set (from 1 to 10 cases). The retrieved cases were analyzed and the following measures were computed:

$$\text{Recall} = \frac{\#(\text{ReferenceCases} \cap \text{RetrievedCases})}{\#\text{ReferenceCases}} \tag{7}$$

$$\text{Precision} = \frac{\#(\text{ReferenceCases} \cap \text{RetrievedCases})}{\#\text{RetrievedCases}} \tag{8}$$

$$\text{Accuracy} = \frac{\#(\text{ReferenceCases} \cap \text{RetrievedCases})}{\#(\text{ReferenceCases} \cup \text{RetrievedCases})} \tag{9}$$

$$\text{F-Measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{10}$$

Fig. 19 presents the results obtained for retrieval set sizes for 1 to 10. Notice that the best retrieval set size is five and that accuracy values are around 35% due to the lack of attributes and methods in the queries.

Fig. 20 shows the results for the second analysis, which was the assessment of the retrieval time performance. We have executed six experiments with different case base sizes from 5 to 29. In each experiment we have run 30 queries to assess the average retrieval time of the algorithms used in REBUILDER UML. As can be seen from the figure, retrieval times are low (below 3 seconds) using a normal PC (1 Gb RAM; 2 GHz dual core; windows XP), but further testing is needed to assess the performance with bigger case bases.

The last analysis was made with eight users with software engineering experience. Three tests were defined, comprising a natural language text describing a software system to be modeled. The first test had only one sentence and was easy to design. The second and third tests had about ten sentences and they are much more complex. The goal was for a user to design a UML class diagram corresponding to the text of each test. A reference diagram for each test was defined: the diagram for test 1 has 9 UML elements, test 2 has 36 elements and test 3 has 43 elements (when are referring to UML elements, we mean classes, relations and attributes). We have defined two test groups: one that was going to use the Enterprise Architect (EA) editor and REBUILDER UML to design the diagrams, and the second group only with EA.

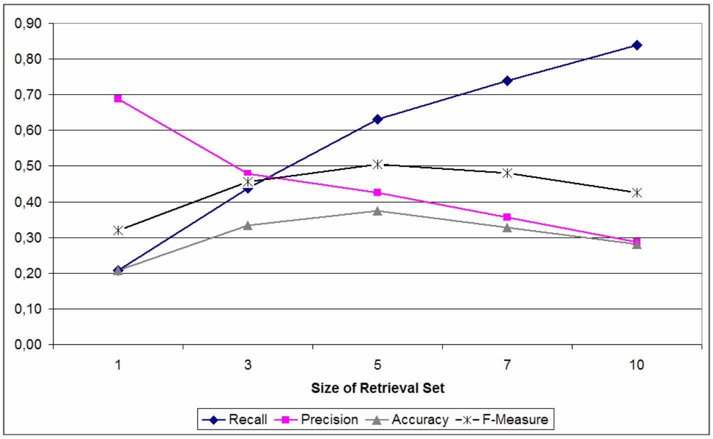


Fig. 19. Recall, precision, accuracy and f-measure experimental results

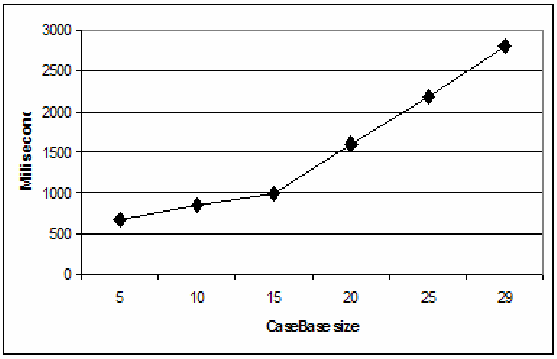


Fig. 20. Average retrieval time (milliseconds) results by query for different case base sizes

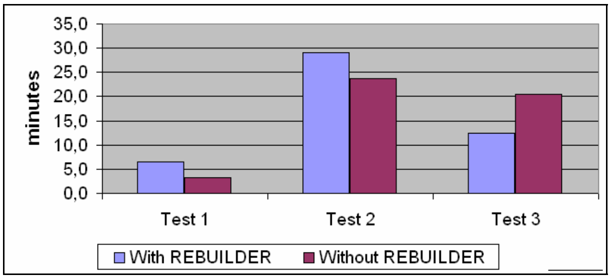


Fig. 21. Average user design time (minutes) for the three tests

We have measured the time each user needed to design each test diagram, and we also identified errors in user diagrams by comparing with the reference ones. The time results are presented in Fig. 21 and error results in Fig. 22. The time results show

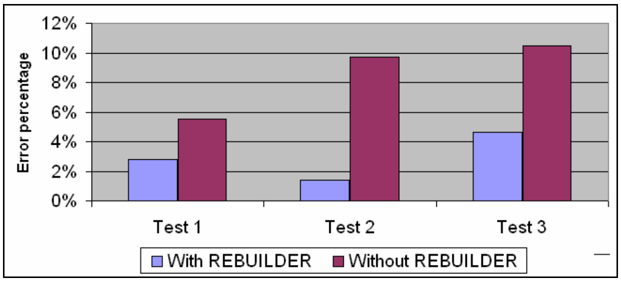


Fig. 22. Average percentage of user errors for the three tests

worse times for users with REBUILDER UML in the first two tests due to the time that users need to take in learning how to use the retrieval tools. But in the third test there is a clear improvement from the users that are using REBUILDER UML (the tests were presented in sequence, from 1 to 3). In the error results, it is clear that using REBUILDER UML and reusing cases is a major benefit for software design. Another major benefit of using our system is that all the diagrams made by users using REBUILDER UML have methods that came from the retrieved diagrams. This allows a faster development of the system being designed, due to code reuse.

10 Related Work

In this work, there is two main types of related works: research related with intelligent CASE tools and systems that can transform software requirements and specifications from text into UML diagrams or other types of software modeling diagrams. In this section we first explore intelligent CASE tools, and then systems similar to the TextToDiagram module.

10.1 Intelligent CASE Tools

REBUILDER – UML descends from REBUILDER I (Gomes et al. 2002, Gomes et al. 2004), and it has two main research influences: CBR software reuse systems and software reuse systems. This section explores these types of systems.

Fernández-Chamizo (Fernández-Chamizo et al. 1996) presented a CBR approach for software reuse based on the reuse and design of Object-Oriented code. Cases represent three types of entities: classes, methods and programming recipes, thus allowing the retrieval of these types of objects. Cases comprise a lexical description (problem), a solution (code) and a justification (code properties). It uses a lexical retrieval algorithm using a natural language query, and a conceptual retrieval using an entity and slot similarity measures. Déjá Vu (Smyth et al. 1992) is a CBR system for code generation and reuse using hierarchical CBR. Déjá Vu uses a hierarchical case representation, indexing cases using functional features. The main improvement of this system is the adaptation-guided retrieval, which retrieves cases based on the case adaptation effort instead of the similarity with the target problem. CAESER (Fouqué et al. 1993) is another code reuse CBR tool. It works at the code level and uses dataflow analysis to acquire functional indexes. The user can retrieve cases from the case library using a prolog-like query goal, which is used by the system to retrieve similar functions. Althoff (Althoff et al. 1997) have a different approach to software reuse and design. Instead of reusing code, they reuse system requirements and associated software development knowledge.

The RSL (Burton et al. 1987) is a software design system that combines several software design tools and library retrieval tools. It allows the reuse of code and design knowledge, and provides several software design tools to work the retrieved objects. RSL uses automatic indexing of components by scanning design documents and code files for specially labeled reuse component statements. Component retrieval can be done using a natural-language query, or using attribute search. Component ranking is an interactive and iterative process between RSL and the user. Prieto-Díaz (Prieto-Díaz 1991) approach to code reuse is based on a faceted classification of software components. Faceted classification has two different aspects according to Prieto-Díaz, it must classify components by the functions it performs and by the environment it works in. Conceptual graphs are used to organize facets, and a conceptual closeness measure is used to compute similarity between facets. He has identified six facets for software components: function, object, medium, system type, functional area and setting. Other work on facets is described by Liao (Liao et al. 1999), where he describes a hybrid similarity scheme using facets but where each facet can have multiple values. Borgo (Borgo et al. 1997) uses WordNet for retrieval of object oriented components. His system uses a graph structure to represent both the query and the components in memory. The retrieval mechanism uses a graph matching algorithm returning the identifiers of all components whose description is subsumed by the query. WordNet is also used for node matching. Helm (Helm et al. 1991) also presents a system for retrieval of Object-Oriented components based on the class source code, and the class documentation, using natural language queries.

10.2 Systems That Transform Text into Software Modeling Diagrams

There are some research groups investigating ways of building conceptual models from software requirements expressed in natural language, we now provide a brief overview of some of these efforts and some concluding remarks.

Illieva (Illieva et al. 2005) use a methodology that can be split into four phases. The first phase of linguistic analysis does the POS tagging of the text followed by chunking (shallow parsing), the second phase of tabular representation identifies the subject, the verb and the object of the various sentences, the third phase of semantic net representation does a translation of subjects and objects to entities, and verbs and prepositions to relations among these entities, finally the fourth phase produce a class diagram representation according to some rules from the semantic net.

In NL-OOPS (Mich 1996) it is used the PLN system LOLITA, which does morphological, syntactic, semantic and pragmatic analysis of the input text and stores the result in a semantic net. NL-OOPS extracts knowledge from this semantic net: static nodes are mapped to classes, associations and attributes, dynamic nodes are mapped to methods. Research groups from Birmingham and Indianapolis (Bryant et al. 2003) developed an approach that uses formal verification of requirements and represent them in XML. Afterwards the requirements are specified according to a two level grammar and finally this specification can be mapped to JAVA, XMI/UML or

VDM++. Li, Dewar and Pooley (Li et al. 2005) developed a methodology that does POS tagging, followed by a simplification step that transforms the original text into a text with the triples: subject, verb and object (SVO) and finally this simplified text is mapped to class diagrams using predefined rules.

11 Conclusions and Future Work

This paper presents a CASE tool capable of helping software designers build diagrams with less errors and at the same time reusing code (from the methods). It is our opinion that it also decreases the time needed to design the UML diagram, but further experimental work needs to be done. There are some limitations in REBUILDER UML that are being addressed, like ontology development. We are working on a tool for extracting ontologies semi automatically, so that it can be used to help the system administrator to develop the ontology. Another issue, also dealing with the ontology is the integration of tools to help the ontology management.

This paper also presents the module TextToDiagram, which makes the translation of natural language text into a class diagram. This approach is based on CBR and is supported by an ontology. The main advantage of our approach is the flexibility that it possesses in relation to its adaptation to the user way of modeling software systems, and to the vocabulary used. It also enables the system to learn new knowledge, thus covering other regions of the search translation space.

References

Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7(1), 39–59 (1994)

Althoff, K.-D.: Case-based reasoning, in Handbook on Software Engineering and Knowl-edge Engineering. In: Chang, S.K. (ed.) World Scientific, pp. 549–588.

Althoff, K.-D., et al.: Case-Based Reasoning for Experimental Software Engineering. Fraunhofer IESE, Berlin (1997)

Boehm, B.: A Spiral Model of Software Development and Enhancement. IEEE Press, Los Alamitos (1988)

Borgo, S., et al.: Using a Large Linguistic Ontology for Internet-Based Retrieval of Ob-ject-Oriented Components. In: SEKE 1997. 9th International Conference on Software Engineering and Knowledge Engineering, Madrid, Spain, Knowledge Systems Institute, Illinois (1997)

Bryant, B., et al.: From natural language requirements to executable models of software components. In: The Monterey Workshop on software engineering for embedded systems: from requirements to implementation (2003)

Burton, B.A., et al.: The Reusable Software Library, pp. 25–32. IEEE Software, Los Alamitos (1987)

Coulange, B.: Software Reuse. Springer, London (1997)

Day, W.H.E.: Properties of Levenshtein metrics on sequences. Bull. Math. Biol. 46 (1984)

Fernández-Chamizo, C., et al.: Supporting Object Reuse through Case-Based Reasoning. In: Smith, I., Faltings, B.V. (eds.) EWCBR 1996. LNCS, vol. 1168, Springer, Heidelberg (1996)

Fouqué, G., Matwin, S.: Compositional Software reuse with Case-Based Reasoning. In: CAIA 1993. 9th Conference on Artificial Intelligence for Applications, Orlando, FL, USA, IEEE Computer Society Press, Los Alamitos (1993)

Gamma, E., et al.: Design Patterns: Elements of Reusable Object-Oriented Software, vol. 395. Addison-Wesley, Reading (1995)

Gomes, P.: A Case-Based Approach to Software Design. In: Department of Informatics Engineering, University of Coimbra, Coimbra (2004)

Gomes, P., et al.: Case Retrieval of Software Designs using WordNet. In: ECAI 2002. European Con-ference on Artificial Intelligence, Lyon, France, IOS Press, Amsterdam (2002)

Gomes, P., et al.: REBUILDER: A CBR Approach to Knowledge Management in Software Design. In: Melnik, G., Holz, H. (eds.) LSO 2004. LNCS, vol. 3096, Springer, Heidelberg (2004)

Helm, R., Maarek, Y.S.: Integrating Information Retrieval and Domain Specific Approaches for Browsing and Retrieval in Object-Oriented Class Libraries. In: Object-Oriented Programming Systems, Languages, and Applications, Phoenix, AZ USA, ACM Press, New York (1991)

Ilieva, M.G., Ormandjieva, O.: Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation. In: NLDB (2005)

Jacobson, I., Griss, M., Jonsson, P.: Software Reuse: Architecture Process and Or-ganization for Business Success. ACM Press, New York (1997)

Kolodner, J.: Case-Based Reasoning: Morgan Kaufman (1993)

Kondrak, G.: N-Gram Similarity and Distance. In: SPIRE 2005. String Processing and Information Retrieval (2005)

Li, K., Dewar, R.G., Pooley, R.J.: Object-Oriented Analysis Using Natural Language Processing (2005)

Liao, S.Y., Cheung, L.S., Liu, W.Y.: An Object-Oriented System for the Reuse of Software Design Items. Journal of Object-Oriented Programming 11(8), 22–28 (1999)

Meyer, B.: Reusability: The Case for Object-Oriented Design, pp. 50–64. IEEE Software, Los Alamitos (1987)

Mich, L.: NL-OOPS: From Natural Language to Object-Oriented Requirements using the Natural Language Processing System LOLITA. Natural language engineering 2(2), 161–187 (1996)

Miller, G., et al.: Introduction to WordNet: an on-line lexical database. International Journal of Lexicography 3(4), 235–244 (1990)

Prieto-Diaz, R.: Implementing Faceted Classification for Software Reuse. Communica-tions of the ACM (May 1991)

Prieto-Diaz, R.: Status Report: Software Reusability. IEEE Software, Los Alamitos (1993)

Project, C.: OpenNLP, Statistical parsing of English sentences (2005)

Seco, N., Gomes, P., Pereira, F.C.: Using CBR for Semantic Analysis of Software Specifications. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, Springer, Heidelberg (2004)

Smyth, B., Cunningham, P.: Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design. In: ECAI 1992. 10th European Conference on Artificial Intelligence, John Wiley & Sons, Chichester (1992)

Tong, C., Sriram, D.: Artificial Intelligence in Engineering Design, vol. 1. Academic Press, London (1992)

Learning from Demonstration and Case-Based Planning for Real-Time Strategy Games

Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram

CCL, Cognitive Computing Lab, Georgia Institute of Technology, Atlanta, GA 30332/0280, USA
{santi,kinshuk,nsugandh,ashwin}@cc.gatech.edu

1 Introduction

Artificial Intelligence (AI) techniques have been successfully applied to several computer games. However, in the vast majority of computer games traditional AI techniques fail to play at a human level because of the characteristics of the game. Most current commercial computer games have vast search spaces in which the AI has to make decisions in real-time, thus rendering traditional search based techniques inapplicable. For that reason, game developers need to spend a big effort in hand coding specific strategies that play at a reasonable level for each new game. One of the long term goals of our research is to develop artificial intelligence techniques that can be directly applied to such domains, alleviating the effort required by game developers to include advanced AI in their games.

Specifically, we are interested in real-time strategy (RTS) games, that have been shown to have huge decision spaces that cannot be dealt with search based AI techniques (Aha et al. 2005;Buro 2003). In this paper we will present a case-based planning architecture which combines ideas from case-based reasoning (Aamodt and Plaza 1994;Kolodner 1993) and planning (Fikes and Nilsson 1971). The proposed architecture integrates planning and execution and is capable of dealing with both the vast decision spaces and the real-time component of RTS games (See Section 2 for a brief introduction to case-based reasoning and planning). Moreover, applying case-based planning to RTS games requires a set of cases with which to construct plans. To deal with this issue, we propose to extract behavioral knowledge from expert demonstrations (i.e. an expert plays the game and our system observes), and store it in the form of cases. Then, at performance time, the system will retrieve the most adequate behaviors observed from the expert and will adapt them to the situation at hand.

As we said before, one of the main goals of our research is to create AI techniques that can be used by game manufacturers to reduce the effort required to develop the AI component of their games. Developing the AI behavior for an automated agent that plays a RTS is not an easy task, and requires a large coding and debugging effort. Using the architecture presented in this paper the game developers will be able to specify the AI behavior just by demonstration; i.e. instead of having to code the behavior using a programming language, the behavior can be specified simply by *demonstrating* it to the system. If the system shows an incorrect behavior in any particular situation, instead of having to find the bug in the program and fix it, the game developers can simply demonstrate the correct action in the particular situation. The system will then incorporate that information in its case base and will behave better in the future.

Another contribution of the work presented in this paper is on presenting an integrated architecture for case-based planning and execution. In our architecture, plan retrieval, composition, adaptation, and execution are interleaved. The planner keeps track of all the open goals in the current plan (initially, the system starts with the goal of winning the game), and for each open goal, the system retrieves the most adequate behavior in the case base depending on the current game state. This behavior is then added into the current plan. When a particular behavior has to be executed, it is adapted to match the current game state and then it is executed. Moreover, each individual action or sub-plan inside the plan is constantly monitored for success or failure. When a failure occurs, the system attempts to retrieve a better behavior from the case base. This interleaved process of case based planning and execution allows the system to reuse the behaviors extracted from the expert and apply them to play the game.

Our application domain is the full game WARGUS (a clone of the popular game Warcraft II). In order to validate our approach, we will deal with the full WARGUS game at the same level of granularity as a human would play it without restrictions. Previous research has shown that real-time strategy games (RTS) such as WARGUS have huge decision spaces that cannot be dealt with search based AI techniques (Aha et al. 2005;Buro 2003). Thus, WARGUS is a good testbed for our approach.

The rest of the paper is organized as follows. Section 2 presents a brief introduction to case-based reasoning (CBR) and planning. Then, Section 3 introduces the proposed case-based planning architecture and its main modules. After that, Section 4 briefly explains the behavior representation language used in our architecture. Section 5 explains the case extraction process. Then sections 6 and 7 present the planning module and the case based reasoning module respectively. Section 8 summarizes our experiments. Section 9 presents a summary of related work. Finally, the paper finishes with the conclusions section.

2 Case-Based Reasoning and Planning

Case-based reasoning (CBR) (Aamodt and Plaza 1994;Kolodner 1993) is problem solving methodology based on reutilizing specific knowledge of previously experienced and concrete problem situations (*cases*).

The activity of a case-based reasoning system can be summarized in the CBR cycle, shown in Figure 1. The CBR cycle consists of four stages: Retrieve, Reuse, Revise and Retain. In the *Retrieve* stage, the system selects a subset of cases from the case base that are relevant to the current problem. The *Reuse* stage adapts the solution of the cases selected in the retrieve stage to the current problem. In the *Revise* stage, the obtained solution is examined by an oracle, which gives the correct solution (as in supervised learning). Finally, in the *Retain* stage, the system decides whether to incorporate the new solved case into the case base or not.

Automated planning (Fikes and Nilsson 1971) is a branch of artificial intelligence concerned about the generation of action sequences or strategies to achieve specific goals. Traditional planning systems use search strategies to explore the space of all possible action sequences and find one plan that will achieve the desired goal. The main issue in automated planning is the fact that the space of all possible plans is enormous, and it is not possible to explore it systematically for non-toy problems. To deal with that

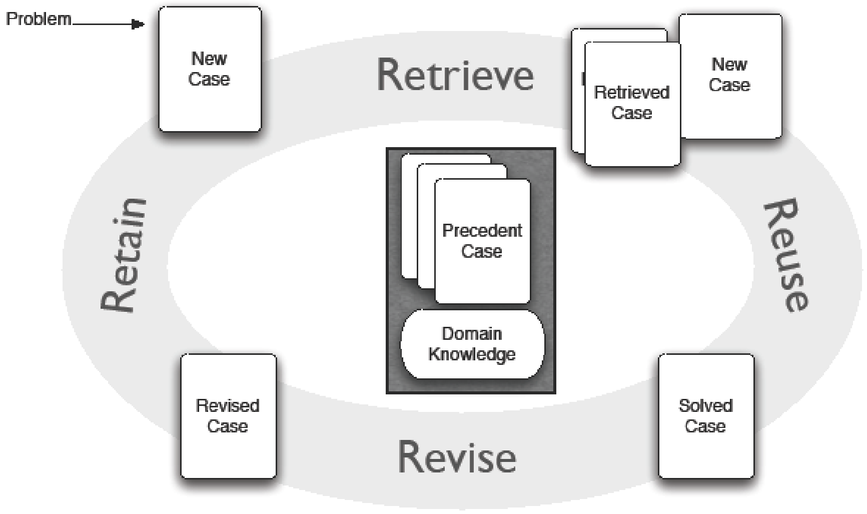


Fig. 1. The case-based reasoning cycle

problem several strategies have been proposed. For instance, hierarchical task network (HTN) planning (Nau et al. 1999), tries to deal with larger problems than standard planning by building hierarchical plans instead of flat sequences of actions.

Case-based planning (Spalazzi 2001) consists of applying case-based reasoning ideas to the problem of automated planning. A case-based planning system reuses previous planning experiences in order to solve new planning problems. This might involve reusing previous successful plans or remembering past planning failures for example. Several approaches for case-based planning have been presented in the past (see (Spalazzi 2001) for an overview).

3 Case-Based Planning in WARGUS

In this section we will present an overview of the proposed architecture to learn behaviors from expert demonstrations in the WARGUS domain. Let us begin by briefly describing the WARGUS game.

WARGUS (See Figure 2) is a real-time strategy game where each player's goal is to remain alive after destroying the rest of the players. Each player has a series of troops and buildings and gathers resources (gold, wood and oil) in order to produce more troops and buildings. Buildings are required to produce more advanced troops, and troops are required to attack the enemy. In addition, players can also build defensive buildings such as walls and towers. Therefore, WARGUS involves complex reasoning to determine where, when and which buildings and troops to build. For example, the map shown in Figure 2 is a 2-player version of the classical map “Nowhere to run nowhere to hide”, with a wall of trees that separates the players. This maps leads to complex strategic reasoning, such as building long range units (such as catapults or ballistae) to attack the other player before the wall of trees has been destroyed, or tunneling early in the game through the wall of trees trying to catch the enemy by surprise.

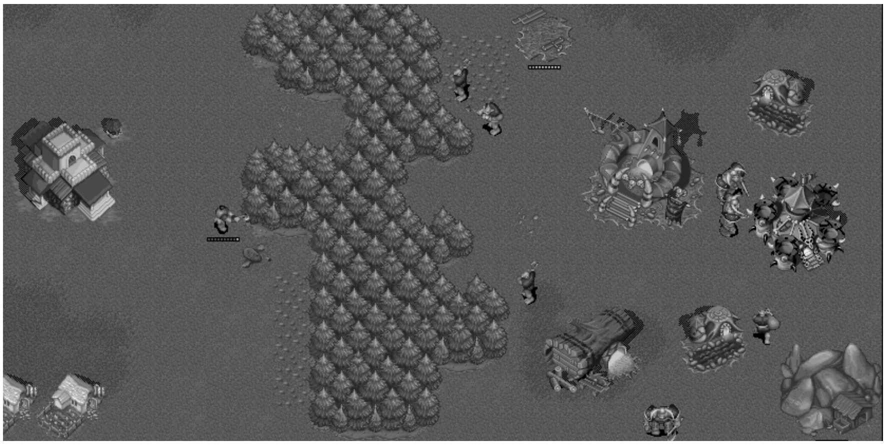


Fig. 2. A screenshot of the WARGUS game

Traditionally, games such as WARGUS use handcrafted behaviors for the built-in AI. Creating such behaviors requires a lot of effort, and even after that, the result is that the built-in AI is static and easy to beat (since humans can easily find holes in the computer strategy). The goal of the work presented in this paper is to ease the task of the game developers to create behaviors for these games, and to make them more adaptive. Our approach involves learning behaviors from expert demonstrations to reduce the effort of coding the behaviors, and use the learned behaviors inside a case-based planning system to reuse them for new situations. Figure 3 shows an overview of our case-based planning approach. Basically, we divide the process in two main stages:

- *Behavior acquisition:* During this first stage, an expert plays a game of WARGUS and the trace of that game is stored. Then, the expert annotates the trace explaining the goals he was pursuing with the actions he took while playing. Using those annotations, a set of behaviors are extracted from the trace and stored as a set of cases. Each case is a triple: situation/goal/behavior, representing that the expert used a particular behavior to achieve a certain goal in a particular situation.
- *Execution:* The execution engine consists of two main modules, a real-time plan expansion and execution (RTEE) module and a behavior generation (BG) module. The RTEE module maintains an execution tree of the current active goals and subgoals and which behaviors are being executed to achieve each of the goals. Each time there is an open goal, the RTEE queries the BG module to generate a behavior to solve it. The BG then retrieves the most appropriate behavior from its case base, and sends it to the RTEE. Finally, when the RTEE is about to start executing a behavior, it is sent back to the BG module for adaptation. Notice that this *delayed adaptation* is a key feature different from traditional CBR required for real-time domains where the environment continuously changes.

In the following sections we will present each of the individual components of our architecture.

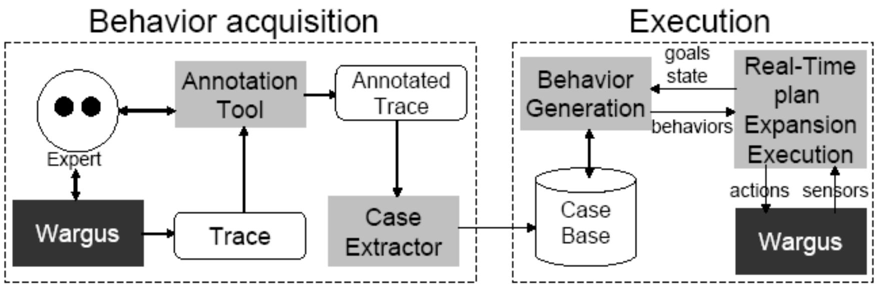


Fig. 3. Overview of the proposed case-based planning approach

4 A Behavior Reasoning Language

In this section we will present the Behavior Reasoning Language used in our approach, designed to allow a system to learn behaviors, represent them, and to reason about the behaviors and their intended and actual effects. Our language takes ideas from the STRIPS (Fikes and Nilsson 1971) planning language, and from the ABL (Mateas and Stern 2002) behavior language, and further develops them to allow advanced reasoning and learning capabilities over the behavior language.

The basic constituent piece is the *behavior*. A behavior has two main parts: a *declarative* part and a *procedural* part. The declarative part has the purpose of providing information to the system about the intended use of the behavior, and the procedural part contains the executable behavior itself. The declarative part of a behavior consists of three parts:

- A *goal*, that is a representation of the intended goal of the behavior. For every domain, an ontology of possible goals has to be defined. For instance, a behavior might have the goal of “having a tower”.
- A set of *preconditions* that must be satisfied before the behavior can be executed. For instance, a behavior can have as preconditions that a particular peasant exists and that a desired location is empty.
- A set of *alive conditions* that represent the conditions that must be satisfied during the execution of the behavior for it to have chances of success. If at some moment during the execution, the alive conditions are not met, the behavior can be stopped, since it will not achieve its intended goal. For instance, the peasant in charge of building a building must remain alive; if he is killed, the building will not be built.

Notice that unlike classical planning approaches, postconditions cannot be specified for behaviors, since a behavior is not guaranteed to succeed. Thus, we can only specify what goal a behavior pursues.

The procedural part of a behavior consists of executable code that can contain the following constructs: *sequence*, *parallel*, *action*, and *subgoal*, where an *action* represents the execution of a basic action in the domain of application (a set of basic actions must be defined for each domain), and a *subgoal* means that the execution engine must find another behavior that has to be executed to satisfy that particular subgoal. Specifically, three things need to be defined for using our language in a particular domain:

- A set of *basic actions* that can be used in the domain. For instance, in WARGUS we define actions such as *move*, *attack*, or *build*.
- A set of *sensors*, that are used in the behaviors to obtain information about the current state of the world. For instance, in WARGUS we might define sensors such as *numberOfTroops*, or *unitExists*. A sensor might return any of the standard basic data types, such as boolean or integer.
- A set of *goals*. Goals can be structured in a specialization hierarchy in order to specify the relations among them.

A goal might have parameters, and for each goal a function *generateSuccessTest* must be defined, that is able to generate a condition that is satisfied only when the goal is achieved. For instance, *HaveUnits(TOWER)* is a valid goal in our gaming domain and it should generate the condition *UnitExists(TOWER)*. Such condition is called the *success test* of the goal. Therefore, the goal definition can be used by the system to reason about the intended result of a behavior, while the success test is used by the execution engine to verify whether a particular behavior succeeds at run time.

Summarizing, our behavior language is strongly inspired by ABL, but expands it with declarative annotations (expanding the representation of goals and defining alive and success conditions) to allow reasoning.

5 Behavior Acquisition in WARGUS

As Figure 5 shows, the first stage of our case-based planning architecture consists of acquiring a set of behaviors from an expert demonstration. Let us present this stage in more detail.

One of the main goals of this work is to allow a system to learn a behavior by simply observing a human, in opposition to having a human encoding the behavior in some form of programming language. To achieve that goal, the first step in the process must be for the expert to provide the demonstration to the system. In our particular application domain, WARGUS, an expert simply plays a game of WARGUS (against the built-in AI, or against any other opponent). As a result of that game, we obtain a game trace, consisting of the set of actions executed during the game. Table 1 shows a snippet of a real trace from playing a game of WARGUS. As the table shows, each trace entry contains the particular cycle in which an action was executed, which player executed the action, and the action itself. For instance, the first action in the game was executed at cycle 8, where player 1 made his unit number 2 build a "pig-farm" at the (26, 20) coordinates.

As Figure 3 shows, the next step is to annotate the trace. For this process, the expert uses a simple annotation tool that allows him to specify which goals he was pursuing for each particular action. To use such an annotation tool, a set of available goals has to be defined for the WARGUS domain.

In our approach, a *goal* $g = name(p_1, \ldots, p_n)$ consists of a goal name and a set of parameters. For instance, in WARGUS, these are the goal types we defined for our experiments:

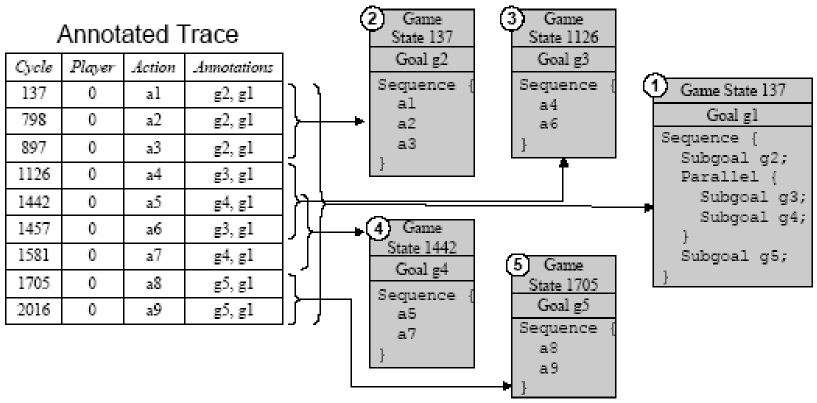
- *WinWargus(player):* representing that the action had the intention of making the player *player* win the game.
- *DefeatPlayer(player,opponent):* the expert wanted to defeat a particular opponent *opponent*.
- *SetupResourceInfrastructure(player,peasants,farms):* indicates that the expert wanted to create a good resource infrastructure for player *player*, that at least included *peasants* number of peasants and *farms* number of farms.
- *AbsoluteBuildUnits(player,type,number)*: the expert wanted to have at least *number* units of type *type*.
- *RelativeBuildUnits(player,type,number):* the expert wanted to have at least *number* units of type *type* in addition to the ones that he had. Notice the difference between the relative and the absolute goal, in the absolute one, the expert might specify, that he wants to have "at least 2 ballistas in total", and in the relative one, he can specify that he wants to have "at least 2 more ballistas in addition to the ones he already have".
- *KillUnit(unit):* representing that the action had the intention of killing the unit *unit*.
- *KillAllUnitsOfType(opponent,type)*: represents that the expert wanted to destroy all the units of type *type* belonging to the player *opponent*.
- *Research(advancement):* the expert wanted to complete the research of a particular advancement *advancement*. In WARGUS, player can research several advancements, that allow their troops to be more effective, or even give access to new kinds of troops.
- *AbsoluteGetResources(gold,wood,oil):* the action had the intention of increasing the resource levels at least to the specified levels in the parameters.
- *RelativeGetResources(gold,wood,oil):* the action had the intention of increasing the resource levels at least in the specified amount in the parameters. Again, the difference between the absolute and relative is that in the relative goal the expert specifies that he wants to have "at least 2000 gold coins", and in the relative on he specifies that he wants to obtain "at least 2000 more gold coins in addition to the gold he already has".

The fourth column of Table 1 shows the annotations that the expert specified for his actions. Since the snippet shown corresponds to the beginning of the game, the expert specified that he was trying to create a resource infrastructure and, of course, he was trying to win the game.

Finally, as Figure 3 shows, the annotated trace is processed by the *case extractor* module, that encodes the strategy of the expert in this particular trace in a series of cases. Traditionally, in the CBR literature cases consist of a problem/solution pair; in our system we extended that representation due to the complexity of the domain of application. Specifically, a case in our system is defined as a triple consisting of a game state, a goal and a behavior. See Section 7 for a more detailed explanation of our case formalism.

Table 1. Snippet of a real trace generated after playing WARGUS

Cycle	Player	Action	Annotation
8	1	Build(2,"pig-farm",26,20)	-
137	0	Build(5,"farm",4,22)	SetupResourceInfrastructure(0,5,2) WinWargus(0)
638	1	Train(4,"peon")	-
638	1	Build(2,"troll-lumber-mill",22,20)	-
798	0	Train(3,"peasant")	SetupResourceInfrastructure(0,5,2) WinWargus(0)
878	1	Train(4,"peon")	-
878	1	Resource(10,5)	-
897	0	Resource(5,0)	SetupResourceInfrastructure(0,5,2) WinWargus(0)
…	…	…	…



Cycle	Player	Action	Annotations
137	0	a1	g2, g1
798	0	a2	g2, g1
897	0	a3	g2, g1
1126	0	a4	g3, g1
1442	0	a5	g4, g1
1457	0	a6	g3, g1
1581	0	a7	g4, g1
1705	0	a8	g5, g1
2016	0	a9	g5, g1

```
Sequence {
  a1
  a2
  a3
}
```

```
Sequence {
  a4
  a6
}
```

```
Sequence {
  Subgoal g2;
  Parallel {
    Subgoal g3;
    Subgoal g4;
  }
  Subgoal g5;
}
```

```
Sequence {
  a5
  a7
}
```

```
Sequence {
  a8
  a9
}
```

Fig. 4. Extraction of cases from the annotated trace

In order to extract cases, the annotated trace is analyzed to determine the temporal relations among the individual goals appearing in the trace. For instance, if we look at the sample annotated trace in Figure 4, we can see that the goal *g2* was attempted *before* the goal *g3*, and that the goal *g3* was attempted *in parallel* with the goal *g4*. The kind of analysis required is a simplified version of the temporal reasoning framework presented by Allen (Allen 1983), where the 13 basic different temporal relations among events were identified. In our framework, we are only interested in knowing if two goals are pursued in sequence, in parallel, or if one is a subgoal of the other. We assume that if the temporal relation between a particular goal *g* and another goal *g'* is that *g* happens *during g'*, then *g* is a subgoal of *g'*. For instance, in Figure 4, *g2*, *g3*, *g4*, and *g5* happen *during g1*; thus they are considered subgoals of *g1*.

From temporal analysis, procedural descriptions of the behavior of the expert can be extracted. For instance, from the relations among all the goals in Figure 4, case number 1 (shown in the figure) can be extracted, specifying that to achieve goal *g1* in the particular game state in which the game was at cycle 137, the expert first tried to achieve goal *g2*, then attempted *g3* and *g4* in parallel, and after that *g5* was pursued. Then, for each one of the subgoals a similar analysis is performed, leading to four more cases. For example, case 3 states that to achieve goal *g2* in that particular game state, basic actions *a4* and *a6* should be executed sequentially.

At the end of the behavior acquisition process, the system has acquired a set of behaviors to achieve certain goals. If this process is repeated several times with several expert traces, the result is that the system will have a collection of behaviors to achieve different goals under different circumstances. The traces are split into individual cases that encode how to attempt individual goals. This allows the system to easily combine cases learnt from different traces at run-time, to come up with new strategies, result of combining parts of different strategies demonstrated to it. For instance, the system might decide to attempt the goal *WinWargus* using the behavior learnt from one trace, but then decide that, due to the current game state, it is better to attempt to kill a particular unit using a behavior learnt from another trace. This provides our system with a great degree of flexibility.

Notice that in our system we don't attempt any kind of generalization of the expert actions. If a particular expert action in the trace is *Build(5,"farm",4,22)*, that is exactly the action stored in a case. Thus, using the learnt cases to play a new scenario in WARGUS, it is very likely that the particular values of the parameters in the action are not the most appropriate for the new scenario (for instance, it might be the case that in the new map the coordinates 4,22 correspond to a water location, and thus a farm cannot be built there). In our system, the adaptation module of the BG component in our architecture is in charge of adapting the parameters of an action to a new scenario (see Section 7).

6 Real-Time Plan Expansion and Execution

During execution time, our system will use the set of cases collected from expert traces to play a game of WARGUS. In particular two modules are involved in execution: a real-time plan expansion and execution module (RTEE) and a behavior generation module (BG). Both modules collaborate to maintain a current *partial plan tree* that the system is executing.

A partial plan tree (that we will refer to as simply the "plan") in our framework is represented as a tree consisting of two types of nodes: *goals* and *behaviors* (following the same idea of the task/method decomposition (chandrasekaran 1990)). Initially, the plan consists of a single goal: "win the game". Then, the RTEE asks the BG module to generate a behavior for that goal. That behavior might have several subgoals, for which the RTEE will again ask the BG module to generate behaviors, and so on. For instance, on the right hand side of Figure 5 we can see a sample plan, where the top goal is to "win". The behavior assigned to the "win" goal has three subgoals, namely "build base", "build army" and "attack". "build base" has already a behavior assigned that has no subgoals, and the rest of subgoals still don't have an assigned behavior. When a goal still doesn't have an assigned behavior, we say that the goal is *open*.

Additionally, each behavior in the plan has an associated state. The state of a behavior can be: *pending*, *executing*, *succeeded* or *failed*. A behavior is pending when it still has not started execution, and its status is set to failed or succeeded after its execution ends, depending on whether it has satisfied its goal or not. A goal that has a behavior assigned and where the behavior has failed is also considered to be open (since a new behavior has to be found for this goal).

Open goals can be either *ready* or *waiting*. An open goal is ready when all the behaviors that had to be executed before this goal have succeeded, otherwise, it is waiting. For instance, in Figure 5, "behavior 0" is a sequential behavior and therefore the goal "build army" is ready since the "build base" goal has already succeeded and thus "build army" can be started. However, the goal "attack" is waiting, since "attack" has to be executed after "build army" succeeds.

The RTEE is divided into two separate modules that operate in parallel to update the current plan: the *plan expansion* module and the *plan execution* module. The plan expansion module is constantly querying the current plan to see if there is any ready open goal. When this happens, the open goal is sent to the BG module to generate a behavior for it. Then, that behavior is inserted in the current plan, and it is marked as pending.

The plan execution module has two main functionalities: a) check for basic actions that can be sent directly to the game engine, b) check the status of plans that are in execution. Specifically it works as follows:

- For each pending behavior, the execution module evaluates the preconditions, and the behavior starts its execution when they are met.
- If any of the execution behaviors have any basic actions that are ready to be executed, the execution module sends those actions to WARGUS to be executed.
- Whenever a basic action succeeds or fails, the execution module updates the status of the behavior that contained it. When a basic action succeeds, the executing behavior can continue to the next step. When a basic action fails, the behavior is marked as failed, and thus its corresponding goal is open again (thus, the system will have to find another plan for that goal).

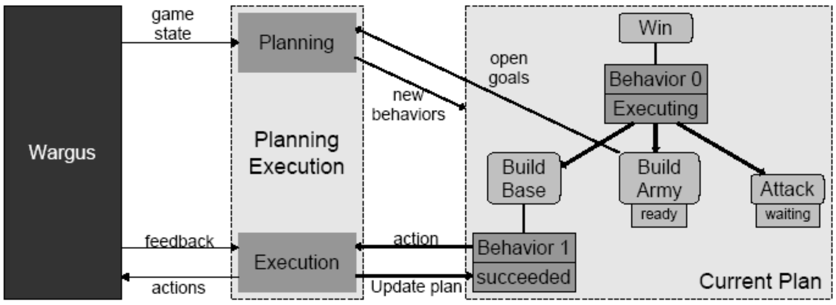
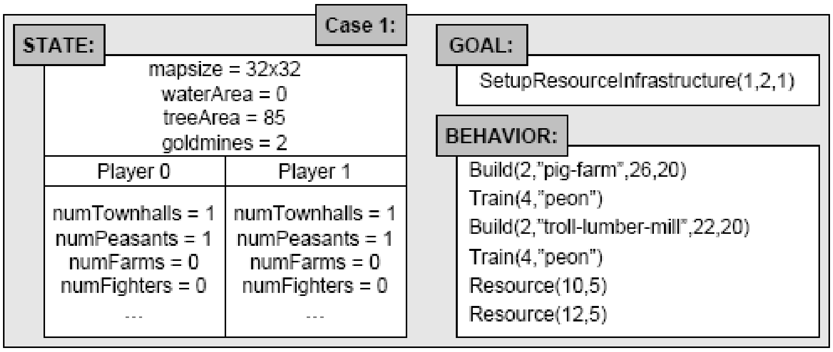


Fig. 5. Interleaved plan expansion and execution


Case 1:

STATE:

mapsize = 32x32
waterArea = 0
treeArea = 85
goldmines = 2

Player 0	Player 1

numTownhalls = 1
numPeasants = 1
numFarms = 0
numFighters = 0
...
	numTownhalls = 1
	numPeasants = 1
	numFarms = 0
	numFighters = 0
	...

GOAL:

```
SetupResourceInfrastructure(1,2,1)
```

BEHAVIOR:

```
Build(2,"pig-farm",26,20)
Train(4,"peon")
Build(2,"troll-lumber-mill",22,20)
Train(4,"peon")
Resource(10,5)
Resource(12,5)
```

Fig. 6. Example of a case extracted from an expert trace for the WARGUS game

- The execution module periodically evaluates the alive conditions and success conditions of each behavior and basic action. If the alive conditions of an executing behavior are not satisfied, the behavior is marked as failed, and its goal is open again. If the success conditions of a behavior are satisfied, the behavior is marked as succeeded. Exactly the same process is followed for basic actions that are currently executing.
- Finally, if a behavior is about to be executed and the current game state has changed since the time the BG module generated it, the behavior is handed back to the BG and it will pass again through the *adaptation* phase (see Section 7) to make sure that the plan is adequate for the current game state.

7 Behavior Generation

The goal of the BG module is to generate behaviors for specific goals in specific scenarios. Therefore, the input to the BG module is a particular scenario (i.e. the current game state in WARGUS) and a particular goal that has to be achieved (e.g. "Destroy the Enemy's Cannon Tower"). To achieve that task, the BG system uses two separate processes: *case retrieval* and *case adaptation* (that correspond to the first two processes of the 4R CBR model (Aamodt and Plaza 1994)).

Notice that to solve a complex planning task, several subproblems have to be solved. For instance, in our domain, the system has to solve problems such as how to build a proper base, how to gather the necessary resources, or how to destroy each of the units of the enemy. All those individual problems are different in nature, and in our case base we might have several cases that contain different behaviors to solve each one of these problems under different circumstances. Therefore, in our system we will have a *heterogeneous* case base. To deal with this issue, we propose to include in each case the particular *goal* that it tries to solve (see Section 5 for a list of defined goals in our WARGUS implementation). Therefore we represent cases as triples: $c = \langle S, G, B \rangle$, where S is a particular game state, G is a goal, and B is a behavior; representing that $c.B$ is a good behavior to apply when we want to pursue goal $c.G$ in a game state similar to $c.S$.

Figure 6 shows an example of a case, where we can see the three elements: a game description, that contains some general features about the map and some information about each of the players in the game; a particular goal (in this case, building the resource infrastructure of player "1"); and finally a behavior to achieve the specified goal in the given map. In particular, we have used a game state definition composed of 35 features that try to represent each aspect of the WARGUS game. Twelve of them represent the number of troops (number of fighters, number of peasants, and so on), four of them represent the resources that the player disposes of (gold, oil, wood and food), fourteen represent the description of the buildings (number of town halls, number of barracks, and so on) and finally, five features represent the map (size in both dimensions, percentage of water, percentage of trees and number of gold mines).

7.1 Behavior Retrieval

The case retrieval process uses a standard nearest neighbor algorithm but with a similarity metric that takes into account both the goal and the game state. Specifically, we use the following similarity metric:

$$d(c_1, c_2) = \alpha d_{GS}(c_1.S, c_2.S) + (1-\alpha) d_G(c_1.G, c_2.G)$$

where d_{GS} is a simple Euclidean distance between the game states of the two cases (where all the attributes are normalized between 0 and 1), d_G is the distance metric between goals, and α is a factor that controls the importance of the game state in the retrieval process (in our experiments we used $\alpha = 0.5$). To measure distance between two goals $g_1 = name_1(p_1, ..., p_n)$ and $g_2 = name_2(p_1, ..., p_m)$ we use the following distance:

$$d_G(g_1, g_2) = \begin{cases} \sqrt{\sum_{i=1...n} \left(\dfrac{p_i - q}{P_i} \right)^2} & if\ name_1 = name_2 \\ 1 & otherwise \end{cases}$$

where P_i is the maximum value that the parameter *i* of a goal might take (we assume that all the parameters have positive values). Thus, when $name_1 = name_2$, the two goals will always have the same number of parameters and the distance can be computed using an Euclidean distance among the parameters. The distance is maximum (1) otherwise.

Notice that this scheme assumes that if two goals have different names, they are totally different. This is just a simplification. A better solution will be to hand-specify the base similarity between goal types. For instance, we could tell the system that the goals *AbsoluteBuildUnits* and *RelativeBuildUnits* are more similar than the goals *AbsoluteBuildUnits* and *Research.* So, in case that there is no behavior in the case base for the goal *AbsoluteBuildUnits*, the system might find a behavior associated with *RelativeBuildUnits* and still be suitable.

7.2 Behavior Adaptation

The result of the retrieval process is a case that contains a behavior that achieves a goal similar to the requested one by the RTEE, and that can be applied to a similar map than the current one (assuming that the case base contains cases applicable to the current map). The behavior contained in the retrieved case then needs to go through the adaptation process. However, our system requires *delayed adaptation* because adaptation is done according to the current game state, and the game state changes with time. Thus it is interesting that adaptation is done with the most up to date game state (ideally with the game state just before the behavior starts execution). For that reason, the behavior in the retrieved case is initially directly sent to the RTEE. Then, when the RTEE is just about to start the execution of a particular behavior, it is sent back to the BG module for adaptation.

The adaptation process consists of a series of rules that are applied to each one of the basic operators of a behavior so that it can be applied in the current game state. Specifically, we have used two adaptation rules in our system:

- *Unit adaptation:* each basic action sends a particular command to a given unit. For instance the first action in the behavior shown in Figure 6 commands the unit “2” to build a “pig-farm”. However, when that case is retrieved and applied to a different map, that particular unit “2” might not correspond to a peon (the unit that can build farms) or might not even exist (the “2” is just an identifier). Thus, the unit adaptation rule finds the most similar unit to the one used in the case for this particular basic action. To perform that search, each unit is characterized by a set of 5 features: owner, type, position (x,y), hit-points, and status (that can be *idle*, *moving*, *attacking*, etc.) and then the most similar unit (according to an Euclidean distance using those 5 features) in the current map to the one specified in the basic action is used.
- *Coordinate adaptation:* some basic actions make reference to some particular coordinates in the map (such as the *move* or *build* commands). To adapt the coordinates, the BG module gets (from the case) how the map in the particular coordinates looks like by retrieving the content of the map in a 5x5 window surrounding the specified coordinates. Then, it looks in the current map for a spot in the map that is the most similar to that 5x5 window, and uses those coordinates. To assess the similarity of two windows, we can simply compare how many of the cells within them are identical. Moreover, we found that assigning more weight to cells closer to the center (e.g. a weight of 32 to the center cell, weight of 3 to its neighbors and weights of 1 to the outermost cells) produced better results.

Once plans are adapted using these two adaptation rules, they are sent back to the RTEE.

8 Experimental Results

To evaluate our approach, we used several variations of a 2-player version of the well known map “Nowhere to run nowhere to hide”, all of them of size 32x32. As explained in Section 3, this map has the characteristic of having a wall of trees that

Table 2. Summary of the results of playing against the built-in AI of WARGUS in several 2-player versions of "Nowhere to run nowhere to hide"

	map1	***map2***	***map3***
trace1	*3 wins*	*3 wins*	*1 win, 1 tie, 1 loss*
trace2	*1 loss, 2 ties*	*2 wins, 1 tie*	*1 tie, 2 losses*
trace1 & trace2	*3 wins*	*3 wins*	*2 wins, 1 tie*

separates the players and that leads to complex strategic reasonings. Specifically, we used 3 different variations of the map (that we will refer as *map1*, *map2* and *map3*), where the initial placement of the buildings (a gold mine, a townhall and a peasant in each side) varies strongly, and also the wall of trees that separates both players is very different in shape (e.g. in one of the maps it has a very thin point that can be tunneled easily).

We recorded expert traces for the first two variants of the map (that we will refer as *trace1* and *trace2*). Specifically, *trace1* was recorded in *map1* and used a strategy consisting on building a series of ballistas to fire over the wall of trees; and *trace2* was recorded in *map2* and tries to build defense towers near the wall of trees so that the enemy cannot chop wood from it. Each trace contains 50 to 60 actions, and about 6 to 8 cases can be extracted from each of them. Moreover, in our current experiments, we have assumed that the expert wins the game, it remains as future work to analyze how much the quality of the expert trace affects the performance of the system.

We tried the effect of playing with different combinations of them in the three variations of the map. For each combination, we allowed our system to play against the built-in AI three times (since WARGUS has some stochastic elements), making a total of 27 games.

Table 2 shows the obtained results when our system plays only extracting cases from *trace1*, then only extracting cases from *trace2*, and finally extracting cases from both. The table shows that the system plays the game at a decent level, managing to win 17 out of the 27 games it played. Moreover, notice that when the system uses several expert traces to draw cases from, its play level increases greatly. This can be seen in the table since from the 9 games the system played using both expert traces, it won 8 of them and never lost a game, tying only once. Moreover, notice also that the system shows adaptive behavior since it was able to win in some maps using a trace recorded in a different map (thanks to the combination of planning, execution, and adaptation).

Finally, we would like to remark the low time required to train our system to play in a particular map (versus the time required to write a handcrafted behavior to play the same map). Specifically, to record a trace an expert has to play a complete game (that takes between 10 and 15 minutes in the maps we used) and then annotate it (to annotate our traces, the expert required about 25 minutes per trace). Therefore, in 35 to 40 minutes of time it is possible to train our architecture to play a set of WARGUS maps similar to the one where the trace was recorded (of the size of the maps we used). In contrast, one of our students required several weeks to hand code a strategy to play WARGUS at the level of play of our system. Moreover, these are preliminary results and we plan to systematically evaluate this issue in future work. Moreover, as we have seen our system is able to combine several traces and select cases from one or the other according to the current situation. Thus, an expert trace for each single map is not needed.

9 Related Work

Recently, the interest for applying artificial intelligence techniques has experienced a notable increase. As a proof of that, we have seen workshops specially dedicated to game AI in recent conferences such as ICCBR 2005 and IJCAI 2005.

Concerning the application of case-based reasoning techniques to computer games, Aha et al. (Aha et al. 2005) developed a case-based plan selection technique that learns how to select an appropriate strategy for each particular situation in the game of WARGUS. In their work, they have a library of previously encoded strategies, and the system learns which one of them is better for each game phase. In addition, they perform an interesting analysis on the complexity of real-time strategy games (focusing on WARGUS in particular). Another application of case based reasoning to real-time strategy games is that of Sharma et al. (Sharma et al. 2007), where they present a hybrid case based reinforcement learning approach able to learn which are the best actions to apply in each situation (from a set of high level actions). The main difference between their work and ours is that they learn a case selection policy, while our system constructs plans from the individual cases it has in the case base. Moreover, our architecture automatically extracts the plans from observing a human rather than having them coded in advance.

Ponsen et al. (Ponsen et al. 2005) developed a hybrid evolutionary and reinforcement learning strategy for automatically generating strategies for the game of WARGUS. In their framework, they construct a set of rules using an evolutionary approach (each rule determines what to do in a set of particular situations). Then they use a reinforcement learning technique called *dynamic scripting* to select a subset of these evolved rules that achieve a good performance when playing the game. There are several differences between their approach and ours. First, they focus on automatically generating strategies while we focus on acquiring them from an expert. Moreover, each of their individual rules could be compared to one of our behaviors, but the difference is that their strategies are combined in a pure reactive way, while our strategies are combined using a planning approach. For our planner to achieve that, we require each individual behavior to be annotated with the goal it pursues.

Hoang et al. (Hoang et al. 2005) propose to use a hierarchical plan representation to encode strategic game AI. In their work, they use HTN planning (inside the framework of Goal-Oriented Action Planning (Orkin 2003)). Further, in (Muñoz et Aha 2004) Muñoz and Aha propose a way to use case based planning to the same HTN framework to deal with strategy games. Moreover, they point out that case based reasoning provides a way to generate explanations on the decisions (i.e. plans) generated by the system.

The HTN framework is related to the work presented in this paper, where we use the task-method decomposition to represent plans. Moreover, in their work they focus on the planning aspects of the problem while in this paper we focus on the learning aspects of the problem, i.e. how to learn from expert demonstrations.

Sánchez-Pelegrín et al. (Sánchez-Pelegrín et al. 2005) also developed a CBR AI module for the strategy game C-evo. They focused on a subproblem of the game (military unit behavior), that also consists of action selection rather than plan construction.

Other work on game AI not related to CBR is that of evolving strategies using reinforcement learning or evolutionary algorithms. For instance, Andrade et al. (Andrade et al. 2005) use a reinforcement learning technique to learn proper difficulty balancing in a real-time fighting game. Their main goal is to ease the effort the developers need to put in adjusting the difficulty level of games. That work shares one goal with ours in the sense that their purpose is to develop tools that can help game developers to include better AI in their games with a reduced amount of effort.

Another work on evolving behaviors is that of Bakkes et al. (Bakkes 2005), where they present TEAM2 as an on-line learning technique to evolve group behaviors. They show that the best-response based TEAM2 is more efficient than the evolutionary algorithm based TEAM in the Quake III game. Moreover, they focus on evolutionary algorithms rather than in planning and case based techniques.

The work presented in this paper is strongly related to existing work in case-based planning (Hammond 1990). Case Based Planning work is based on the idea of planning by remembering instead of planning from scratch. Thus, a case based planner retains the plans it generates to reuse them in the future, uses planning failures as opportunities for learning, and tries to retrieve plans in the past that satisfy as many of the current goals as possible. Specifically, our work focuses on an integrated planning and execution architecture, in which there has been little work in the case based planning community. A sample of such work is that of Freβmann et al. (Freβmann et al. 2005), where they combine CBR with multi-agent systems to automate the configuration and execution of workflows that have to be executed by multiple agents.

Integrating planning and execution has been studied in the search based planning community. For example, CPEF (Myers 1999) is a framework for continuous planning and execution. CPEF shares a common assumption with our work, namely that plans are dynamic artifacts that must evolve with the changing environment in which they are executing changes. However, the main difference is that in our approach we are interested in case based planning processes that are able to deal with the huge complexity of our application domain.

10 Conclusions

In this paper we have presented a case based planning framework for real-time strategy games. The main features of our approach are a) the capability to deal with the vast decision spaces required by RTS games, b) being able to deal with real-time problems by interleaving planning and execution in real-time, and, c) solving the knowledge acquisition problem by automatically extracting behavioral knowledge from annotated expert demonstrations in form of cases. We have evaluated our approach by applying it to the real-time strategy WARGUS with promising results.

The main contributions of this framework are: 1) a case based integrated real-time execution and planning framework; 2) the introduction of a behavior representation language that includes declarative knowledge as well as procedural knowledge to allow both reasoning and execution; 3) the idea of automatic extraction of behaviors from expert traces as a way to automatically extract domain knowledge from an expert; 4) the idea of heterogeneous case bases where cases that contain solutions for several different problems (characterized as *goals* in our framework) coexist and 5) the introduction of *delayed adaptation* to deal with dynamic environments (where adaptation has to be delayed as much as possible to adapt the behaviors with the most up to date information).

As future lines of research we plan to experiment with adding a case retention module in our system that retains automatically all the adapted behaviors that had successful results while playing, and also annotating all the cases in the case base with their rate of success and failure allowing the system to learn from experience. Additionally, we would like to systematically explore the transfer learning (Sharma et al. 2007) capabilities of our approach by evaluating how the knowledge learnt (either from expert traces or by experience) in a set of maps can be applied to a different set of maps. We also plan to further explore the effect of adding more expert traces to the system and evaluate if the system is able to properly extract knowledge from each of them to deal with new scenarios.

Further, we would like to improve our current planning engine so that, in addition to sequential and parallel plans, it can also handle conditional plans. Specifically, one of the main challenges of this approach will be to detect and properly extract conditional behaviors from expert demonstrations.

Acknowledgements. The authors would like to thank Kane Bonnette, for the java-WARGUS interface; and DARPA for their funding under the Integrated Learning program TT0687481.

References

Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7(1), 39–59 (1994)

Aha, D., Molineaux, M., Ponsen, M.: Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 5–20. Springer, Heidelberg (2005)

Allen, F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26(11), 832–843 (1983)

Andrade, G., et al.: Extending Reinforcement Learning to Provide Dynamic Game Balancing. In: IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, pp. 7–12 (2005)

Bakkes, S., Spronk, P., Postma, E.: Best-Response Learning of Team Behavior in Quake III. In: IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, pp. 13–18 (2005)

Buro, M.: Real-time Strategy Games: A New {AI} Research Challenge. In: Proceedings of IJCAI 2003, pp. 1534–1535. Morgan Kaufmann, San Francisco (2003)

Chandrasekaran, B.: Design problem solving: a task analysis. Artificial Intelligence Magazine 11(4), 59–71 (1990)

Fikes, R., Nilsson, N.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2(3/4), 189–208 (1971)

Freβmann, A., et al.: CBR- based Execution and Planning Support for Collaborative Workflows. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 271–280. Springer, Heidelberg (2005)

Hammond, K.: Case Based Planning: A Framework for Planning from Experience. Cognitive Science 14(3), 385–443 (1990)

Hoang, M., Lee-Urban, S., Muñoz-Avila, H.: Hierarchical Plan Representations for Encoding Strategic Game AI. Proceedings of AIIDE 2005, 63–68 (2005)

Kolodner, J.: Case-based Reasoning. Morgan Kaufmann, San Francisco (1993)

Mateas, M., Stern, A.: A behavior language for story-based believable agents. IEEE intelligent systems and their applications 17(4), 39–47 (2002)

Muñoz-Avila, H., Aha, D.: On the Role of Explanation for Hierarchical Case-Based Planning in Real-Time Strategy Games. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, Springer, Heidelberg (2004)

Myers, K.: CPEF: A Continuous Planning and Execution Framework. Artificial Intelligence Magazine 20(4), 63–69 (1999)

Nau, D., et al.: SHOP: Simple Hierarchical Ordered Planner. Proceedings of IJCAI 1999, 968–975 (1999)

Orkin, J.: Applying Goal-Oriented Action Planning to Games. In: AI Game Programming Wisdom II, pp. 217–227. Charles River Media (2003)

Ponsen, M., et al.: Automatically acquiring adaptive real-time strategy game opponents using evolutionary learning. In: Proceedings of the 20th National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence, pp. 1535–1540 (2005)

Sánchez-Pelegrín, R., Gómez-Martín, M.A., Díaz-Agudo, B.: A CBR Module for a Strategy Videgame. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 217–226. Springer, Heidelberg (2005)

Sharma, M., et al.: Transfer Learning in Real Time Strategy Games Using Hybrid CBR/RL. In: Proceedings if IJCAI 2007, pp. 1041–1046. Morgan Kaufmann, San Francisco (2007)

Spalazzi, L.: A Survey on Case-Based Planning. Artificial Intelligence Review 16(1), 3–36 (2001)

A CBR System: The Core of an Ambient Intelligence Health Care Application

Juan M. Corchado, Javier Bajo, and Yanira de Paz

Departamento Informática y Automática
Universidad de Salamanca
Plaza de la Merced s/n, 37008, Salamanca, Spain
corchado@uosal.es, jbajope@usal.es, yanira@usal.es

1 Introduction

This paper presents a case-based reasoning system developed to generate an efficient and proactive ambient intelligent application. Ambient Intelligence (AmI) proposes a new way to interact between people and technology, where this last one is adapted to individuals and their context (Friedewald and Da Costa 2003). The objective of Ambient Intelligence is to develop intelligent and intuitive systems and interfaces capable to recognize and respond to the user's necessities in a ubiquitous way, providing capabilities for ubiquitous computation and communication, considering people in the centre of the development, and creating technologically complex environments in medical, domestic, academic, etc. fields (Susperregui *et al.* 2004). Ambient Intelligence requires new ways for developing intelligent and intuitive systems and interfaces, capable to recognize and respond to the user's necessities in a ubiquitous way, providing capabilities for ubiquitous computation and communication. The multi-agent systems (Wooldridge and Jennings 1995) have become increasingly relevant for developing distributed and dynamic intelligent environments. A case-based reasoning system (Aamodt and Plaza 1994) has been embedded within a deliberative agent and allows it to respond to events, to take the initiative according to its goals, to communicate with other agents, to interact with users, and to make use of past experiences to find the best plans to achieve goals. The deliberative agent works with the concepts of Belief, Desire, Intention (BDI) (Bratman 1987), and has learning and adaptation capabilities, which facilitates its work in dynamic environment.

With the appearance of AmI-based systems, one of the most benefited segments of population will be the elderly and people with disabilities. It will improve important aspects of their life, especially health care (Emiliani and Stephanidis 2005). There is an ever growing need to supply constant care and support to the disabled and elderly (Nealon and Moreno 2003) and the drive to find more effective ways to provide such care has become a major challenge for the scientific community. Today, the number of Europeans over 60 years represents more than 25% of the population and it is estimated that in 20 years this percentage will rise to one third of the population (Camarinha-Matos and Afsarmanesh 2002). In the United States of America it is expected that in 2020 people over 60 will represent about 1 of 6 citizens (Kohn *et al.* 1999). Furthermore, over 20% of people over 85 years have a limited capacity for independent living, requiring continuous monitoring and daily care. The Institute of Medicine has studied the role of information technology in improving health care delivery in the US. In (Kohn *et al.* 1999), the Institute presents a strategy and an action plan to foster innovation and improve the delivery of care. The need to reinvest in the system is underlined and as such six health care aims are defined; to be safe, effective, patient-centered, timely, efficient and equitable. Ten guidelines for the redesign of the system are given attention on the role of the patient and improvements in knowledge, communication and safety mechanisms. Moreover, the Institute proposes a strategy to improve safety in health care based on the study of medical errors. The proposed system presented here has been conceived and developed taking these considerations into account.

The importance of developing new and more reliable ways to provide care and support to the elderly is underlined by this trend (Camarinha-Matos and Afsarmanesh 2002), and the creation of secure, unobtrusive and adaptable environments for monitoring and optimizing health care will become vital. Some authors (Nealon and Moreno 2003) consider that tomorrow health care institutions will be equipped with intelligent systems capable of interacting with humans. Multi-agent systems and architectures based on intelligence devices have recently been explored as supervision systems for medical care for the elderly or Alzheimer patients (Corchado *et al.* 2007). These intelligence systems aim to support them in all aspects of their daily life, predicting potential hazardous situations and delivering physical and cognitive support.

Radio Frequency IDentification (RFID) (Sokymat 2006) is an automated data-capture technology that can be used to electronically identify, track, and store information about products, items, components or people. It is most frequently used in industrial/manufacturing, transportation, distribution, and warehousing industries, however, there are other growth sectors including health care. The proposed system uses microchips mounted on bracelets worn on the patient's wrist or ankle, and sensors installed over protected zones, with an adjustable capture range up to 2 meters. The microchips or transponders help locate the patients, which can be ascertained by consulting the CBR agents installed in personnel PDAs.

The proposed system has been developed for facilitating the management and control of geriatric residences. The aim of this paper is to present the CBR based agent and to demonstrate how its planning mechanism improves the medical assistance in geriatric residences. These agents also facilitate the nurses' and doctors' work by providing updated information about patients and emergencies, as well as historical data. We believe that our system can easily be adapted to work in many other similar industrial environments.

This chapter is organized as follows: the next section describes the need for new Ambient Intelligence solutions and the advantages of using agents and case-based planning mechanisms for the development of intelligent environments. Section 3 describes the proposed CBP mechanism in detail. In Section 4 the mechanism proposed in Section 3 is applied to a geriatric residence. Finally, Section 5 presents the results and conclusions obtained in this research.

2 CBR-Based Architecture for AmI Solutions

Case-based Reasoning (CBR) is a type of reasoning based on the use of past experiences to solve new problems (Aamodt and Plaza 1994). The purpose of a CBR system is to solve new problems by adapting solutions that have been used to solve similar problems in the past. A case can be defined as a past experience, and is composed of three elements: A problem description which describes the initial problem, a solution which provides the sequence of actions carried out in order to solve the problem, and the final state which describes the state achieved once the solution was applied.

A CBR system manages cases (past experiences) to solve new problems. The way in which cases are managed is known as the CBR cycle. The CBR cycle shown in Figure 1 consists of four sequential phases: retrieve, reuse, revise and retain. The retrieve phase starts when a new problem description is received. Similarity algorithms are applied in order to retrieve from the cases memory the cases with a problem description more similar to the current one. Once the most similar cases have been retrieved, the reuse phase begins. In this phase the solutions of the cases retrieved are adapted to obtain the best solution for the current case. The revise phase consists of an expert revision of the solution proposed. Finally, the retain phase allows the system to learn from the experiences obtained in the three previous phases and updates the memory case in consequence.

AmI proposes new possibilities for solving a wide scope of problems and a new interaction way between people and technology, where this last one is adapted to individuals and their context, showing a vision where people are surrounded by intelligent interfaces merged in daily life objects (Emiliani and Stephanidis 2005). This

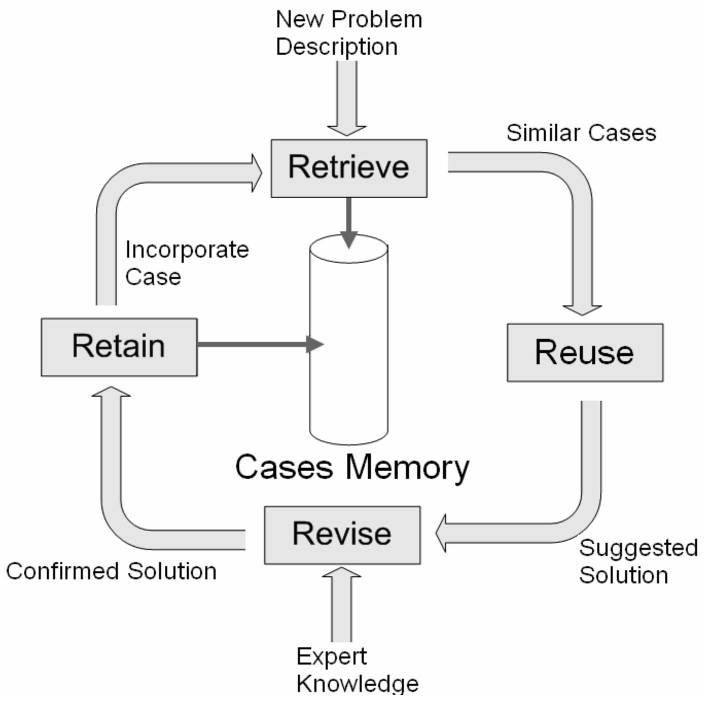


Fig. 1. Diagram including a CBR-BDI agent reasoning cycle

adaptation requires creating computing-capable environments with intelligent processing and communication services. One of the possibilities for executing complex computational tasks is the use of CBR systems, which are capable of learning from their initial knowledge and autonomously adapting themselves to environmental changes. As CBR systems use their experiences from previous similar situations to resolve a new problem, it will be possible to personalize a user access if we take into account similar past cases. These similar past cases can contain memories about users with similar profiles trying to resolve a problem in similar context conditions.

The CBR adaptation and learning mechanism needs updated information from the environment and communication paths, with both the system users and the system elements. The multiagent technology (Wooldridge and Jennings 1995) can be used for solving distributed systems in which CBR systems may act. In this paper it is proposed that the use of agents reason and solve problems using embedded CBR systems (Corchado and Laza 2003). The agents and multi-agent systems (MAS) have become increasingly relevant for developing distributed and dynamic intelligent environments (Corchado *et al.* 2007). One of the advantages of the agents is their adaptability to work in mobile devices, so they support wireless communication (Wi-Fi, Bluetooth,WiMAX, UMTS, etc.) which facilitates the portability to a wide range of mobile devices. This advantage makes the agents and multiagent systems appropriate to be applied to the development of ubiquitous and mobile environments and to obtain context information. Agents are computational entities that can be characterized through their capacities in areas such as autonomy, reactivity, pro-activity, social abilities, reasoning, learning and mobility (Wooldridge and Jennings 1995). These capacities make the multi-agent systems appropriate for constructing intelligent environments. An agent can act as an interface between the user and the rest of the elements of the intelligent environment. Furthermore, given the adaptability of agents to mobile devices (with low memory and processing resources), it is possible to provide ubiquitous and transparent interactions, even personalizing the user access. If we can provide an agent with the appropriate reasoning mechanism, for example a CBR system, it could be able to adapt itself to environmental changes or make predictions based on previous knowledge or experience (Corchado and Laza 2003). In this sense an agent which integrates a CBR system will be context-sensitive and will take decisions allowing it to automatically adapt itself to the changes on its surroundings.

Certain interface agents incorporate intelligent mechanisms to interact with the users of the multiagent system. Moreover, an agent can integrate certain middleware that allows it to interact with different context sensitive technologies, such as radio frequency identification (RFID) (Sokymat 2006), automation devices (ZigBee) (ZigBee Standards Organization 2006), etc. This last agent type will be able to obtain information of its surroundings in execution time, as well as to respond to the environmental changes. Then, if a CBR-based intelligent agent could be able to exchange information with the agents specialized in obtaining context sensitive information, and interact with interface agents, it could be possible to construct intelligent environments allowing personalized services. Figure 2 shows a scheme of such CBR-based intelligent environment. In Figure 2 it is possible to observe how users can gain access to the system by means of their personal agent installed in mobile devices or personal computers. Besides, it is possible to observe how different technologies integrate within the multiagent system in order to achieve the objectives of the intelligent

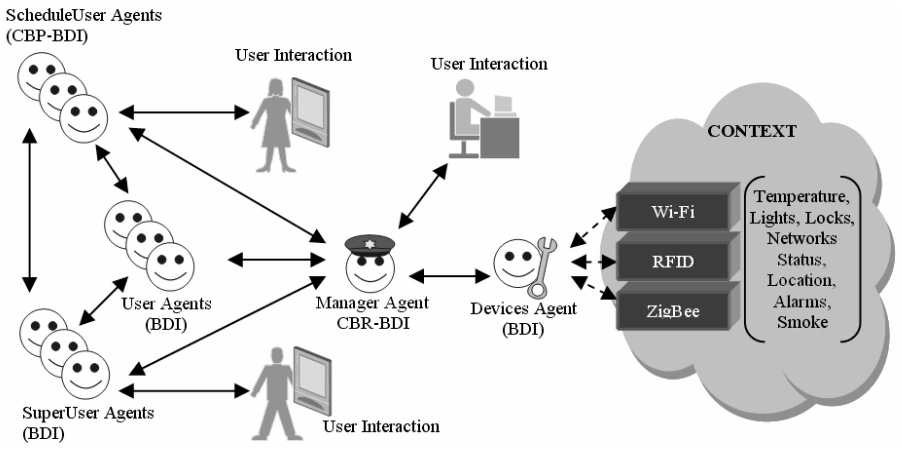


Fig. 2. Architecture scheme for intelligent environments construction

environment. The core of the system is the CBR agent, which provides reasoning and planning mechanisms.

3 CBR Planning Mechanism

There are many industrial environments requiring dynamic planning systems that allow them to respond to changes in the environment and to provide efficient plans in execution time, for example, optimising the working rotas. A CBP planning mechanism is a type of CBR system (Bajo *et al.* 2007) specialized on plan generation. The CBP presented within this work calculates the *most re-plan-able intention (MRPI),* which is the plan than can be easily substituted by others in case the initial plan gets interrupted. In a dynamic environment, it is important to have an alternative plan to maintain the efficiency of the system. The MRPI avoids coming back to the initial plan. It uses the concept of a geodesic plan: *"plan of minimum risk"* or the plan with a higher density function of plans around it. In an environment that changes unpredictably, any plan that is distal to the geodesic plan means that a certain risk is accepted. The CBP mechanism follows the 4 stages of a CBR system (Retrieval, Reuse, Review and Retain). During each reasoning stage the CBP performs the following actions:

Retrieval stage. During this step the CBP mechanism selects the plans from the plan-base, which are similar or contain similar information to the problem case that describes the aims of a given user and his/her restrictions and user profile. For the recovery of similar cases to the current problem, the multivariate technical of conglomerates analysis was used, that is the hierarchical method. The reason that the method of conglomerates analysis was chosen, is because the recovered cases will be more similar to the current problem than with other methods. Since it is an analysis type in which the variables are considered from a vectorial point and the comparisons are made keeping in mind the components of that vector. If there are two cases that have all the same components except one, this method no longer considers them the same. The general idea consists of distributing n data, of dimension p, in "conglomerates" or groups (clusters) formed by data that are "similar to each other". To define the conglomerates, it is necessary to choose distances and measures of previous similarity. The usual distances are the euclidean and the distance of Mahalanobis. The algorithm of grouping hierarchical begins with n conglomerates (one for each individual observation). In successive steps the conglomerates unite for likeness. The process of the algorithm can be graphically represented in a dendrogram. The number of conglomerates can decide the starting from the dendrogram, stopping the process of mass at the moment that the application of the algorithm takes to unite conglomerates that are very distant (Martin 2001).

Reuse stage. During this stage the CBP mechanism creates a vectorial hyper dimensional space that includes all the problem restrictions (time, food, equipment, rooms, etc.) using a B-splines technique. Then all possible plans that satisfy a given user requirement are identified. Such plans are the geodesic curves (of the vectorial hyper dimensional space), which can be calculated using the Euler Theorem (Bellman 1957). Such geodesic plans guaranty minimum risk and constant efficiency, while satisfying the user objectives. The geodesic plan with the Minimum Jacobi Field will be the *most re-plan-able intention.* The Minimum Jacobi Field identifies the plan with more density of the other plans around it, and therefore, if the plan gets interrupted, a closer plan will be easily identified to carry out the user objectives. The CBP system recommends their users to use the *most re-plan-able intentions*.

Review stage: During the revision stage the user confirms the plan, indicating his/her satisfaction degree.

Retain stage: During the retain stage the plans are stored depending on their efficiency.

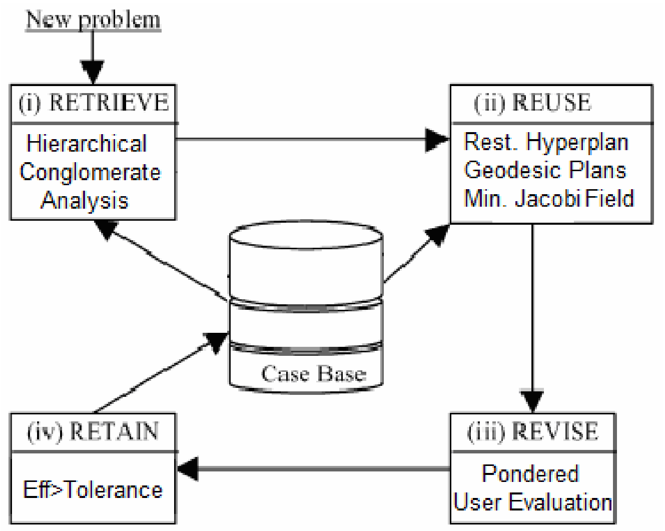


Fig. 3. Case-based planning mechanism for dynamic environments

In Figure 3 it is possible to observe the steps carried out in each of the stages of the CBP system. The reuse stage implements the MRPI replaning mechanism (Bajo *et al.* 2007). When an interruption occurs, the system initiates a new CBP cycle, taking into account the tasks previously accomplished. That is, in the new retrieval stage, plans with a similar problem description to the current situation (after the interruption) will be recovered. The MRPI guarantees that at least some of the plans closest to the initial geodesic plan will be recovered (the rest of the plans are not valid anymore because of the restrictions, that the tasks have already accomplished, etc.) together with new plans. The proposed CBP mechanism has been tested in a real environment as shown in the next section.

4 Case Study: Geriatric Residences

The CBP planning mechanism can be very useful to facilitate planning or scheduling tasks in dynamic industrial environments. In this work we have focused on geriatric residences because they are one of the objectives of Ambient Intelligence (Friedewald and Da Costa 2003). The system prototype can easily be implemented on diverse dynamic scenarios, with some changes according to the users and project necessities. The CBP presented in this work focuses on scheduling the nurses working day and takes into account nurses' profiles, tasks that must be completed, available time and resources. The CBP needs to interact with its surroundings: patients, nurses, doctors, management personnel and intelligent devices.

4.1 System Overview

A real scenario that has been studied, the Alzheimer Santísima Trinidad Residence of Salamanca, has collaborated in the development of the intelligence environment presented here, providing their know-how and experimenting with the prototype developed. This residence is intended for people over 65 years, and has the following services and facilities available among others: TV room, geriatric bathroom, hairdressing salon, medical service, religious attention, occupational therapy, technical assistance, terrace, garden, laundry service, clothes adjustment, infirmary, reading room, living room, visitor room , cafeteria, social worker, chapel, elevator, customized diet, and multipurpose room. The residence has been equipped with RFID and Wi-Fi technologies as can be seen in Figure 4. 42 ID door readers (Hitag HT RM401and mobile WorkAbout Pro RFID) have been installed, one on each door and elevator, 4 controllers, one at each exit, one on the first floor hall and another on the second floor hall, and 36 bracelets (Sokymat ID Band Unique Q5 with a chip Hitag S 256) (Sokymat 2006), one for each patient and the nurses, shown in Figure 4. The ID door readers get the ID number from the bracelets and send the data to the controllers which send a notification to the Manager agent.

The intelligence environment has been modelled by means of four different types of agent, as can be seen in Figure 4:

- Patient agent manages the patient's personal data and behaviour (monitoring, location, daily tasks, and anomalies). Every hour it validates the patient's location, monitors the patient's state and sends a copy of its memory base (patient's state, goals and plans) to the manager agent in order to maintain backups. The patient's state is instantiated at execution time as a set of beliefs and these beliefs are controlled through goals that must be achieved or maintained. The beliefs that were seen to define a general patient's state at the Santísima Trinidad Residence of Salamanca were: weight, temperature, blood pressure, feeding (diet characteristics and next time to eat), oral medication, parental medication, posture change, toilet use, personal hygiene, and exercise. The beliefs and goals used for every patient depend on the plan (treatment) or plans that the doctors prescribe. The patient's agent monitors the patient's state by means of the goals. To know if a goal has been achieved or has failed, it is necessary to maintain continuous communication with the rest of the agents. At least once per day, depending on the corresponding treatment, the patient agent must contact the nurse agent. The patient's agent must have periodic communication with the doctor agent. Finally the patient agent must ensure that all the actions indicated in the treatment are fulfilled.

- The manager agent plays two roles, the security role that controls the patients' location and manages locks and alarms; and the Manager role that manages the medical record database and the doctor-patient and nurse-patient assignment. It must provide security for the patients and medical staff and the patients, doctors and nurse assignment must be efficient. The Manager agent controls all the tasks that must be accomplished in the residence. These tasks are assigned to the Nurse and Doctor agents. The Manager agent integrates a CBR mechanism to assign tasks to each nurse or doctor. The agent uses the experience obtained in previous assignments together with the nurses and doctor profiles and the time restrictions in order to optimize the tasks distribution.
- The doctor agent treats patients. The Doctor agent needs to interact with the Patient agent to order a treatment and receive periodic reports, with the Manager agent to consult medical records and assigned patients, and with the Nurse agent to ascertain the patient evolution.
- The nurse agent schedules the nurse's working day obtaining dynamic plans depending on the tasks needed for each assigned patient. Nurse agent manages nurses' profiles, tasks, available, time and resources. The generated plans must guarantee that all the patients assigned to the nurse are given care. The nurse can't exceed 8 working hours. Every agent generates personalized plans depending on the nurse's profile and working habits. The Nurse agent integrates a CBP planning mechanism to schedule the nurse's working day. Every day the Nurse agent receives the tasks assigned by the Manager agent. The Nurse agent uses the CBP mechanism to schedule the tasks and provides the nurse with a plan. Moreover, if any kind of interruption occurs during the plan execution, the Nurse agent uses its replanning capability to resolve the problem.

Manager and Patient agents run in a central computer, but the Nurse agents run on mobile devices, so a robust wireless network has been installed as an extension to the existing wired LAN.

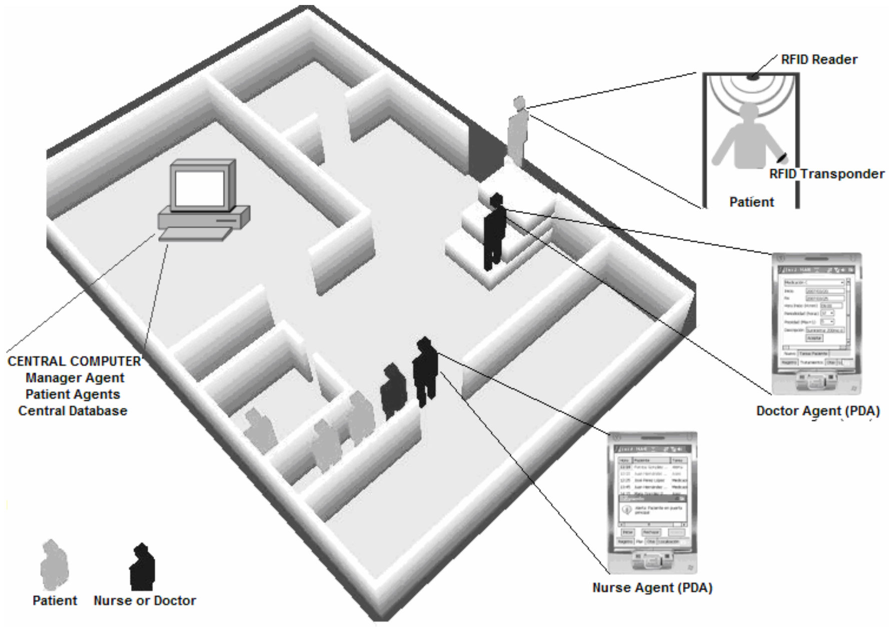


Fig. 4. Wireless technology organization schema

4.2 Embedding a CBR System within a Deliberative Agent

The CBR planning mechanism presented in Section 3 has been used to model the scheduling of the nurses working times dynamically in a geriatric residence (Corchado *et al.* 2007). A large quantity of measurements have been taken in order to standardise the time taken to arrive at a given room, or to take a patient from one room to another (depending on the level of dependence of the patient). These times are included directly in the time assigned for each task. The CBP Nurse agent constructs plans as a sequence of tasks that need to be carried out by a nurse.

The first step when working with CBRs is obtaining a definition of the case structure. A case structure is composed of three elements: a problem description, the solution applied to solve the problem and the result obtained for the solutions applied (Aamodt and Plaza 1994). For the current problem the case is described as follows:

- Problem description: Describes the initial information available for generating a plan. As can be seen in Table 1 the problem description for planning a nurse's working day consists of a case identification, the list of tasks that the nurse has to accomplish, the nurse's profile, the nurse's preferences and temporal restrictions.

  Where the structure of a Task can be observed in Table 6, and the structure corresponding to a nurse profile can be observed in Table 5.

Table 1. Problem description for a planning a nurse working day

Problem Description Field	Field Type
CaseId	Integer
Tasks	ArrayList of Tasks
Nurse Profile	NurseProfile
Nurse Preferences	NursePreferences
Restrictions	ArrayList of Restriction

- Solution: Describes the actions carried out in order to solve the problem description. As can be seen in Table 2 it contains the case identification, the timetables and the priorities for each task and the route or sequence of tasks.

Table 2. Solution (Plan) proposed to the nurse

Solution Field	Field Type
Case d	Integer
TimeTables	ArrayList of TimeTable
Route	Route

Where the TimeTables field contains a list of Timetable elements. A Timetable element, as shown in Table 3, indicates the identification of a task, the priority of a task, the desired starting and finishing times for a task and the room identification if required.

Table 3. Timetable for a task

TimeTable	Field Type
Priority	Float
StartTime	Time
FinishTime	Time
Task	Float
Room	Float

The Route field contains a description of the activities carried out in a room of the residence, as shown in Table 4.

Table 4. Route describing the activities executed by a nurse in a given room

Route Fields	Field Type
room	Room
ArrivalTime	Time
ServiceTime	Time
patient	Patient
Tasks	ArrayList of Task
NextRoom	Route

- Efficiency: contains the case id and a number indicating the plan efficiency

For each of the rooms in the geriatric residence, the system manages information related to the room location (in coordinates), access (doors) location (in coordinates), room type and resources available.

In our case study, at the beginning of each working day, every nurse receives a list of tasks that must be completed. The tasks must be assigned to the nurses. It is necessary to take into account that each nurse has a different profile according to their qualification and the tasks that they usually carry out. It is considered appropriate to manage the profiles of the nurses because there are some nurses who perform tasks with greater skill or who carry out tasks faster. The structure of a nurse profile can be observed in Table 5. Table 5 shows the abilities of a nurse, her service time and timetable preferences and restrictions.

Table 5. NurseProfile

NurseProfile Fields	Field Type
Skills	ArrayList of Skill
ServiceTime	Time
Timetable preferences	Date
TImetable restrictions	Date

One of the Manager agent's duties is the assignation of nurses to patients. This assignation is carried out through a CBR reasoning motor integrates within the Manager Agent. When a new assignation of tasks needs to be carried out to the nurses, both past experiences, such as the nurse profile, and the needs of the current situation are retrieved. In this way tasks are allocated to a nurse. These tasks may correspond to the same patient or to a number of patients. The task structure is shown in Table 6. Moreover, as mentioned above, the profile of each nurse is taken into account. For example, not all nurses are equally qualified for rehabilitation. If one nurse is more qualified in the area, she will be allocated the patients whose need for rehabilitation are greater, always taking into account that the nurse cannot work more than 8 hours, so that the number of patients assigned depends on the time needed to carry out the rehabilitation. The Manager agent takes into account how those patients who receive rehabilitation are improving, the arrival of new patients, holiday rotas etc. As such, the allocation of tasks needs to be set on a daily basis.

Table 6. Task structure

Task Field	Field Type
taskData	TaskData
ServiceTime	Time
Date	Date
Room	Room
Resources	ArrayList of Resource

Once the assignation of tasks to a nurse has been completed, the assignation is communicated to the corresponding Nurse agent. That is, the Nurse agent receives a new problem description needing to be resolved. From this moment on, the CBP planning process begins. The Nurse agent must take into account the time that the nurse has available and the time required for each task. Moreover, the resources available and the location of the patients involved are also taken into account. Then the stages of the CBP cycle are executed:

- Retrieval: In the retrieval stage, the descriptions of similar problems are recovered. In order to do this, the Nurse agent allows the application of the hierarchical conglomerate analysis algorithm (Martin 2001). In this step, those problem descriptions found within a range of similarity close to the original problem description are recovered from the cases base. Every retrieved case "k" related to a task "i", is represented by means of a vector X_{ki}.

  $$X_{ki} = \left(x_{ki1}, x_{ki2}, x_{ki3}, x_{ki4}, x_{ki5}, x_{ki6}, x_{ki7}, x_{ki8}, x_{ki9}\right)$$

  Where x_{ki1} indicates the priority for task i, x_{ki2} indicates the patient location, x_{ki3} indicates the start time, x_{ki4} indicates the finish time, x_{ki5} indicates the resources required, x_{ki6} indicates the service rate, x_{ki7} indicates the nurse assigned, x_{ki8} indicates the plan executed and x_{ki8} indicates the nurse qualification.

  Every task "i" corresponding to the new problem "j" is represented by jeans of the vector Y_{ji}

  $$Y_{ji} = (x_{ji1}, x_{ji2}, x_{ji3}, x_{ji4}, x_{ji5}, x_{ji6})$$

  Where x_{ji1} indicates the priority for task i, x_{ji2} indicates the patient location, x_{ji3} indicates de start time for task i, x_{ji4} the finish time for task i, x_{ji5} the resources required to accomplish the task and x_{ji6} the nurse qualification required.

  In order to find those cases containing the task "i" which are similar to the current problem, the euclidean distance is applied. The first five vector components of each case are significant, so they are taken into account. In the first step, the closest case to the current problem is grouped. In the next step, those cases grouped in the first step are considered as a new group, the distances (similarities) among groups are recalculated for the new configuration and the two closest groups are put together. This process is repeated successively until only one group is obtained. The final result is a dendogram, a typical graph used to represent clustering analysis.
- Reuse: Once the most similar problem descriptions have been recovered, the reuse stage recovers the solutions (plans) associated with them. One solution contains all the plans (sequences of tasks) that were carried out in order to achieve the objectives of the Nurse agent for a problem description (assuming that replanning is possible) in the past, as well as the efficiency of the

solution being supplied. The recovered solutions are combined, as explained in (Corchado *et al.* 2007), to construct a plan. At this time the Nurse agent takes control of the processing of the plan (scheduling). The re-planning mechanism is centred around the objectives and resources needed by each task, as well as on the objectives that the nurse needs to perform and the resources available in order to carry out the global plan. The objectives or global plans that each nurse has, are to attend to the patients and not to work for over eight hours. The time available is a problem restriction. This available time will influence the hyper plan of restrictions, specifically, the range of positive values that the z axis takes from this hyper plan. The resources necessary for some of the tasks are food, equipment and rooms. Finally, the Nurse agent takes care of incidents and interruptions that may occur during replanning.

A plan can be interrupted for different reasons. Those which have been taken into account within the residence are: a resource failure, a patient suffers some sort of crisis and requires unforeseen attention, the patient has an unexpected visit or that visits to the patient have exceeded the permitted time or an emergency situation. If the planner finds itself in a situation where the plan is interrupted, it rejects the initial plan and seeks an alternative one. The first thing that needs to change is the task order, attempting to maintain the assignation originally allocated by the Manager agent. The new plan must meet the initial objectives. In the event that this is impossible, the nurses will need to be reassigned. This reassignment will attempt to limit changes to a minimum. For reassignment it is necessary to take into account the tasks that were assigned to the nurses, the development of the plans (which tasks have been carried out and which still need to be done) and the profiles of the nurses (prioritising preparation for the task that cannot be covered). The nurse who is assigned the task should replan in order to include the new task. In the event that the replanning is positive (the tasks that still need to be done and the new task can be carried out) the process is complete. If the replanning is negative, the next nurse down in the ranking will be used.

- Review: The nurse evaluates the plan
- Retain: Lastly, depending on the efficiency of the plan, it will be stored together with its level of efficiency within the beliefs base.

5 Results and Conclusions

The CBP planning mechanism which allows replanning in execution time, integrated within an intelligent agent, has been tested over the last few months. To test the system 30 patient agents, 10 nurse agents, 2 doctor agents and 1 manager agent were instantiated. During the testing period the systems usefulness has been evaluated from different points of view. The system, which in turn improves patient care, facilitates the more flexible assignation of the working shifts at the residence; since the workers have reduced the time spent on routine tasks and can assign this time to extra activities, such as exercising the patients, learning, carrying out leisure activities or just talking with the patients or with their families. Their work is automatically monitored, as well as the patients' activities. The stored information may be analysed with knowledge discovery techniques and may help to improve the quality of life for the patients and the efficiency of the centre. The security of the centre has also been improved in three ways: the system monitors the patients and guarantees that each one of them is in the right place; secondly, only authorised personnel can gain access to the residence protected areas, and thirdly, the information is stored in a more secure way using redundancy and generating continuous backups. The access to information has been protected in order to guarantee confidentiality.

The agent technology provides a framework for distributed problem solving and data communication. The characteristics of agents, such as autonomy, reactivity, proactivity, social abilities, reasoning, learning and mobility (Wooldridge and Jennings 1995) are adequate to fulfil the AmI necessities. Moreover, the incorporation of case-based reasoning mechanisms (Corchado and Laza 2003; Bajo *et al.* 2007) facilitates ubiquitous computation capabilities. An agent can act as an interface between the user and the environment. In this way, it is possible to get that the services provided by the multiagent system and the technology in the environment can be adapted to the user necessities. Finally, agents can be executed on mobile devices, which facilitate ubiquitous capabilities. These characteristics make the proposed architecture appropriate for developing AmI (Friedewald and Da Costa 2003) scenarios. The manager agent assigns tasks and monitors the patients and nurses locations. Nurse and Doctor agents run on handheld devices and provide easy-to-use interfaces.

The planning mechanism has been evaluated taking into account the characteristics of a CBR system (Aamodt and Plaza 1994). In this sense, the parameters that influence the efficiency of each of the stages of the proposed CBP cycle have been studied in detail. In Figure 5 it is possible to see the interaction between the variables

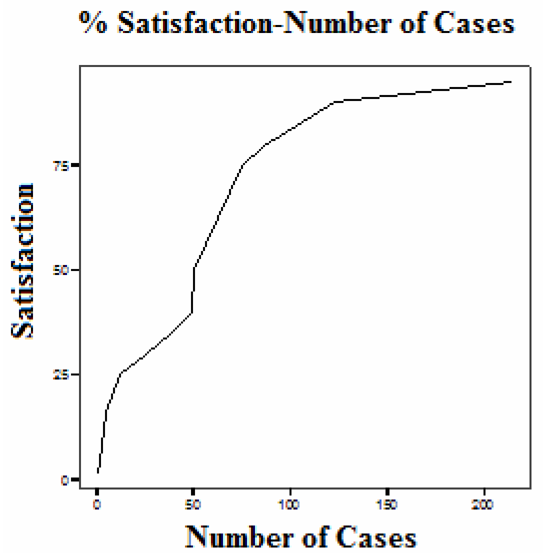


Fig. 5. % Average satisfaction degree related to the average number of retrieved cases

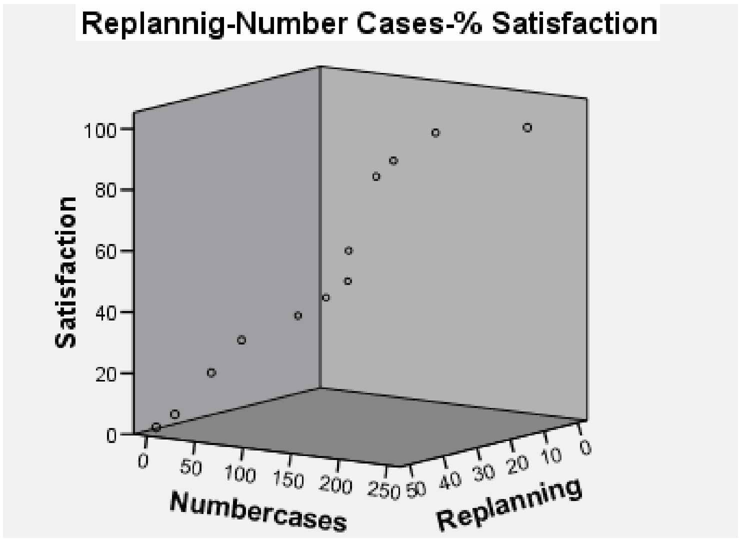


Fig. 6. Interaction between average number of cases retrieved, average number of replannings required per plan and the nurse satisfaction degree

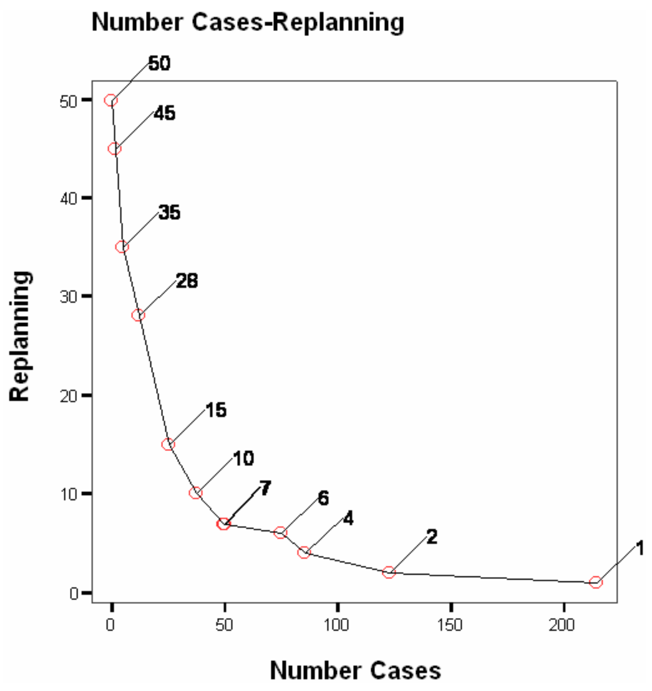


Fig. 7. Number of replannings related to cases retrieved

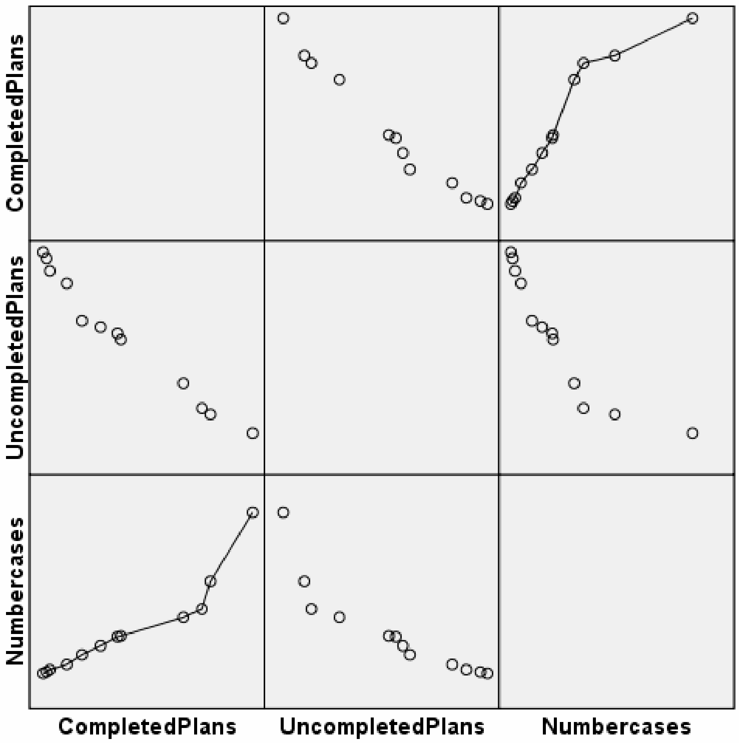


Fig. 8. Plans completed and plans failed related to the average number of cases retrieved

corresponding to the average satisfaction degree for a nurse related to the plan success and the number retrieved from the base case in order to provide the plan. As explained in Section 2, one of the key concepts in the efficiency of a CBR system consists of the retrieval algorithm implemented, and specifically, in the similitude metric applied. The success or efficiency of a plan can be calculated through the results obtained for each of the tasks of the plan and the evaluation provided by the nurse that has completed the plan. On the other hand, the number of retrieved cases allows evaluating the strategy proposed in the retrieval stage. The results presented in Figure 5 show how for a larger number of retrieved cases from the cases memory, the satisfaction degree indicated by nurses increases. That is, the nurses can better achieve their initial objectives when the number of similar past experiences available is larger. The relationship between the number of retrieved cases and the nurse satisfaction degree is stabilized for a number of cases around 110 and 200.

The case-based planning mechanism proposed in the frame of this research comes characterized by its capability for dynamic replanning in execution time. In order to obtain a better understanding of the success of the CBP planning system, it is of interest to take into account the average number or replanning actions executed for a plan. Figure 6 shows the interaction between the number of retrieved cases, the average satisfaction degree (percentage) and the average number of replans per plan in the proposed CBP system. In Figure 6 it can be observed how the number of retrieved cases increases, the satisfaction degree grows and the average number of replannings required per plan decreases.

In Figure 7 it has been represented in the interaction that exists between the average number of cases retrieved in order to solve a plan and the average number of replannings required. It is possible to detect an inverse relationship between both variables, that is, when the average number of retrieved cases increases the average number of replannings required decrease and vice versa. Moreover, the graph presents an inflection point of 50 retrieved cases, corresponding to an average of 7 replannings. This fact let's us conclude that from 50 retrieved cases on, the number of replannings decreases smoothly. It is necessary to notice that a CBR system needs initial knowledge and a certain learning period to provide good results.

Another good indicator to evaluate the CBP is the success of the plans suggested. The plans recommended to the nurses can be successfully completed (replanning can be necessary or not) or failed. In the same way as the average number of replannings and the nurse satisfaction degree was influenced by the number of cases retrieved. The number of plans successfully completed depends on the retrieval strategy implemented. Figure 8 presents a 3x3 matrix representing the interaction between the

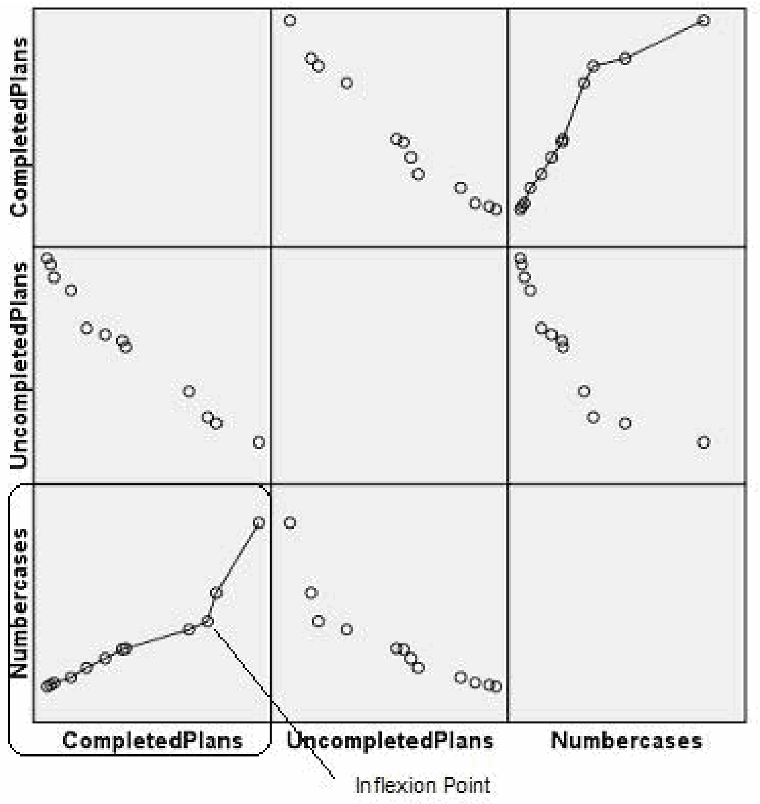


Fig. 9. Inflection point in the interaction between the average number of completed plans and the average number of retrieved cases

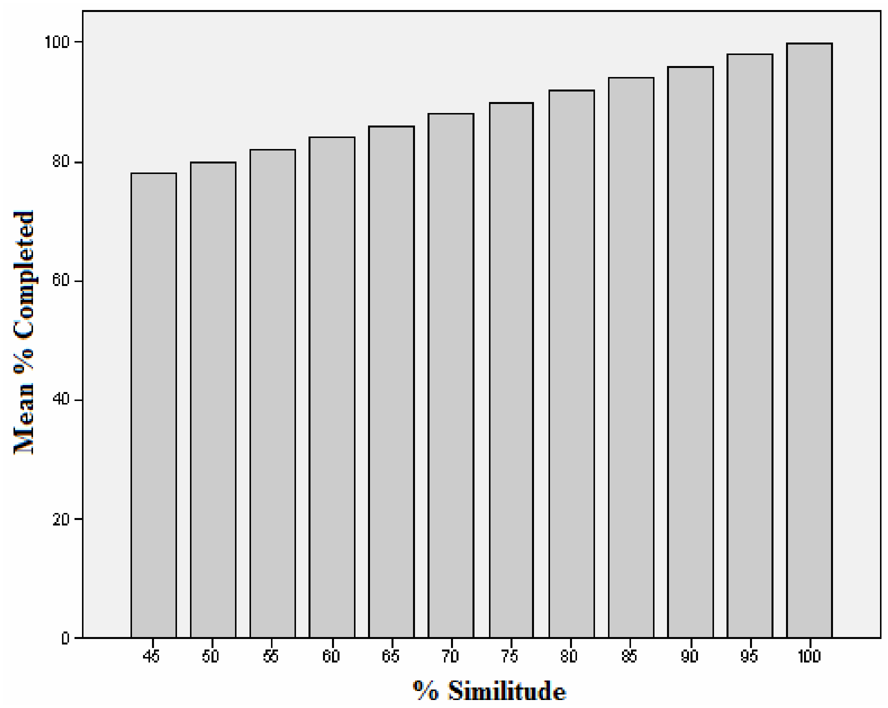


Fig. 10. % Plans completed related to the % of similitude for retrieved cases

average number of retrieved cases, the average number of completed plans and the average number of failed plans. Figure 8 let's us conclude that as the number of cases retrieved increases, the number of plans completed successfully increases and the number of plans failed decreases. In Figure 9 we can see that exits an inflection point in the row indicating the average number of retrieved cases and the column indicating the average number of plans completed. Such a point indicates, that from a certain number of retrieved cases on, the number of plans successfully completed stabilizes. Even, if the number of retrieved cases increases dramatically, the success for the proposed plans decreases.

Finally, the influence of the similitude degree obtained for the retrieved cases has been evaluated. Figure 10 presents the interaction between the percentage of plans successfully completed and the similitude degree obtained during the retrieval stage. In Figure 10 it can be seen that, generally, as more similar the retrieved cases are to the current problem, the possibility of plan success increases. CBP is based on the use of past experiences to resolve new problems. Obviously, if we can obtain past plans containing similar tasks and these plans were executed by nurses with similar skills, the probability of success increases. However, the CBP is highly affected by environmental changes, and the success for a plan depends not only on previous experiences but also on the incidents that can happen during the execution of the plan, which cause events of interruption and the corresponding replanning actions.

In the future, several industrial environments will require the use of new technologies that allow the personnel to carry out their tasks more efficiently. We have shown the potential of the CBP planning mechanism in a distributed intelligent environment on health care, providing a way to respond to some challenges of health care, related for example to the identification, control and health care planning. In addition, the use of RFID technology on people provides a high level of interaction among users and patients through the system and is fundamental in the construction of the intelligent environment. Furthermore, the use of mobile devices, when used correctly, can facilitate social interactions and knowledge transfer. The work presented can be easily adapted to similar environments by configuring certain parameters of the planning mechanism.

Acknowledgements

This work has been partially supported by the MCYT projects TIN2006-14630-C03-03/ and TIC2003-07369-C02-02 and the JCYL project SA104A05. Special thanks to Sokymat by the RFID technology provided and to Telefónica Móviles (Movistar) for the wireless devices donated.

References

Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7, 9–59 (1994)

Bajo, J., Corchado, J.M., Rodríguez, S.: Intelligent Guidance and Suggestions Using Case-Based Planning. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 389–403. Springer, Germany (2007)

Bratman, M.E.: Intentions, Plans and Practical Reason. Harvard University Press, Cambridge, M.A. (1987)

Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton, New Jersey (1957)

Camarinha-Matos, L., Afsarmanesh, H.: Design of a Virtual Community Infrastructure for Elderly Care. In: Camarinha-Matos, L. (ed.) Proceedings of PRO-VE 2002, Sesimbra, Portugal (2002)

Corchado, J.M., Laza, R.: Constructing Deliberative Agents with Case-based Reasoning Technology. International Journal of Intelligent Systems 18(12), 1227–1241 (2003)

Corchado, J.M., et al.: Intelligent Environment for Monitoring Alzheimer Patients. In: Agent Technology for Health Care. Decision Support Systems, Elsevier Science, Amsterdam (2007)

Emiliani, P.L., Stephanidis, C.: Universal access to ambient intelligence environments: opportunities and challenges for people with disabilities. IBM Systems Journal (2005)

Foster, D., McGregor, C., El-Masri, S.: A Survey of Agent-Based Intelligent Decision Support Systems to Support Clinical Management and Research. In: Armano, G., et al. (eds.) Proceedings of MAS*BIOMED 2005, Utretch, Netherlands (2005)

Friedewald, M., Da Costa, O.: Science and Technology Roadmapping: Ambient Intelligence in Everyday Life (AmI@Life). Working Paper. Institute for Prospective Technology Studies IPTS, Seville (2003)

Kohn, L.T., Corrigan, J.M.: Donaldson, To Err is human: Building a Safer Health System. Committee on Quality of Health Care in America, Institute of Medicine. National Academy Press, Washington, DC (1999)

Martín, Q.: Course about treatment of statistics data with SPSS, Hesperides (2001)

Nealon, J., Moreno, A.: Applications of Software Agent Technology in the Health Care domain. Whitestein series in Software Agent Technologies, Birkhauser (2003)

Sokymat (2006), http://www.sokymat.com

Susperregi, L., et al.: Una arquitectura multiagente para un Laboratorio de Inteligencia Ambiental en Fabricación. 1er. Taller de Desarrollo de Sistemas Multiagente (DESMA). Málaga, Spain (2004)

Wooldridge, M., Jennings, N.R.: Agent Theories, Architectures, and Languages: A Survey. In: Wooldridge, Jennings (eds.) Intelligent Agents, pp. 1–22. Springer, Heidelberg (1995)

ZigBee Standards Organization, ZigBee Specification Document 053474r13. ZigBee Alliance (2006)

Soft Query-Answering Computing in P2P Systems with Epistemically Independent Peers

Zoran Majkić[1] and Bhanu Prasad[2]

[1] ETF, Applied Mathematics Department, University of Belgrade, Serbia
majkic@etf.bg.ac.yu

[2] Department of Computer and Information Sciences, Florida A & M University, Tallahassee, FL, 32307, USA
bhanu.prasad@famu.edu

1 Introduction to P2P Systems: A Survey

Knowledge-based systems typically deals with incomplete and uncertain knowledge. Numerous extensions have been made to the logic programming and deductive databases in order to handle this incompleteness/uncertainty. These extensions can broadly be characterized into non-probabilistic and probabilistic formalisms. Approximate (uncertain) information can be considered as a kind of *relativization* of truth values for sentences. This is the standard approach used in several many-valued logics. For example, these many-valued logics include the smallest 4-valued logic programming based on Belnap's bilattice [1] with two additional logic values ('unknown' and 'possible' values for incomplete and locally-inconsistent information respectively), and infinitary fuzzy logics and their extensions [2] and combinations [3] also. In query-answering from distributed databases, especially in the web framework, we often have to deal with partial information and query-algorithms which are defined for real-time applications. Hence the obtained results are generally incomplete. That is, the *soft computing* query-answering in web database applications is naturally correlated with effective non-omniscient query-agents. There does not exist a centralized control mechanism to handle the entire information in web database applications. The complete answers to such queries are practically undecidable because an enormous amount of time is required to compute the answers. As a result, such answering systems cannot be used in practice. Some approximative algorithms are needed to obtain reasonable but incomplete answers. In addition, these algorithms need to be parametrically incremental.

P2P systems offer an alternate to the traditional client-server systems in some application domains. In P2P systems, every node (peer) acts as both the client and the server. Also, in these systems, part of the overall information will be available from an Internet-scale distributed environment. P2P computing has already been proved to be effective in many applications such as web search and bandwidth sharing [4], file sharing [5, 6], message interchange [7, 8], scientific computing [9], etc.

P2P computing has received a significant attention from the industry as a distributed computing framework that is lying between the traditional distributed systems and the web. P2P applications can be categorized according to the following four major directions [10]: instant messaging, file or data sharing, distributed computing and remote collaboration.

Some important applications include the following: ICQ [7] allows users to be notified when their friends are online. Also the users are allowed to send instant messages to their online friends. SETI@Home [9] is a distributed computing scientific experiment. It allows the task of dividing the large volumes of radio-telescope data processing into a set of subtasks and then send them to the peers for remote computation, in order to contribute to the search of extraterrestrial life. Groove [11] allows its users to create a secure shared space for collaborative activity. This includes managing, organizing and creating new workspaces, inviting others to workspaces, sending messages, monitoring the workspace and contact activity and setting up personal preferences. Napster [6] is a file sharing application, where the users can locate music files on other nodes and download them directly from those locations.

Napster employs a centralized database with references to files on peers. Gnutella [12] is another well-known P2P system that has no central database but requires a communication-intensive search mechanism. The gnutella-compatible P2P system [13] is a Gridella system that is based on the Peer-Grid (P-Grid) approach and provides a decentralized and scalable data access structure. P-Grid is a virtual binary tree that distributes replication over a community of peers and supports efficient search where the search time and the number of generated messages grow at $O(\log_2 n)$ for the number of data items n in the network. Peers in the P-Grid perform the construction and search/update operations in an unreliable environment where there is no centralized control or global knowledge. P-Grid's search structure exhibits the following properties: (1). It is completely decentralized, (2). All its peers serve as the entry points for the search, (3). Its interactions are strictly local, (4). It uses some randomized algorithms for the purpose of access and search, (5). It can give the probabilistic estimates for the search request success and (6). Its search is robust against node failures. Gridella is written in Java and is realized in 2002 under the GNU general public license.

At first glance, many of the challenges in designing the P2P systems seem to fall clearly under the category of distributed systems. However, upon closer examination, the fundamental problem in most P2P systems is the placement and retrieval of *data*. Indeed, current P2P systems focus strictly on handling semantic-free, large-granularity requests for objects by accepting an identifier. However, this feature limits their utility and also restricts the techniques that might be employed to distribute the data. These current sharing systems are largely limited to the applications in which the objects are large, opaque, and atomic, and whose content is well-described by their name. Also, they are limited in caching, pre-fetching, and in pushing the content to the object level, without knowing the overlap between the objects.

These limitations arise because the P2P world does not have suitable mechanisms in the areas of semantics, data transformation, and data relationships. Yet these are some of the core strengths for the data management community. In addition, queries, views and integrity constraints can be used to express relationships between the existing objects. Some of these new approaches to the P2P integration are provided.

One of the new P2P architectures is the Piazza system [14] where the data origins serve the original content and also the peer nodes cooperate with each other to store the materialized views and to answer the queries. These nodes are connected by bandwidth-constrained links and advertise their materialized views and belong to the spheres of cooperation with which they share the resources. The overall cost of answering a query includes the transfer cost from the storage provider or data origin to the query evaluator, the cost of the resources utilized at the query evaluator and other nodes, and the cost of transferring the results back to the query initiator. The *data placement problem* is to distribute the data and the work in such a way that the complete query workload is answered within the existing resource and bandwidth constraints and with the lowest cost possible. While a cursory glance at the data placement problem suggests many similarities with the multi-query optimization in a distributed database, there are substantial differences also. These differences are due to the lack of a centralized schema and a centralized administration in P2P systems. Ideally, the Piazza system accepts the current query workload, finds the commonalities among the queries, exploits the materialized views whenever it is cost-effective, distributes the work under resource and bandwidth constraints, and determines whether certain results should be materialized for future use. For scalability reasons, these decisions are made at a sphere of cooperation level rather than on a global basis. In order to perform this optimization, Piazza addresses a number sub-problems.

(1) Propagating the information about the materialized view: When a query is posed, the first step is to consider whether it can be answered using the data at the "nearby" storage providers and to evaluate the cost of doing so. This requires the query initiator to be aware of the existing materialized views and properties such as the location and data freshness.
(2) Consolidating the query evaluation and data placement: A node may pose a query that cannot be evaluated with the data available from known peers. In this case, the data must be retrieved directly from the data origins. However, at any given point, there may be many similar non-evaluable queries within the same sphere of cooperation and as a result, the sphere should choose an optimal strategy for evaluating all of them.
(3) Guaranteeing data freshness: In order to support dynamic data as well as dynamic workloads, Piazza must refresh materialized views when original data is updated. For scalability reasons, expiration times on data items are used rather than using a coherence protocol. This arrangement reduces the network traffic but does not achieve the strong semantics of the traditional databases.

The other, more semantic, approach is provided by Serafini et al. [15]. This paper introduces the Local Relational Model (LRM) as a data model that is specifically designed for P2P applications. LRM assumes that the set of all data in a P2P network consists of local (relational) databases, each with a set of acquaintances, which define the P2P network topology. For each acquaintance link, domain relations define the translation rules between the data items and coordination formulas define the semantic dependencies between the two databases. The two main goals of the data model are to allow inconsistent databases and to support semantic interpretability in the absence of a global scheme. LRM is semantically characterized in terms of *relational spaces*. A relational space is a pair that consists of a set of databases (the peers) and a domain relation which makes the relations among the domains of the databases explicit. The LRM semantics is a variation of the semantics of the distributed First Order Logic (FOL) and is an extension of the local models semantics proposed by Ghidini and Giunchiglia [16]. The coordination formulas, that relate the contents of the peer databases and also define the meaning of a coordination formula to be satisfied (with respect to a relational space), are used as the *deductive rules* and define a global answer to a query with respect to the relational space. The intuition is to compute the union of all the answers of the peer databases by considering the information carried by the domain relations. Reiter [17] has shown that any database can be uniquely represented by a generalized relation theory. He generalized this result by showing that a relational space is uniquely represented by a new kind of formal system called *multi-context system* that consists of a set of generalized relational theories (one per database) and a set of coordination rules.

The first seminal work which introduces the *autoepistemic semantics* for P2P databases, based on known (i.e. certain) answers from the peers, is presented by Lenzerini and Majkic [18] in February 2003. Six months later, this work was also referred and compared by the Franconi's group in their university report [19]. This modal logic framework for P2P database systems also guarantees the *decidability* for query answering, which is not supported by the first-order semantics. Franconi's group published another paper [20] whose content is identical to the university report [19] in all technical aspects except the fact that [20] has a reference to [21] instead of [18]. As a result, the trace to the original idea presented by Lenzerini and Majkic [18] was virtually lost.

The autoepistemic semantics for a peer database can serve as an example for the foundation of a sound query answering mechanism of a peer database that is based on *soft computing*. Unlike the classic first order logic, this work is intended to develop the decidable sound approximations for query answering in web applications. These approximations are based on non omniscient reasoning and incremental partial query-answering algorithms. The intensional logic (in this new framework) that is used for *inter-peer* mappings also extends the peer database semantics into the whole P2P system *flexible* query-answering semantics [22].

Comparative analysis

The *autoepistemic semantics* for a peer-database, specified as an encapsulation of a Data Integration System (DIS) with a *decidable* query answering [18], is compared with other approaches [19]. Such a formal framework does not specify *if* and *how* the local epistemic knowledge of any peer is influenced by the epistemic knowledge of other peers. In fact, it is possible to distinguish at least two extreme scenarios presented in the literature developed from the seminal article [18]. The extreme scenarios are *strongly-coupled* and *weakly-coupled* P2P database systems. They are explained.

1. The 'strong' (*extensional*) Global or Local As View (GLAV) mapping which directly extends the data-integration paradigm to the P2P systems also [21, 23, 24]. It is paraphrased by an *imperative* sentence '*John must know all facts about the "Italian art in the 15th century" known by Peter*', where 'John' and 'Peter' are two different peers, and
2. The 'weak' (*intensional*) mapping [25, 26, 27] is paraphrased by a *belief*-sentence '*John believes that Peter also knows something about "Italian art in the 15th century"*'. Such mappings are weaker than internal extensional GLAV mappings of a peer. As a result, they guarantee the independence of peer individuality in the presence of map-

pings between the peers also (more details are available in Majkic [25]). They also distinguish *certain* answers of a peer 'John' from a *possible* answers of 'Peter'.

The motivation for the second approach discussed in this paper is briefly explained. It is well known that P2P mappings based on GLAV logical mappings between the peer ontologies are very *restrictive*. That is, the logic implications used to impose direct semantic relationships between the peer ontologies are too much constrictive in the sense that given such mappings, it is not free to independently change the proper local ontology of some peer. The internal peer mapping and also the external mapping with other peers have the *same* (extensional) expressive power so that the peer individuality is destroyed. The logic implications from other peers toward the considered peer impose strong requirements for the local peer ontology. This is a problem for web semantic P2P integration where different peers are managed *without* a centralized administration. Thus, such ontology integration is not adequate for internet-based P2P systems that are large.

A detailed introduction and the fundamental differences between the two approaches (i.e., centralized and non-centralized) can be found in the recent article published by Majkic [25]. As a result, the present paper provides only a comparative analysis with the recent article.

This paper deals with the *full* epistemic independency of peer databases. The databases can change their ontology and/or extension of their knowledge independently from other peers without any communication to other peers. This new semantics for a P2P database system incorporates an intensional kind of semantics for mapping between the completely independent peers. The mapping is fundamentally different from the Local As View (LAV) and Global As View (GAV) mappings traditionally adopted in data integration community. This is the reason why we obtain the fully modular P2P system in which each module supports the query-answering of a single peer that is based only on the known answers and can easily be implemented by a data-intensive grid computation technology. In addition, each grid node is responsible for wrapping the data of one particular peer from the web during the query-answering process. Having fully independent peers, it is enough to associate each pair (peer, query formulae) to a particular resource of grid computing in order to obtain a known answer from such peer. In this way, we intend to obtain some very robust P2P systems that are able to answer the user queries even when the intended mappings between the peers become incorrect With Respect To (w.r.t.) the successively modified ontologies (relational database schemas) of peers by their independent group of developers. In addition, our intension is to have the possibility for naturally mapping the P2P database systems into grid computation. The paper by Majkic [25] has clearly defined such a framework for the P2P database systems. It has presented the encapsulation of peer database into an Abstract Data Type (ADT) with query-answering methods and also defined the semantics for the intensional equivalence. This is based on the Montague's modal semantics for natural languages [28] which can be considered as part of a *sound* query rewriting *algorithm* presented in this paper.

The complete integration of the *intensional equivalence* for a query answering in P2P systems needs the definition of the new *intensional* modal First Order Logic (FOL). However the modal FOL is not the primary issue of this paper. Instead, the focus is on soft computing of query answers based on the incomplete and incremental algorithms. The full intensional logic formalization of such *non omniscient* intensional contextual reasoning for *query-agents* in P2P can be found in Majkic [29].

The rest of the paper is organized as follows: A brief introduction to the data integration methodology for WEB applications that are used to define a peer database for semantically rich P2P database systems is presented in Section 2.

Section 3 presents a formal P2P framework based on the epistemically independent database peers and intensional mapping (that is based on intensionally equivalent views of different peers) between them.

In Section 4, we introduce the intended semantics for epistemically independent peers in two equivalent forms. The first one is the *algebraic* specification of the encapsulation of a standard DIS into peer. The specification is implemented by using an ADT with an input and output interface. The input for such an ADT is a user query, while the output is a known answer of the DIS (answers that are true for all models of this database. A number of different models is the result of the incomplete web information obtained by the wrappers). The method of this ADT is implemented by the query-rewriting algorithms used in GLAV data integration. The second one is *logic* specification based on the autoepistemic modal semantics where the set of Kripke possible worlds [30] is the set of models of peer database, and the universal modal "know" operator is same as in the S5 Kripke framework [31]. After that, an incremental query-answering algorithm is provided for the whole P2P system. The soft computing query-answer is based on the intensional view-based mappings between peers. It is the partial disjoint union of the answers of different peers.

Section 5 is more theoretical addendum, dedicated to the specialists in modal logics. The basic autoepistemic mono-modal semantics (see Appendix) used for a particular database peer is extended to the whole network of database peers in the P2P system. This extension is performed by using the hybrid modal logic in order to define a "know" modal operator for each particular database peer. It is also shown that the logic has correct semantics for the soft computing query-rewriting algorithm given in Section 4.

2 An Example Peer Database Defined by WEB Data Integration

In this section, we illustrate the formalization of a data integration system [32] that is based on the relational model with integrity constraints.

In the relational model, predicate symbols are used to denote the relations in the database, whereas constant symbols denote the values stored in the relations. We assume to have a fixed (infinite) alphabet Γ of constants. Unless specified otherwise, we consider only the databases defined over such an alphabet. In such a setting, the Unique Name Assumption (UNA) is implicit (that is, different constants denote different objects).

A *relational schema* (or simply a *schema*) consists of the following:

1. An *alphabet* $\mathcal{A}$ of predicate (or relation) symbols, each one with an associated arity. Arity represents the number of arguments of a given predicate or the number of attributes of a given relation.
2. A set $\Sigma_{\mathcal{G}}$ of *integrity constraints*. That is, assertions on the symbols of the alphabet

$\mathcal{A}$. The assertions express the conditions that are intended to be satisfied in every database that is coherent with the schema. In the proposed framework, we consider two kinds of constraints:
2.1 *Key constraints* (we assume that, in the global schema, there is exactly one key constraint for each relation).
2.2 *Foreign key constraints*: a foreign key constraint is a statement of the form $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, where r_1 and r_2 are relations and $\mathbf{A}$ is a sequence of distinct attributes of r_1 and $\mathbf{B}$ is $key(r_2)$. Such a constraint is satisfied in a database $\mathcal{DB}$ if for each tuple t_1 in $r_1^{\mathcal{DB}}$ there exists a tuple t_2 in $r_2^{\mathcal{DB}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$ where $t_1[\mathbf{A}]$ is the projection of the tuple t_1 over $\mathbf{A}$.

A *relational database* (or simply a *database*) $\mathcal{DB}$ for a schema $\mathcal{C}$ is a set of relations in which the constants are atomic values and in addition, there is one relation $r^{\mathcal{DB}}$ of arity n for each predicate symbol r of arity n in the alphabet $\mathcal{A}$. The relation $r^{\mathcal{DB}}$ is the interpretation in $\mathcal{DB}$ of the predicate symbol r, in the sense that it contains the set of tuples that satisfies the predicate r in $\mathcal{DB}$.

A *relational query* is a formula that specifies a set of tuples that needs to be retrieved from a database. We consider the class of *safe conjunctive queries* in which the answer to a query q of arity n over a database $\mathcal{DB}$ for $\mathcal{G}$, denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $(c_1, \ldots, c_n)$ such that when each x_i is replaced with c_i then the formula $\exists(y_1, \ldots, y_n).conj(x_1, \ldots, x_n, y_1, \ldots, y_m)$ evaluates to true in $\mathcal{DB}$.

When $n = 1$ and all variables $y_i, 1 \le i \le m$ are replaced by the constants in Γ then the queries become Yes/No type.

2.1 DIS

A **DIS** $\mathcal{I}$ is a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

- The *global schema* is expressed in the relational model with constraints $\Sigma_{\mathcal{G}}$.
- The *source schema* is expressed without integrity constraints.
- The *mapping* $\mathcal{M}$ is defined following the GAV approach. That is, to each relation r of the global schema $\mathcal{G}$ we associate a query $\rho(r)$ over the source schema $\mathcal{S}$. Logically, such mapping is *standard material implication* $\rho(r)(X) \Rightarrow r(Y)$ where $X \subseteq Y$ are the sets of attributes of these (virtual) predicates (in the case when $X \subset Y$ we have *incomplete* information).

We call any database for $\mathcal{G}$ as *global database* for $\mathcal{I}$ or simply *database* for $\mathcal{I}$.
Let $\mathcal{D}$ be a finite *source database instance* for $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ that is constituted by one relation $r^{\mathcal{D}}$ for each source r in $\mathcal{S}$. A database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal* with respect to $\mathcal{D}$ if:

- $\mathcal{B}$ satisfies the integrity constraints $\Sigma_{\mathcal{G}}$ of $\mathcal{G}$.
- $\mathcal{B}$ satisfies $\mathcal{M}$ with respect to $\mathcal{D}$. I.e., for each relation r in $\mathcal{G}$, the set of tuples $r^{\mathcal{B}}$ that $\mathcal{B}$ assigns to r is a superset of the set of tuples $\rho(r)^{\mathcal{D}}$ computed by the associated query $\rho(r)$ over $\mathcal{D}$, i.e., $\rho(r)^{\mathcal{D}} \subseteq r^{\mathcal{B}}$.
- we denote by $sem^{\mathcal{D}}(\mathcal{I})$ the set of databases for $\mathcal{I}$ that are legal w.r.t. $\mathcal{D}$, i.e., that satisfies both the constraints of $\mathcal{G}$ as well as the mapping $\mathcal{M}$ with respect to $\mathcal{D}$. If $sem^{\mathcal{D}}(\mathcal{I}) \neq \emptyset$, then $\mathcal{I}$ is said to be *consistent* w.r.t. $\mathcal{D}$.

Note that the above definition amounts to consider any view $\rho(r)$ as *sound* [33, 34]. It means that the data provided by the sources is only an (incomplete) subset (can be a proper subset) of the data that would satisfy the relations of the global schema.

By the definition above, it is clear that the semantics of a data integration system is formulated in terms of a *set* of databases, rather than a single one.

Retrieved global database $ret(\mathcal{I}, \mathcal{D})$ for a given finite *source database instance* $\mathcal{D}$ is defined as follows. For each relation r of the global schema, we compute the relation $r^{\mathcal{D}}$ by evaluating the query $\rho(r)$ over the source database $\mathcal{D}$. We assume that for each relation r of the global schema, the query $\rho(r)$ over the source schema $\mathcal{S}$ that the mapping $\mathcal{M}$ associates to r preserves the key constraint of r (this may require that $\rho(r)$ implements a suitable duplicate record elimination strategy). This assumption is to make sure that the retrieved global database satisfies all the key constraints in $\mathcal{G}$.

Query-answering in DIS

Let q be a conjunctive query to a DIS $\mathcal{I}$ (atoms in q have symbols in $\mathcal{G}$ as predicates). The *set of certain answers* $q^{\mathcal{I},\mathcal{D}}$ to q w.r.t. $\mathcal{I}$ and $\mathcal{D}$ is the set of tuples t of constants of the same arity as q, and $t \in q^{\mathcal{B}}$, for each $\mathcal{B} \in sem^{\mathcal{D}}(\mathcal{I})$.

Let $\Delta_{\mathcal{I}}$ be a set of formulas belongs to a language $\mathcal{L}$ used to define the DIS $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a formula A belongs to $\mathcal{L}$. We denote a *model-theoretic consequence* relation (logical entailment) with $\vDash$, defined as follows: $\Delta_{\mathcal{I}} \vDash A$ when all models of the formulas contained in $\Delta_{\mathcal{I}}$ are models of A; $\vDash_{\mathcal{B}} A$ when a formula A is true in a particular model $\mathcal{B}$. Then, the certain answer to $q(\mathbf{x})$ is given in the following way:
$q^{\mathcal{I},\mathcal{D}} = \{ \, t \mid t \in \Gamma^{arity(q)}$ and $\vDash_{\mathcal{B}} q(t)$ for each $\mathcal{B} \in sem^{\mathcal{D}}(\mathcal{I})\}$.

As a result, in this framework with key and foreign key constraints, we can visualize a DIS $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ as a logical theory $\mathcal{P}_{\mathcal{G}}$ (a definite logic program is provided by Cali et al. [32]) that is composed by: (1). A retrieved global database $ret(\mathcal{I}, \mathcal{D})$ as an extensional part of a database theory and (2). The integrity constraints for a global schema as its intensional part. The existential quantifiers in the foreign key constraints are eliminated by introducing the appropriate Skolem functions:
$HT(\mathcal{G}) = \{f_{r,i} \mid r \in \mathcal{G}$ and $i \leq arity(r)$ and $i \notin key(r)\}$.
Each $f_{r,i}$ is a function symbol with the same arity as the number of attributes of $key(r)$, i.e., $arity(f_{r,i}) = arity(key(r))$.

Intuitively, the role of the term $f_{r,i}(\alpha_1, \ldots, \alpha_k)$ is to denote the value in the i-th column of the tuple of r having $\alpha_1, \ldots, \alpha_k$ in the key columns. The domain of such functions is the alphabet Γ.

The construction of the canonical *model* for a global schema of this logical theory $\mathcal{P}_{\mathcal{G}}$ is presented by Cali et al. [32]. This model is an universal (canonical) one. That is, it is an *initial* minimal Herbrand model [35] such that for every other legal database model $\mathcal{B}$ of the global schema, there is a unique homomorphism [32] $\psi : can(\mathcal{I}, \mathcal{D}) \longrightarrow \mathcal{B}$ and defines the following sound and complete query rewriting algorithms:

- $exp_{\mathcal{G}}(\_)$ that expands the original conjunctive query $q(\mathbf{x})$ over a global schema into $exp_{\mathcal{G}}(q(\mathbf{x}))$ query over $ret(\mathcal{I}, \mathcal{D})$.
- $unf_{\mathcal{M}}(\_)$ algorithm which unfolds the resulting query over $ret(\mathcal{I}, \mathcal{D})$ and returns with the query $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q(\mathbf{x})))$ over a source data base $\mathcal{D}$, such that for each tuple $t \in \Gamma^{arity(q)}$, $t \in q(\mathbf{x})^{\mathcal{I},\mathcal{D}}$ iff $t \in [unf_{\mathcal{M}}(exp_{\mathcal{G}}(q(\mathbf{x})))]^{\mathcal{D}}$

so that the certain answer to a conjunctive query $q(\mathbf{x})$ is equal to $q(\mathbf{x})^{\mathcal{I},\mathcal{D}} =_{def} \{t \mid t \in \Gamma^{arity(q)} \ and \ \vDash_{M_{\mathcal{D}}} unf_{\mathcal{M}}(exp_{\mathcal{G}}(q(t)))\}$, where $M_{\mathcal{D}}$ is the unique minimal Herbrand model of a source database $\mathcal{D}$.

2.2 Finite Canonical Database

Here we introduce a new approach to canonical model and this approach is more close to the data exchange approach presented by Fagin et al. [36]. The approach is not restricted to the existence of query-rewriting algorithms. As a result, it can be used to define Coherent Closed World Assumption [37] for data integration systems in the absence of query-rewriting algorithms also. The construction of the canonical *model* for a global schema of the logical theory $\mathcal{P}_{\mathcal{G}}$ for data integration system is similar to the construction of the canonical *database* $can(\mathcal{I}, \mathcal{D})$ described by Cali et al. [32]. The *difference* lies in the fact that in the construction of this revisited canonical model denoted by $can_M(\mathcal{I}, \mathcal{D})$, for a global schema, fresh *marked null values* (set Ω of Skolem constants [35]) are used instead of the terms involving Skolem functions. This follows the idea of the construction of the restricted chase of a database described by David et al. [38]. Thus, we enlarge a set of constants of our language by $\Gamma_U = \Gamma \bigcup \Omega$.

Topor and Sonenberg [35] informally proposed the term *canonical model* to describe a model that is selected (often from many incomparable minimal Herbrand models) to represent the 'meaning' of logical programs. Another motivation for concentrating on canonical models is the view [39] that many logic programs are appropriately thought of as having two components: an *intensional* database (IDB) that represents the reasoning component and the *extensional* database (EDB) that represents a collection of facts. Over the course of time, we can "apply" the same IDB to many quite different EDBs. In this context it makes sense to think of the IDB as if it is implicitly defining a transformation from an EDB to a set of derived facts. We would like the set of derived facts to be the canonical model.

Now, we inductively construct the revisited canonical database model $can_M(\mathcal{I}, \mathcal{D})$ over the domain Γ_U by starting from $ret(\mathcal{I}, \mathcal{D})$ (here is an EDB) and repeatedly applying the following rule based on IDB:

if $(x_1, \dots, x_h) \in r_1^{can_M(\mathcal{I},\mathcal{D})}[\mathbf{A}]$, $(x_1, \dots, x_h) \notin r_2^{can_M(\mathcal{I},\mathcal{D})}[\mathbf{B}]$, and the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is in $\mathcal{G}$,
then insert in $r_2^{can_M(\mathcal{I},\mathcal{D})}$ the tuple t such that
- $t[\mathbf{B}] = (x_1, \dots, x_h)$, and
- for each i such that $1 \leq i \leq arity(r_2)$, and i not in $\mathbf{B}$, $t[i] = \omega_k$, where ω_k is a fresh marked null value.

Note that the above rule does enforce the satisfaction of the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ (which is a part of IDB), by adding a suitable tuple in r_2: the key of the new tuple is determined by the values in $r_1[\mathbf{A}]$, and the values of the non-key attributes are formed by means of the fresh marked values ω_k during the application of the rule above.

The above rule defines the "immediate consequence" monotonic operator T_B defined by:

$T_B(I) = I \bigcup \{ A \mid \; A \in B_{\mathcal{G}}, A \leftarrow A_1 \wedge ... \wedge A_n$ is a ground instance of a rule in $\Sigma_{\mathcal{G}}$ and $\{A_1, ..., A_n\} \in I\}$ where, at the beginning $I = ret(\mathcal{I}, \mathcal{D})$, and $B_{\mathcal{G}}$ is a Herbrand base for a global schema. Thus, $can_M(\mathcal{I}, \mathcal{D})$ is a least fixpoint of this immediate consequence operator.

Example 1: Suppose that we have two relations r and s in $\mathcal{G}$, both of arity 2, and having the first attribute as the key and that the following dependencies hold on $\mathcal{G}$:
$r[2] \subseteq s[1], \quad s[1] \subseteq r[1]$.

Suppose that the retrieved global database (EDB) stores a single tuple (a, b) in r. Then, by applying the above rule, we insert the tuple (b, ω_1) in s; successively we add (b, ω_2) in r, then (ω_2, ω_3) in s and so on. Observe that the two dependencies are cyclic. In this case the construction of the canonical database requires an *infinite* sequence of applications of the rules. The following table represents the correspondence between the old and revisited canonical database:

$r^{can(\mathcal{I},\mathcal{D})}$	$s^{can(\mathcal{I},\mathcal{D})}$	$r^{can_M(\mathcal{I},\mathcal{D})}$	$s^{can_M(\mathcal{I},\mathcal{D})}$
a, b	$b, f_{s2}(b)$	a, b	b, ω_1
$b, f_{r2}(b)$	$f_{r2}(b), f_{s2}f_{r2}(b)$	b, ω_2	ω_2, ω_3
$f_{r2}(b), f_{r2}^2(b)$	$f_{r2}^2(b), f_{s2}f_{r2}^2(b)$	ω_2, ω_4	ω_4, ω_5
$f_{r2}^2(b), f_{r2}^3(b)$	$f_{r2}^3(b), f_{s2}f_{r2}^3(b)$	ω_4, ω_6	ω_6, ω_7
..	..	..	..

Thus, the canonical model $can_M(\mathcal{I}, \mathcal{D})$ is a legal database model for the global schema. □

Let us also introduce an unary predicate $Val(x)$, such that for any constant $c \in \Gamma_U$, $Val(c)$ is true if $c \in \Gamma$, false otherwise. Each certain answer of the original user query $q(\mathbf{x})$, $\mathbf{x} = \{x_1, ..., x_k\}$ over a global schema is equal to the answer $q_L(\mathbf{x})^{can_M(\mathcal{I},\mathcal{D})}$ of the *lifted* query $q_L(\mathbf{x}) \equiv q(\mathbf{x}) \wedge Val(x_1) \wedge ... \wedge Val(x_k)$ over this canonical model. Thus, in the cases when it is possible to materialize this canonical model, certain answers could be obtained over such a database. Usually it is not possible because (as in the example above) this canonical model is *infinite*. In that case, we can use the revisited fixpoint semantics described by Majkic [40]. This is based on the fact that after some point, the new tuples added into a canonical model insert only new Skolem constants that are not useful in order to obtain *certain* answers. In fact, Skolem constants are not part of any certain answer to conjunctive query. Consequently, we are able to obtain a *finite subset* of a canonical *database*, which is big enough to obtain certain answers.

Example 2: If we consider Example 1, the finite database $r = \{(a, b), (b, \omega_2), (\omega_2, \omega_4)\}, \quad s = \{(b, \omega_1), (\omega_2, \omega_3)\}$ is such a finite least fixpoint which can be used in order to obtain certain answers to the lifted queries. □

In fact, we introduced marked null values (instead of Skolem functions) in order to define and materialize such *finite* database. It *is not* a model of the DIS (which is infinite) but has all necessary query-answering properties. It is able to give all certain answers to the conjunctive queries over a global schema. Thus, it can be materialized and used for query answering, instead of query-rewriting algorithms. Let us denote such finite database by $\mathcal{C}_M(\mathcal{I},\mathcal{D})$. So, we can prove the following property:

Proposition 1. *Yes/No query* $q(\mathbf{c})$, $\mathbf{c}=\{c_1,...,c_n\}\in \Gamma^n, n = arity(q)$ *over a canonical database* $\mathcal{C}_M(\mathcal{I},\mathcal{D}) \subseteq can_M(\mathcal{I},\mathcal{D})$ *and the rewritten query* $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q(\mathbf{c})))$ *over a source database* $\mathcal{D}$*, return the same logical true/false value.*
There is a unique homomorphism $\varphi : can_M(\mathcal{I},\mathcal{D}) \longrightarrow can(\mathcal{I},\mathcal{D})$.

Proof. The first part of this proposition is a direct consequence of the query rewriting algorithms and it is also based on the fact that each Yes/No query over a source database $\mathcal{D}$ and over a canonical database $can_M(\mathcal{I},\mathcal{D})$ returns with the true/false value. Proof of the second part: from a definition of certain answers (that is, answers which are true in all models of a database with incomplete information. Each possible completion of this incomplete information will provide a possible model) we have that $q^{\mathcal{I},\mathcal{D}} = \{\ t \mid t \in \Gamma^{arity(q)}$ and $\vDash_{\mathcal{B}} q(t)$ for each $\mathcal{B} \in sem^{\mathcal{D}}(\mathcal{I})\}$. Thus, $t \in q^{\mathcal{I},\mathcal{D}}$ (or, equivalently, $\vDash_{can_M(\mathcal{I},\mathcal{D})} q(t)$) iff $\vDash_{\mathcal{B}} q(t)$ for each $\mathcal{B} \in sem^{\mathcal{D}}(\mathcal{I})$, but, $\vDash_{can_M(\mathcal{I},\mathcal{D})} q(t)$ iff $\vDash_{M_{\mathcal{D}}}$ $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q(t)))$.

Example 3: If we consider Example 1, we define φ homomorphism as follows: for any constant in Γ, it is an identity function (this is an intrinsic homomorphism property). For Skolem constants we have that $\varphi(\omega_1) = f_{s2}(b)$, $\omega_{2i} = f_{r,2}^{2i-1}(b)$ and $\omega_{2i+1} = f_{s,2}(\omega_{2i}) = f_{s,2}f_{r,2}^{2i-1}(b)$, for $i = 1, 2, ...$ □

In order to define a peer database with relational view-based logic schemata (peer ontology), the previous example on the application of data integration in web applications will be used as the building blocks for the P2P systems defined in the next section. The possibility of using the negation logic operator also for querying such DISs with generalized closed-world assumption was elaborated by Majkic [37].
By abstracting the internal structure of such peers, we will obtain ADTs for peers, while their relational view-based external interface (for the users) will be used for intensional mappings between the peers based on relational views.

3 Formal Definition for the P2P Framework

In this paper we propose a formal P2P framework based on the following considerations [18]:

- The main point is that every peer can be seen as an ADT which acts at the same level, with no unifying structure above them. The importance of the formalization of a database peer as an ADT is that it is *completely independent* from other peers so that it is able to give the complete known answer w.r.t. to its local information. In addition, it acts as an independent data module in the web framework. In order to respond to the complex user queries (union of conjunctive queries) we assume that the expressive power of peers can be generally given by a single-encapsulated data integration semantic. In this way, by considering the (incomplete) sources extracted by wrappers, we may enrich the peer database schema by integrity constraints in order to overcome incompleteness of heterogenous web information. We assume that each ADT peer has a unique model or a *canonical (universal)* [32, 36] global database (as presented in Section 2.2) that responds to the user queries by *certain* (i.e., known [41]) answers.

- What is needed here is: given any two peer databases then a mechanism that is able to define mappings between them and without resorting to any unifying (global) conceptual structure.
- We need a completely decentralized network of database peers. All peers serve as entry points for search, offering their relational schema in order to formalize user queries.
- Query answering is fundamentally based on the interactions which are strictly local and are guided by locally defined mappings of a considered peer w.r.t. other peers in a network.
- We do not want to limit a-priori the topology of the mapping assertions between the peers in the system. In particular, we do not impose the acyclicity of assertions.
- We seek for a semantic characterization that leads to a situation where soft computation of query answering is decidable and possibly polynomially tractable. In other words, it is theoretically incomplete and context dependent w.r.t. the peer where the user query was formulated. This contextual dependence is based on the fact that the obtained answer will be complete only from the data peer where the query is addressed. Other answers from other locally 'connected' peers will give other possible information which is theoretically approximated and incomplete.

In what follows, we conceive a peer P_i as an ADT software module characterized by a network ontology G_i expressed in a language $\mathcal{L}_{\mathcal{O}}$ over an alphabet $\mathcal{A}_{\mathcal{G}_i}$. The internal structure of a peer database is hidden to the user and is encapsulated in such a way that only its logical relational schema $\mathcal{G}_{T_i}$ (global schema if it is a DIS described in Section 2, $\mathcal{I}_i = (\mathcal{G}_i, \mathcal{S}_i, \mathcal{M}_i)$, where $\mathcal{G}_i = (\mathcal{G}_{T_i}, \Sigma_{T_i})$, Σ_{T_i} are the integrity constraints, $\mathcal{S}_i$ is a source schema and $\mathcal{M}_i$ is a set of generally GLAV mappings between a global schema $\mathcal{G}_{T_i}$ and a source schema $\mathcal{S}_i$) can be seen by the users. The peer database is able to respond to the union of conjunctive queries by *certain* answers.

In order to be able to communicate with other peer P_j in the network $\mathcal{N}$, each peer P_i has also an export-interface module $\mathcal{M}^{ij}_{EXP}$ composed by the groups of ordered pairs of intensionally-equivalent queries. We define any two queries q_i and q_j intensionally-equivalent if and only if (iff) the set of variables (i.e., attributes) in the heads of these two queries are equal and both queries define the same concept (meaning) [42, 43]. We denote this intensional equivalence by $q_i \approx q_j$. Intuitively, $q_i \approx q_j$ means that the concept in the ontology $\mathcal{G}_i$ (represented by q_i) and the concept in the ontology $\mathcal{G}_j$ (represented by q_j) have the same meaning but their extension in the actual world are independent. It is different from the case when this mapping between the views (conjunctive queries) of two peers is defined as a material implication $q_i \Rightarrow q_j$[23] where the extension of q_i must be contained in the extension of q_j.

In the simplest case, q_i is an atomic concept in the corresponding network ontology. However, in general, the mapping concepts between different ontologies requires to use queries over such ontologies. This is the reason why we introduce the query language $\mathcal{L}_{\mathcal{M}}$ in our formalization.

Note that $q_i \approx q_j$ does *not* mean that q_i logically implicates q_j or vice versa, as in GLAV mapping definitions. As a result, we denote these view based particular P2P mappings by the new symbol $\approx$ which represents an *intensional equivalence relation* (which is reflexive, symmetric and transitive) between the views.

Definition 1. *The P2P network system $\mathcal{N}$ is composed by $2 \leq N$ independent peers. Each peer module P_i is defined as follows:*

$$P_i := \langle (\mathcal{G}_i, \mathcal{S}_i, \mathcal{M}_i), \bigcup_{i \neq j \in N} \mathcal{M}^{ij}_{EXP} \rangle$$

where $\mathcal{I}_i = (\mathcal{G}_i, \mathcal{S}_i, \mathcal{M}_i)$ is the encapsulated data integration system with $\mathcal{G}_i = (\mathcal{G}_{T_i}, \Sigma_{T_i})$: only $\mathcal{G}_{T_i}$, which is its logical relational schema, is its non-hidden part and can be seen by the users in order to formulate a query. $\mathcal{M}^{ij}_{EXP}$ is a (possibly empty) interface to other peer P_j in the network, defined as a group of query-connections, denoted by $(q^{ij}_{1k}, q^{ij}_{2k})$ where q^{ij}_{1k} is a conjunctive query defined over $\mathcal{G}_{T_i}$, while q^{ij}_{2k} is a conjunctive query defined over the ontology $\mathcal{G}_{T_j}$ of the connected peer P_j:

$$\mathcal{M}^{ij}_{EXP} = \{(q^{ij}_{1k}, q^{ij}_{2k}) \mid q^{ij}_{1k} \approx q^{ij}_{2k},\ and\ 1 \leq k \leq \mid ij \mid \}$$

where $\mid ij \mid$ denotes the total number of query-connections of a peer P_i towards a peer P_j.

4 Soft Computing Incremental P2P Query Answering

We assume that the conjunctive queries to a P2P network $\mathcal{N}$ are specified over the network ontology $\mathcal{G}_{T_i}$ of a peer P_i and are expressed in a conjunctive query language $\mathcal{L}_{\mathcal{O}}$ over an alphabet $\mathcal{A}_{\mathcal{G}_i}$. A query is intended to provide the specification of the data to be extracted from the P2P network $\mathcal{N}$. We need that each peer P_i (which, with its DIS with web incomplete information, can have many legal database instances) responds to the user conjunctive query q_C by providing only **known** answers [33] (i.e., the answers that are true in all legal database instances. In other words, the answers that are true in all minimal Herbrand models of a peer database).

4.1 Known Answers of a Peer Database

There are two alternative ways to formally consider such a *known* query answering:

- **ADT Framework:** We can enrich the global schema $\mathcal{G}_{T_i}$ by a new unary predicate $Val(\_)$ such that $Val(c)$ is true if $c \in$ **dom** (domain of values of the local ontology) is a constant of the local ontology of a given peer. The user's queries $q_C(\mathbf{x})$, where $\mathbf{x} = x_1, .., x_k$ is a non empty set of variables, over the global schema of a peer database are *conjunctive queries*. The ADT module of a peer transforms any original user's query $q_C(\mathbf{x})$ into a **lifted query** (denoted by q) over the peer's global schema such that $q := q_C(\mathbf{x}) \wedge Val(x_1) \wedge ... \wedge Val(x_k)$.

  The universal (canonical) database $can_M(\mathcal{I}, \mathcal{D})$ of the encapsulated DIS with the source database $\mathcal{D}$ (presented in Section 2.2) has the interesting property of faithfully representing all the legal databases (the construction of the canonical database is similar to the construction of the *restricted chase* of a database described by David et al. [38]).

Thus, theoretically, the lifted query will filter only the known answers from $can_M(\mathcal{I}, \mathcal{D})$ (answers without Skolem constants (introduced for incomplete information [40])). In practice, we do not materialize this canonical database in order to give the answer to the query. Instead, we use a *query rewriting techniques under constraints* in DISs (for example, a data integration system with GAV mapping and with the key and inclusion integrity constraints [32] is presented in Section 2.2) to submit the obtained *rewritten* query directly to the source databases (of this DIS) extracted from the web by the wrappers.

- **Modal Logic Framework:** The following Levesque [44] queries should be the formulas in an epistemic modal logic. Queries should be permitted to address the aspects of the external world as represented by the databases as well as the aspects of the database itself (i.e. aspects of what the database knows about the external world [41]). In this context, the integrity constraints are best viewed as sentences about *what the database knows*. The constraints are represented as logic sentences in FOL without free variables and all the variables are quantified by the universal quantifier.

  On this view, integrity constraints are *modal sentences* and hence they are formally identical to a strict subset of permissible database queries. They have defined a first-order modal language $KFOPCE$ with a single universal modal operator $\Box \equiv K$ (K represents "know"). The resulting modal logic is weak S5 (also known K45) with transitive and Euclidean accessibility relation. In fact, given a single data integration system, the possible worlds are legal databases of this DIS (i.e., minimal Herbrand models of a peer's database) with each one connected to all other. Thus a modal query formula $\Box q_C$ is true (certain) iff q_C is true at all possible worlds (i.e., legal databases) accessible from that world. It is in agrement with the S5 modal logic where the same fixed set of worlds can be thought of as being accessible from any word.

  It is proved [41] that such modal semantics leads to first order, indeed Prolog-like, query evaluators. Generalizing the conjunctive first order queries to normal $KFOPCE$ queries genuinely increases the expressiveness of the query language without sacrificing the computational advantages of the first order query evaluation.

  We will generalize the above approach to a P2P network of disjoint peers (DISs) in a "locally modular" way. The generalization is performed by introducing some new peer or by eliminating some old peer but the frame of this new mono-modal logic ML_{DB} is only locally changed in a very bounded way. The resulting accessibility relation of this mono-modal logic for a P2P system is composed by completely *disjoint partitions* of worlds. Each partition corresponds to one particular peer and is composed by all legal databases (possible worlds) of that peer, connected each one with all others. As a result, each single peer can be considered as "local" S5 modal epistemic logic that represents the incompleteness of the peer's database.

  We cannot use only the simple $\Box q_C$ formulae in order to obtain known (certain) answers of the given particular peer, because we need to specify also *this* particular peer of a P2P system in the syntax of the query: for such necessity we will use a *hybrid extension* of a mono-modal logics (see the Appendix).

4.2 Soft Computing Query Answering Algorithm

The general scenario for query answering in a P2P network system $\mathcal{N}$ composed of the set of peers $P_i := \langle(\mathcal{G}_i, \mathcal{S}_i, \mathcal{M}_i), \bigcup_{i \neq j \in N} \mathcal{M}^{ij}_{EXP}\rangle$, $1 \leq i \leq N$, can be described as follows:
Given an initial conjunctive user query $q_C(\mathbf{x})$ (where $\mathbf{x} = x_1, .., x_k$ is a non empty set of variables) over the global schema of a peer P_k, we denominate the tuple $(q_k(\mathbf{x}), P_k)$ as the "answering pair". Other answering pairs $(q_j(\mathbf{x}), P_j)$ are obtained from other peers, where $q_j(\mathbf{x})$ is the rewritten query (intensionally equivalent to the original user's query) for the j-th peer in the network.
Let us denote the list of all answering pairs obtained by a query-rewriting algorithm for a given user query as $\mathcal{L}_{ans}$. Intuitively, such a list contains all the candidate partial answers that are possible to the user query. We denote the partial answer of the i-th peer to the user query as q^{P_i}, by considering that such an answer is a *known* answer for the DIS encapsulated into this peer. The maximal answer $\mathcal{A}_{ns}$ to the user query is the union of partial answers of *all* the answering pairs in the P2P network.
The answering list $\mathcal{L}_{ans}$ is dynamically generated during the answering process by a query-agent, at each step when some peer gives the answers to the rewritten query for it: a query agent can use the network interface of this peer to verify if there is some other peer connected to it is able to answer the user's query. Now we are ready to describe the general query answering algorithm Alg_{P2P} :

Definition 2. *Let K be a finite natural number which limits the partial answers for every peer during the process of query answering of a P2P system (we consider that a peer can be "called" by other peers to answer the query more than one time during the whole query-answering transaction).*

1. *Beginning: impose $\mathcal{A}_{ns} = \{\}$ and $\mathcal{L}_{ans} = (q_k(\boldsymbol{x}), P_k)$, where $q_k(\boldsymbol{x}) = q_C(\boldsymbol{x})$ is the original user query with a tuple of attributes in **x**, specified over the global scheme $\mathcal{G}_{T_k}$ of the initially connected (by user) peer P_k.*
2. *Take the first answering pair $(q_i(\boldsymbol{x}), P_i)$ from $\mathcal{L}_{ans}$, which still didn't answer to the query. If there is no such pair, the answering process ends with the maximal answer in $\mathcal{A}_{ns}$; otherwise the known answers of this peer database are computed and they are added to $\mathcal{A}_{ns}$.*
3. *Consider only the interface group $\mathcal{M}^{ij}_{EXP} = \{(q^{ij}_{1k}, q^{ij}_{2k}) \mid q^{ij}_{1k} \approx q^{ij}_{2k},$ and $1 \leq k \leq \mid ij \mid\}$ of the current peer P_i (toward a peer P_j), if P_i does not present more than K times in the list $\mathcal{L}_{ans}$. If such an interface group does not exist then go to Step 2.*
   Otherwise, build the local mapping system as follows:
   a) *for each intensional query connection $(q^{ij}_{1k}, q^{ij}_{2k}) \in \mathcal{M}^{ij}_{EXP}$, define a predicate r^{ij}_k and two distinct mapping systems $\mathcal{M}_{LAV}$ and $\mathcal{M}_{GAV}$, by inserting the corresponding mappings (**y** is a tuple of attributes for these views):*
   $(q^{ij}_{1k}(\boldsymbol{y}) \Longrightarrow r^{ij}_k(\boldsymbol{y})) \in \mathcal{M}_{LAV}$
   $(q^{ij}_{2k}(\boldsymbol{y}) \Longrightarrow r^{ij}_k(\boldsymbol{y})) \in \mathcal{M}_{GAV}$

   b) *Try to rewrite the query $q_i(\boldsymbol{x})$ (for example by using an algorithm, analogous to the MiniCon [45], using $\mathcal{M}_{LAV}$ mappings) of the current peer into the query*

$q_G(\boldsymbol{x})$, that is composed by the predicate letters r_k^{ij} , $1 \leq k \leq\mid ij \mid$. If it is not possible, go to the next interface group of the current peer (Step 3).

c) Now rewrite (for example by unfolding, using $\mathcal{M}_{GAV}$ mappings) the query $q_G(\boldsymbol{x})$ into the query $q_j(\boldsymbol{x})$ over the logical scheme $\mathcal{G}_{T_j}$ of the peer P_j, and insert this new answering pair $(q_j(\boldsymbol{x}), P_j)$ in to the list $\mathcal{L}_{ans}$ (if it already does not exist in $\mathcal{L}_{ans}$). Go to the next interface group of the current peer (Step 3).

Notice that the algorithm is parameterized by $K = 1, 2...$ and is monotone with respect to K. That is, the number of tuples in the answer to a query grows with the value of K. As a result, each finite value of K will determinate a *sound* query-answer. In order to obtain a *complete* query-answering as well, we must also possibly allow the infinite value for K in the case when the mappings between the peers in a P2P systems are not acyclic. At this point, we assume that in a P2P system with a cyclic P2P mappings there can exist infinite loops during this query answering in the process described by Alg_{P2P}. As a result, if the value of K is infinite, Alg_{P2P} theoretically may not terminate. Due to this reason, in practice, we use only approximative answers to the queries based on this soft computing finite incremental answering algorithm.
We do not exclude the possibility of defining a sound and complete query-answering algorithm for cyclic intensionally based P2P mappings. Such investigation will be explored in the future. In this paper, we consider only finite values for K. Consequently, the sound property of the algorithm Alg_{P2P} is only considered here.

Theorem 1. ***Termination of the answering algorithm:*** *Let $\mathcal{N}$ be a finite P2P network system and q_C be a conjunctive query over a peer P_i. Then, for a given finite value of K, the query answering algorithm Alg_{P2P} terminates and the result of this query is a finite union of certain answers in $\mathcal{A}_{ns}$.*

Notice that given the same initial user-defined conjunctive query q_C over different peers, they will generally generate different maximal answers on a given P2P network system. The answers to a query are topology-dependent.

Incremental query answering: the process of answering the queries can also be controlled by the user. This can be achieved by introducing an option for the user to interrupt the execution of the algorithm at its Step 2. The system (for example, query agent) may present partial results after each peer has answered the original user query. If the user is satisfied with the obtained results, he/she can interrupt the process. In web applications, this termination is usually an important requirement because the time needed to obtain the maximal answers can be considerably long.

5 Embedding P2P into Hybrid Modal Logic

In one way, “hybridization” can be defined as the enrichment of ordinary modal logic with mechanisms for naming and reasoning about individual elements in the domain of a model. Hybrid logic was introduced by Prior [46], the inventor of temporal logic in the mid 1960s, and it played a fundamental role in his philosophical work. The technical development of hybrid logic was initiated by Prior’s student Bull [47]. Bull proved a completeness result for a hybrid logic containing nominals, the $\forall$ and $\exists$ binders, the universal modality and also the path nominals. The subject then seems to have lain dormant for over a decade until hybrid languages were independently reinvented by a group of logicians from Bulgaria [48]. Hybrid logic entered in its current phase in the 1990s, when a new generation of logicians became involved [49].

The metatheoretic considerations in a Gabbay-style rule [50] (see $(**)$ in Appendix) are ”local” ones. We need that some formulae be true at exactly **one** point (i.e., possible world) in any model of ML_{DB} (a particular database). So, if we want to name the points in $\mathcal{W}$ of a model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V})$ and remain in the framework of modal logic, we should find a way of naming the states (i.e., points) using the formulae. We can do this by introducing a second sort of atomic formula namely *nominals*. Syntactically these will be ordinary atomic formulae but they will have an important *semantic* property: nominals will be true at exactly *one* point in any model; nominals “name” this point by being true there and nowhere else.

We also need (for example, for general interpolation results) the kind of close fit between the syntax and semantics in the same signature. That is, we need to be able to name and refer to the states in a signature independent manner. In the first-order logic, this is done with the help of variables and the quantifiers. But an interpolation result based on such an extension would rather be uninteresting. This is because, adding full classical quantification over nominals gives us the power of the first-order correspondence language: the $\forall$ binder is arguably too strong. Any language that employs $\forall$ obscure the locality intuition that is central to Kripke semantics [31]. Fundamental to Kripke semantics is the relativization of semantic evaluation to possible worlds in $\mathcal{W}$. That is, in order to evaluate a modal formula, we need to specify some world $m \in \mathcal{W}$ (the *current* world) and begin the evaluation *there*. The function of the modalities is to scan the worlds accessible from m, the worlds accessible from those worlds, and so on. In brief, m is the starting point in the step-wise local exploration of the model. Languages that allow variables to be bound to *arbitrary* worlds don’t mix well with this intuition.

We would like a general interpolation result in a **restricted** extension that is modally natural. Such an extension has been the focus of attention in the recent work on hybrid logic. The basic idea is this: instead of making use of full classical quantifiers over points, we add a “**local**” binder called $\downarrow$. Such a local language is called $\mathcal{H}(\downarrow, @)$. Now the $\downarrow$ binder (locally) binds variables to the *current* world. In essence, it enables us to create a name for the here-and-now.

The $@$ operator (which does *not* bind variables) is a natural counterpart to $\downarrow$. Whereas $\downarrow$ “stores” the current world (by binding a variable to it), $@$ enables to ”retrieve” the worlds. More precisely, a formula of the form $@_x\varphi$ is an instruction to *move* to the world labeled by the variable x and evaluate φ there.

The key result that will emerge is: $\mathcal{H}(\downarrow, @)$ is not merely local but it is the language which *characterizes* locality. More precisely, $\mathcal{H}(\downarrow, @)$ corresponds to the fragment of first-order logic which is invariant under generated submodels. It is shown that equipping first-order modal logic with $\downarrow$ and $@$ yields systems with Craig interpolation and Beth definability (Kit Fine [51] showed that both Beth’s definability theorem and Craig’s interpolation theorem *fail* for first-order S5 modal logic, without any assumption on the domains). This holds for the logic of any class of frames definable in the bounded fragment of FOL (irrespective of whether constant, expanding, contracting or arbitrary domains are assumed) and makes it possible to blend the first-order Henkin techniques with the use of modal canonical models [52]. It is known that the language $\mathcal{H}(\downarrow, @)$ has an undecidable satisfiability problem. It is also known that its sublanguage $\mathcal{H}(@)$ is decidable and enjoys a weak interpolation [53].

5.1 Hybrid Modal Syntax

Hybrid modal syntax is a standard enrichment of modal logics to obtain a ***Hybrid quantified modal languages*** $\mathcal{H}(\downarrow, @)$. Let $WFF_{ML_{DB}}$ be the well founded formulae of ML_{DB} defined in precedence. Then we define its enriched hybrid extension HML_{DB} in BNF as follows: for any nominal i in the set of nominals NOM and for any variable v (which denotes points) in the set of variables $PVAR$ disjoint from the variables we already have:
$WFF := WFF_{ML_{DB}} \ | \ i \ | \ @_i\varphi \ | \ v \ | \ @_v\varphi \ | \downarrow v.\varphi$
for any nominal i in NOM, we shall call the symbol sequence $@_i$ a satisfaction operator and the symbol $\downarrow$ a "local" binder operator. The difference between nominals and variables is simply this: nominals $i \in NOM$ cannot be bound by $\downarrow$, whereas world variables $v \in PVAR$ can.

The notions of *free* and *bound* world variable, *substitution*, and "world symbol t being substitutable for x in φ " are defined in the same manner that is familiar in the first-order logic, with $\downarrow$ as the only binding operator: we use $\varphi[t/s]$ to denote the formula obtained by replacing all free instances of the world symbol t by the world symbol s. A *sentence* is a formula containing no free object or world variables. The sentences ϕ which are Boolean combination of the form $@_i\varphi$ for $i \in NOM$ are called *closed* terms.

In this way we internalize a relation " $\models$ " into the object language and we shall allow ourselves to form a new kind of formula: given a nominal i , and any formula φ , we can form $i : \phi$. These are the formulae that mirror the " $\models$ " relation; a formula of the form $i : \phi$ will be true at any point in a model if and only if φ is true at the (unique) point that i labels.

5.2 Hybrid Modal Semantics

Nominals name the databases as follows. We introduce the function $\mathcal{V}_N$ from the set NOM (nominals) into a $\mathcal{W}$ such that nominals name the databases (points of $\mathcal{W}$) in the unique way. Nominals "name" the unique point $\mathcal{A}$ (database) at which they are true. The enriched Hybrid model (w.r.t. the standard Kripke model $(\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V})$ defined in Appendix) is $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$. The following terminology will be helpful. In any hybrid model $\mathcal{M}$, for any nominal n , $\mathcal{V}_N(n)$ is a point (database) $\mathcal{A}$. We call the unique point $\mathcal{A} = \mathcal{V}_N(n)$ the denotation of n in $\mathcal{M}$. An assignment for a hybrid model is a function that performs the following: for each nominal variable v it assigns an element of $\mathcal{W}$; for each first-order variable x it assigns an element of $\mathcal{S}$.
Nominals are simply a special sort of propositional symbols constrained in the way they are interpreted. As a result, the following are true:

- $\mathcal{M} \models_{\mathcal{A}} n \quad$ iff $\mathcal{A} = \mathcal{V}_N(n) \quad , \ n \in NOM$
- $\mathcal{M} \models_{\mathcal{A},\, g} @_n q \quad$ iff $\quad \mathcal{M} \models_{\mathcal{A}',\, g} q$ where $\mathcal{A}' = \mathcal{V}_N(n)$

- $\mathcal{M} \models_{\mathcal{A}\, g} v$ iff $\mathcal{A} = g(v)$, $v \in PVAR$
- $\mathcal{M} \models_{\mathcal{A},\, g} @_v q$ iff $\mathcal{M} \models_{\mathcal{A}',\, g} q$ where $\mathcal{A}' = g(v)$
- $\mathcal{M} \models_{\mathcal{A},\, g} \downarrow v.q$ iff $\mathcal{M} \models_{\mathcal{A},\, g^v_{\mathcal{A}}} q$ where $g^v_{\mathcal{A}}$ is the assignment which differs from g only in that $g^v_{\mathcal{A}}(v) = \mathcal{A}$

The intuitive meaning of $\downarrow v.q$ at point $\mathcal{A}$ is best explained dynamically: bind the variable v to the current point $\mathcal{A}$ and continue evaluating q at $\mathcal{A}$ under the new assignment (i.e., the $\downarrow$ creates a name for the "here and now" - it is an intrinsically local binder).

Remark: This hybrid language HML_{DB} contains only familiar modal mechanisms: nominals are atomic formulae; the satisfaction operators and binder operator are actually normal modal operators (they satisfy the Kripke schemata (K). For any nominal i , $@_i(\varphi \Longrightarrow \psi) \Longrightarrow (@_i\varphi \Longrightarrow @_i\psi)$ is valid and they satisfy the Rule of Necessitation (RN): if φ is valid, then $@_i\varphi$ is also valid. Analogously, the same is true for the binder modal operator as well).

Hybrid logics are extensions of traditional modal logics in which it is possible to name points (databases in our case). They improve the behavior of the underlying modal logic (allow us to repair the failures of interpolation in the first-order modal logic [53] in a very general way) and many modally undefinable properties (such as irreflexivity which is important for the database frame of our modal logic) are definable in hybrid logic.

5.3 Model and Semantics for P2P Network System

Let us first define a virtual network database schema $\mathcal{N}_{DB}$ (without integrity constraints) for the given P2P network $\mathcal{N}$ composed by N peers P_i:

$\mathcal{N}_{DB} = \biguplus_{P_i \in \mathcal{N}} \bigcup_{\mathcal{M}^{ij}_{EXP} \in \mathcal{P}_i} \{ r^{ij}_{ik} \mid$ variables of r^{ij} are those used in the query q^{ij}_{1k} and $(q^{ij}_{1k}, q^{ij}_{2k}) \in \mathcal{M}^{ij}_{EXP}\}$
where r^{ij}_{ik} are the new relation symbols (predicates) of this database and " $\biguplus$ " is the disjoint union operation.

The hybrid model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$ for each P2P network system $\mathcal{N}$ is a particular S5 model where $\mathcal{W}$ and $\mathcal{R}$ are defined as follows:

- $\mathcal{W}$ contains an isolated point (the network database) $\mathcal{N}_{DB} = \mathcal{V}_N(n_N)$ (which is the denotation of n_N in $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$) and for each peer $\mathcal{P}_i$ we define a disjoint partition composed by the set of possible worlds (points) $sem^{\mathcal{P}}_i(\mathcal{I}_i) \subseteq \mathcal{W}$.
- $\mathcal{R}$ is a binary relation of "accessibility" on a $\mathcal{W}$, i.e., a subset of $\mathcal{W} \times \mathcal{W}$ defined as follows: for each peer P_i we create a **disjoint partition** $sem^{\mathcal{P}}_i(\mathcal{I}_i) \times sem^{\mathcal{P}}_i(\mathcal{I}_i) \subseteq \mathcal{R}$ of this binary relation.

Notice that the *disjoint composition*, indexed by each peer P_i in $\mathcal{N}$, of the network database $\mathcal{N}_{DB}$ and of the frame w.r.t. the model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$, is a premise for the pure **modular development and maintenance** of the P2P systems:

- **Conservative upgrading**: When we add a new peer, we simply add a new disjoint partition in the frame of the model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$ and we add the new disjoint part of the network database $\mathcal{N}_{DB}$, without any change to the precedent structure.

- **Local updates**: When we modify a peer, we modify only its disjoint partition in the frame of the model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$ and we eventually modify only its disjoint part of the network database $\mathcal{N}_{DB}$.
- **Preserving integrity**: When we eliminate a preexisting peer, we simply eliminate its disjoint partition from the frame of the model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$ and we eliminate its disjoint part from the network database $\mathcal{N}_{DB}$.

Now we are able to translate directly the mappings of a P2P network system $\mathcal{N}$ into the ordinary syntax of this Hybrid modal logic HML_{DB}. For every peer P_i in a P2P network system $\mathcal{N}$, we do as follows:

- **Internal encapsulated mappings**: All assertions of integrity constraints Σ_{T_i}, all assertions of mapping $\mathcal{M}_i$ and all facts (ground terms) of a "local" logical theory of the DIS encapsulated into the database peer P_i, are prefixed by the modal operator $@_i$, where the point (legal database of the peer P_i) $\mathcal{V}_N(i) \in sem_i^{\mathcal{P}}(\mathcal{I}_i)$ is the denotation of i in $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{V}_N)$.
- **Network interface mappings**: For every query-connection $(q_{1k}^{ij}, q_{2k}^{ij}) \in \mathcal{M}_{EXP}^{ij}$, i.e., $q_{1k}^{ij} \approx q_{1k}^{ij}$, we define the pair of closed sentences based on the logical implication:
  $\forall \mathbf{x}(@_i \Box q_{1k}^{ij}(\mathbf{x}) \Longrightarrow @_{n_N} r_{ik}^{ij}(\mathbf{x}))$
  $\forall \mathbf{x}(@_j \Box q_{2k}^{ij}(\mathbf{x}) \Longrightarrow @_{n_N} r_{ik}^{ij}(\mathbf{x}))$
  where $r_{ik}^{ij} \in \mathcal{N}_{DB}$, and **x** is a tuple of (object) variables.

Remark: The translated P2P mappings, from intensional equivalence $\approx$ of the query-connections $(q_{1k}^{ij}, q_{2k}^{ij})$ contained in the peer's ADT's into the logic formulae, are not reachable by traditional GLAV mappings used in the data integration/exchange systems. In fact none of these two paired queries (expressed over two different peers) implicates the other one.

Thus, a P2P system has its *proper P2P mapping semantics*, expressed exclusively by implications from queries expressed over global schemas of peers, to the predicates of the virtual network database $\mathcal{N}_{DB}$. It is important to underline that this network database $\mathcal{N}_{DB}$ **is not** any kind of global ontology of a P2P system and that it is not the object for an user query. Its rule is only a technical one and its parts are locally and dynamically reconstructed by the query answering algorithm only.

Now we are able to give the main result about a general framework for P2P network systems, given by our definition [1].

Theorem 2. ***General Framework for P2P Network Systems:*** *The logical theory of each P2P network system $\mathcal{N}$ can be formalized by closed sentences of the Hybrid sublanguage $\mathcal{H}(@)$ for the S5 normal mono-modal logic, enriched by nominals $n \in NOM$ and by the satisfaction modal operators $@_n$.*

Proof: By the translation of the P2P network system into a hybrid modal logic $\mathcal{H}(\downarrow, @)$, we obtain only the formulas of its sublanguage $\mathcal{H}(@)$ which enjoys weak interpolation and is decidable [53]. $\square$

The consequence of this principal theorem is that the proper hybrid modal HML_{DB} logic for P2P network systems is just the hybrid modal logic $\mathcal{H}(@)$. It is known that the addition of nominals and satisfaction operators increases the expressive power at our disposal in a fundamental way.

Now we are ready to correlate the algorithm Alg_{P2P} with the Hybrid modal semantics of P2P systems:

Theorem 3. *Let $\mathcal{N}$ be a P2P system, q_C be an initial user query specified over a peer P_i, and $q_{P2P} = \bigvee_{n\in\mathcal{N}\ \&\ 1\leq j\leq K} @_n\Box q_{nj}$, where $q_{i1} = q_C$ and q_{nj}, $1 \leq j \leq K$ are the queries, rewritten by the algorithm Alg_{P2P}, for a peer P_n $(n \in \mathcal{N})$. Then the answer of the logical theory, obtained by the hybrid language translation of this P2P in to this query q_{P2P}, is equal to the answer of the initial user query q_C obtained by the algorithm Alg_{P2P}.*

Notice that in this intensional semantics of P2P mappings with a query algorithm Alg_{P2P}, the strong closure relationship between the semantics and the query rewriting algorithm Alg_{P2P} (as in the case of DISs) *does not exist*. Indeed, such a strong closure for a DIS is based on the fact that *there exists* the global schema (with integrity constraints) as a base for the definitions of an user query. The strong closure means that the answers to the algorithmically rewritten queries over the source databases coincide (are equal) with the known tuples (of a type defined by the head of the original user query) in the canonical database of the global schema. That is, the extensions of the original user query and of the rewritten query over the sources (i.e., the data extracted by the wrappers of a given peer database) are *equal* in every possible world (thus, also in the *actual* world) of this intensional Montague's logic.

But different from a single peer (DIS), the well defined global schema of a P2P database system does not exist: P2P system is only the dynamic network of *independent* peers and there is no any centralized management of the global schema of the whole P2P system (the whole P2P system is unknown to any user). As a result, it is not possible to define a query over the entire P2P global schema. Each user query is defined only over a specific data schema *of some particular* peer. Unlike strong closure in DISs, the answer to such user query is bigger than certain (known) tuples in the canonical database of the global schema of query-interrogated peer. The added tuples are obtained from other peers and are determined by intensionally equivalent peer-interconnections and by the K-parameterized algorithm Alg_{P2P}. Thus we have to move from the FOL closure used in the standard DISs to the closure in the intensional modal FOL which must be used for intensionally-based P2P mappings. This is a topic for future investigations.

Here we have another kind of the question: for a given P2P system interconnections, what is the value of K for which we can obtain the maximal possible answer to the given query.

The above theorem implies that given an initial user query q_C over a peer P_i for different values of the parameter K we obtain *different rewritten queries* over the logical theory obtained by the hybrid language translation of this P2P system. The valid tuples (of a type defined by the head of the initial user query q_C) of each rewritten query q_{P2P} are equal to the answer given by the sound algorithm Alg_{P2P}. Thus, there is only a closure between the answering algorithm for the initial user query q_C and the semantics of the *rewritten query* q_{P2P} over the logical theory obtained by the P2P hybrid language translation.

6 Conclusion

Query answering in P2P network systems is an important problem addressed by the research community as well as the industry. As presented earlier, the problem raises a multitude of challenges, ranging from theoretical foundations to practical implications. The algorithms for answering queries using views are already incorporated into a number of data integration systems with integrity constraints. We consider that such techniques can be "locally" used in order to obtain certain answers from single peers. The difficulties basically arise because of the need of dealing with incomplete information in the web. By the encapsulation of database integration system into each peer, we obtain an adequate answer for the web based rich-ontology applications.

We have shown that the nature of *weak* (non invasive) P2P mappings between peers has different semantics w.r.t. the general GLAV mappings in data integration (or data exchange) systems. Such more-complex and intensional-based semantics is a reason for the introduction of the intensional FOL and the hybrid S5 modal logic in order to obtain the general logical theory for the meaning-based P2P network mapping systems.

The modular ADT view for peers is also an adequate solution for other software engineering challenges. For example, effective grid-based query computing.

The P2P design problem is often treated as a problem of *search* through a set of peer configurations. In each configuration, we need to determine whether the workload queries anticipated on some particular peer can be answered using its interface module to other peers, and estimate the cost of the configuration. In particular, this raises to the challenge of developing the *incremental* algorithms for answering the queries.

It is easy to verify that the technique presented in this paper can be adapted to deal with the case of *unions* of conjunctive queries. Finally, we are working on several optimization strategies in the query-rewriting techniques which may be used in order to propagate queries from one peer to the others in a P2P network system. We also investigate the computational complexity of computing the query answers in this context.

The soft computing query answering is the result of an unsatisfactory relationship between the Alg_{P2P} and the semantical formalization. Indeed, the correspondence between the Alg_{P2P} and the proposed semantics is that Alg_{P2P} is sound by not complete. Alg_{P2P} captures only a substantial part of the answer to an user query, by a partial usage of the intensional equivalences defined in the P2P mapping system. Such approximation of query-answering in P2P systems, with weak mappings between peers, is one of the possible pragmatic solutions for the classic FOL undecidability of query answering in web based P2P systems in which there are cyclic mappings between the peers.

References

1. Majkić, Z.: Autoepistemic belief-revision for integration of mutually inconsistent knowledge. In: IICAI 2007, 3rd edn. Indian International Conference on Artificial Intelligence, Pune, India, December 17-19 (2007)
2. Majkić, Z.: Beyond fuzzy: Parameterized approximations of Heyting algebras for uncertain knowledge. In: IICAI 2005, 2nd edn. Indian International Conference on Artificial Intelligence, Pune, India, December 20-22 (2005)

3. Majkić, Z.: Intuitionistic truth-knowledge symmetric bilattices for uncertainty in intelligent systems. In: IS 2006, 3rd edn. Int.IEEE Conf. on Intelligent Systems, London, UK, September 4-6 (2006)
4. Hyperbee, World-wide web (2007), http://www.hyperbee.com
5. Sourceforge, World-wide web (2007), http://freenet.sourceforge.com
6. Napster, World-wide web (2001), http://www.napster.com
7. ICQ, World-wide web (2007), http://www.icq.com
8. Jabber Inc, World-wide web (2007), http://www.jabber.org
9. SETI, World-wide web (2007), http://setiathome.ssl.berkeley.edu
10. Zaihrayeu, I.: Query answering in peer-to-peer database networks. Technical Report DIT-03-012, University of Trento, Italy (2003)
11. Groove, World-wide web (2007), http://office.microsoft.com/en-us/groove
12. Gnutella, World-wide web (2007), http://www.gnutella.com
13. Aberer, K., et al.: Improving data access in P2P systems. IEEE Internet Computing, Los Alamitos (2002)
14. Gribble, S., et al.: What can databases do for Peer-to-Peer? In: WebDB Workshop on Databases and the Web (2001)
15. Serafini, L., et al.: The local relational model: Model and proof theory. Technical Report 0112-23, ITC-IRST (2001)
16. Ghidini, C., Giunchiglia, F.: Local models semantics or contextual reasoning = locality + compatibility. Artificial Intelligence 127, 221–259 (2001)
17. Reiter, R.: Towards a logical reconstruction of relational database theory. In: Brodie, M.L., Mylopoulos, J., Schmidt, J.W. (eds.) On Conceptual Modeling: Perspectives from Artificial Intelligence Databases and Programming Languages (1984)
18. Lenzerini, M., Majkić, Z.: General framework for query reformulation. Semantic Webs and Agents in Integrated Economies, D3.1, IST-2001-34825 (February 2003)
19. Franconi, E., et al.: A robust logical and computational characterization of Peer-to-Peer data systems. Technical Report DIT-03-051, University of Trento, Italy (September 2003)
20. Franconi, E., et al.: A robust logical and computational characterization of Peer-to-Peer data systems. In: Proc. of the Int. Workshop On Databases, Inf.Systems and P2P Computing, Berlin, Germany (September 2003)
21. Calvanese, D., et al.: Semantic data integration in P2P systems. In: Proc. of the Int. Workshop On Databases, Inf.Systems and P2P Computing, Berlin, Germany (September 2003)
22. Majkić, Z.: Flexible intentional query-answering for RDF Peer-to-Peer systems. In: Larsen, H.L., et al. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 7–10. Springer, Heidelberg (2006)
23. Calvanese, D., et al.: Logical foundations of Peer-to-Peer data integration, PODS 2004, June 14-16, Paris, France (2004)
24. Calvanese, D., et al.: Inconsistency tollerance in P2P data integration: an epistemic approach. In: Proc. 10th Int. Workshop on Database Programming Language (2005)
25. Majkić, Z.: Intensional semantics for P2P data integration. LNCS Journal on Data Semantics VI, Special Issue on Emergent Semantics (April 15, 2006)
26. Majkić, Z.: Weakly-coupled P2P system with a network repository. In: WDAS 2004. 6th Workshop on Distributed Data and Structures, Lausanne, Switzerland, July 5-7 (2004)
27. Majkić, Z.: Massive parallelism for query answering in weakly integrated P2P systems. In: Workshop GLOBE 2004, Zaragoza, Spain, August 30-September 3 (2004)
28. Montague, R.: Formal philosophy. selected papers of Richard Montague. In: Thomason, R. (ed.), Yale University Press, New Haven, London, pp. 108–221 (1974)

29. Majkić, Z.: Non omniscient intensional contextual reasoning for query-agents in P2P systems. In: IICAI 2007. 3rd Indian International Conference on Artificial Intelligence, Pune, India, December 17-19 (2007)
30. Kripke, S.A.: A completeness theorem in modal logic. The Journal of Symbolic Logic 24, 1–14 (1959)
31. Kripke, S.A.: Semantic analisys of modal logic. The Journal of Symbolic Logic 24, 323–324 (1959)
32. Calì, A., et al.: Data integration under integrity constraints, pp. 262–279 (2002)
33. Lenzerini, M.: Data integration: A theoretical perspective, pp. 233–246 (2002)
34. Alon, Y.: Answering queries using views: A survey. 10(4), 270–294 (2001)
35. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases (1995)
36. Fagin, R., et al.: DATA Exchange: Semantics and query answering (2003)
37. Majkić, Z.: Querying with negation in data integration systems. In: IDEAS. 9th International Database Engineering and Application Symposium, Montreal, Canada, July 25-27, 2005, pp. 58–70. IEEE Computer Society Press, Los Alamitos (2005)
38. Johnson, D.S., Klug, A.C.: Testing containment of conjunctive queries under functional and inclusion dependencies. Journal of Computer and System Sciences 28(1), 167–189 (1984)
39. Reiter, R.: On closed world data bases. In: Gallaire, H., Minker, J. (eds.) Logic and Databases, pp. 119–140 (1978)
40. Majkić, Z.: Fixpoint semantics for query answering in data integration systems. In: AGP03 - 8.th Joint Conference on Declarative Programming, Reggio Calabria, pp. 135–146 (2003)
41. Reiter, R.: What should a database know? 14, 127–153 (1990)
42. Majkić, Z.: Weakly-coupled ontology integration of P2P database systems. In: 1st Int. Workshop on Peer-to-Peer Knowledge Management (P2PKM), Boston, USA, August 22 (2004)
43. Majkić, Z.: Intensional logic and epistemic independency of intelligent database agents. In: WSPI 2005. 2nd International Workshop on Philosophy and Informatics, Kaiserslautern, Germany, April 10-13 (2005)
44. Levesque, H.J.: All I know: A study in autoepistemic logic 42, 263–310 (1990)
45. Halevy, A.Y.: Theory of answering queries using views 29(4), 40–47 (2000)
46. Prior, A.: Past, Present, and Future (1967)
47. Bull, R.: An approach to tense logic. Theoria 12, 171–182 (1970)
48. Passy, S., Tinchev, T.: An essay in combinatory dynamic logic. 93, 263–332 (1991)
49. Blackburn, P.: Representation, reasoning, and relational structures: A hybrid logic manifesto. Methods for Modalities 1, Logic Journal of the IGPL 8, 339–365 (2000)
50. Gabbay, D.: An irreflexivity lemma. In: Monnich, U. (ed.) Aspects of Philosophical Logic, Riedel, pp. 67–89 (1981)
51. Fine, K.: Failures of the interpolation lemma in quantified modal logic. Journal of Symbolic Logic 44, 201–206 (1979)
52. Arces, C., Blackburn, P., Marx, M.: Reparing the interpolation theorem in quantified modal logic (2001), available at http://www.hylo.net
53. Areces, C., Blackburn, P., Marx, M.: Hybrid logics: Characterization, interpolation and complexity. Tecn.Rep. CLAUS-Report 104, Universitat des Saarlandes (1999)

A APPENDIX: Standard Mono-modal Logic and P2P Database Theory with Strong Implication-Based Mapping between Peers

From a logical point of view, we consider a database $\mathcal{A}$ as a logical theory (formalized by the First Order Data Base Language ($FODBL$): the set of first-order formulae written in the relational language $FODBL$ for a given set of basic predicates (database relations). The concepts of the schema and integrity constraints are borrowed without modification from the classical definition which comprises of attribute domains, relations (schema + value) and integrity constraints (they express a condition on the extension of the database), where $\mathcal{A}$ is composed (as in, for example, Deductive databases) by an **extensional** database, denoted by $\mathcal{A}_E$ (only ground terms), and an **intensional** database denoted by $\mathcal{A}_I$, so that:

- the extensional database is the set of ground instances of atoms defining the extensions of the *basic predicates* (that is, a relational database instance);
- the intensional database is the set of deduction rules that defines the virtual predicates and integrity constraints. We denote the set of all models of $\mathcal{A}$ which satisfy this set of integrity constraints (it is a non empty set iff $\mathcal{A}$ is consistent database) by $Sem_{\mathcal{A}}$.

If $\mathcal{A}$ is a DIS then $\mathcal{A}_E$ is its retrieved global database, while $\mathcal{A}_I$ is the set of integrity constraints. We formally introduce the syntax and the semantics for a Modal Logic for Databases (ML_{DB}) as follows:

A.1 Syntax

- All formation rules of the subset of classical predicate logic, composed by only conjunctive relational queries with union, are also the formation rules of this mono-modal logic.
- If $\varphi(\mathbf{x})$ is a formula then $\Box\varphi(\mathbf{x})$ (“it ought to be that $\varphi(\mathbf{x})$”) and $\Diamond\varphi(\mathbf{x})$ (“it can be that $\varphi(\mathbf{x})$”), where $\Diamond \triangleq \neg\Box\neg$, are formulae of this modal logic.

A.2 Model and Semantics

We consider a model $\mathcal{M}$ for this modal logic as a quadruple $(\mathcal{W}, \mathcal{R}, \mathcal{S}, \mathcal{V})$ where:

- $\mathcal{W}$ is the union of $Sem_{\mathcal{A}}$ for each $\mathcal{A}$ in a given set of databases. Each element in $\mathcal{W}$ is also called as “points” or (possible) worlds.
- $\mathcal{R}$ is a binary relation of “accessibility” on a $\mathcal{W}$, i.e., a subset of $\mathcal{W} \times \mathcal{W}$, defined as follows:
  for each consistent database $\mathcal{A}$ we create a **disjoint partition** $Sem_{\mathcal{A}} \times Sem_{\mathcal{A}} \subseteq \mathcal{R}$ of this binary relation. In this way the binary relation of each partition is reflexive, symmetric and transitive. Thus, this is a S5 modal logic as presented by Levesque [50]. The only difference is that $\mathcal{R}$ is not an unique relation as in the standard S5 modal logic but it is the collection of disjoint partitions.
- $\mathcal{S}$ is a non-empty domain of individuals (union of the domains of all attributes in a given database universe).
- $\mathcal{V}$ is a function that assigns to each pair, that consists of a n-place predicate constant r and of an element $\mathcal{A} \in \mathcal{W}$, a function $\mathcal{V}(r, \mathcal{A})$ from S^n to $\{1, 0\}$.

The point is: for example, some predicates might be true over some different sets of objects in different points. The semantics of the formulae of this modal logic is described by means of the notation $\mathcal{M} \models_{\mathcal{A},\ g} q$ with the meaning: “q is true in the model $\mathcal{M}$ at the point A and for the assignment function g”.

If q is an expression of this modal logic language, we shall represent the semantic value of q in the model $\mathcal{M}$ at the point A and for the assignment function g by the notation $[q]^{\mathcal{M},\mathcal{A},g}$. Thus we have the following equivalence that holds for any formula q of the language: $\mathcal{M} \vDash_{\mathcal{A},\, g} q \iff ([q]^{\mathcal{M},\, \mathcal{A},\, g})$.
If r is a predicate constant we have $[r]^{\mathcal{M},\, \mathcal{A},\, g} = \mathcal{V}(r, \mathcal{A})$.

The interpretation of the formulae of this modal logic is then given by the semantic rules of the classical predicate language (without functional symbols), where g is replaced by $(\mathcal{A}, g)$ everywhere. The additional semantic rules relative to the modal operators $\Box$ and $\Diamond$ are as follows:

- $\mathcal{M} \vDash_{\mathcal{A},\, g} \Box q$ iff $\mathcal{M} \vDash_{\mathcal{A}',\, g} q$ for every $\mathcal{A}'$ in $\mathcal{W}$ such that $(\mathcal{A}, \mathcal{A}') \in \mathcal{R}$.
- $\mathcal{M} \vDash_{\mathcal{A},\, g} \Diamond q$ iff there exists a $\mathcal{A}'$ in $\mathcal{W}$ such that $(\mathcal{A}, \mathcal{A}') \in \mathcal{R}$ and $\mathcal{M} \vDash_{\mathcal{A}',\, g} q$.

A formula q is said to be *true in a model* $\mathcal{M}$ if $\mathcal{M} \vDash_{\mathcal{A},\, g} q$ for each assignment function g and a point $\mathcal{A} \in \mathcal{W}$. A formula is said to be *valid* if it is true in each model. From the definition of the frame $(\mathcal{W}, \mathcal{R})$ we can see that it is a disjoint union of a strongly connected S5 partitions for each database $\mathcal{A}$.

A.3 Axiom System

Thus, the obtained mono-modal logic is a *smallest normal* S5 (also known as KT45) modal logic with the following axiom system (Σ_M, R_M) :

- The set of axioms Σ_M is composed by the subset of axioms of the classical predicate logic (without functional symbols) and modal axioms):
  Kripke schemata (K) $\Box(a \Longrightarrow b) \Longrightarrow (\Box a \Longrightarrow \Box b)$
- The inference rules R_M are:
  Modus Ponens (MP) if $a, a \Longrightarrow b$ then b
  Substitution rule (US) if a then a' (a' is a substitution instance of a)
  Rule of necessitation (RN) if a then $\Box a$

Let $\mathcal{A}$ and $\mathcal{B}$ be any two databases, φ_A be a set of all integrity constraints for the database $\mathcal{A}$ and ϕ_B a set of all integrity constraints for the database $\mathcal{B}$. Thus, in a modal logic ML_{DB}, for any assignment function g:

$$\mathcal{M} \vDash_{\mathcal{A},\, g} \varphi_A \quad and \quad \mathcal{M} \vDash_{\mathcal{B},\, g} \phi_B$$

Any standard strong mapping, that is based on logic implications between the peers knowledge and uses the known answers from $\mathcal{A}$ to $\mathcal{B}$, can be expressed in the following way (in a Gabbay-style rule) [50]:

- (**) " $\mathcal{M} \vDash_{\mathcal{A},\, g} \Box q_A(\mathbf{x})$ implies $\mathcal{M} \vDash_{\mathcal{B},\, g} \Box q_B(\mathbf{x})$ "

This mapping is a metatheoretic consideration. I.e., it is a formula of neither the modal logic ML_{DB} nor the $FODBL$. Thus, in order to define the mappings between databases, based on the known answers of these databases, we need a "hybrid" extension [49] of the ordinary modal logic.

Notice that instead of the strong mappings based on logic implication between the (known) views of peers (expressed by (**)), we use the weak mapping between the peers for our epistemically *independent* peers. Here, $\Box q_A(\mathbf{x})$ of the first peer $\mathcal{A}$ is *intensionally equivalent* to $\Box q_B(\mathbf{x})$ of the second peer $\mathcal{B}$.

Power Distribution System Fault Diagnosis Using Hybrid Algorithm of Fuzzy Classification and Artificial Immune Systems*

Le Xu[1] and Mo-Yuen Chow[2]

[1] Quanta Technology LLC, 4020 Westchase Blvd, Suite 300, Raleigh, NC 27607, USA
[2] Dept. of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695, USA

1 Introduction

As a vital lifeline of the modern society for maintaining adequate and reliable flows of energy, power distribution systems deliver the electricity from high voltage transmission circuits to customers. Any interruption in their service may cause economical loss, damage equipments, and even endanger people lives. When a power outage (i.e., the loss of the electricity supply to an area) occurs, it is of great importance to diagnose the fault and restore the system in a timely manner in order to maintain the system availability and reliability. However, the restoration process may take from tens of minutes to hours. Most utilities for safety concerns do not restore the system until the outage cause has been found: linemen may need to inspect the distribution lines section by section in an attempt to find evidences (e.g., burn marks on the pole for possible lightning caused faults, dead animal bodies for possible animal activity related faults) and to ensure safety prior to re-energizing the system (e.g., no down distribution lines). Sometimes specific crew need to be further dispatched for advanced tasks such as the removal of fallen trees. Effective identification of either the cause or the location of the outage can provide valuable information to expedite the restoration procedure.

This work has been focused on fault diagnosis which helps to narrow down the aspects that have to be examined and to better schedule crew activities. For example, the dispatch center can inform the linemen to focus on certain types of outage causes or dispatch appropriate crew earlier to the outage locations in need. Fault diagnosis in many cases can be considered as a classification task in that the cause of a fault under investigation is to be categorized into one of existing cause classes.

There have been a large number of researches on various classification technologies. Empirical studies show that many classification algorithms have sufficiently similar accuracies on well-processed data such that the differences between those algorithms are statistically insignificant (Han and Kamber 2001). It may not be true when data imperfection issues are involved. One such example is the data imbalance (i.e., at least one of the classes significantly outnumbers some others) often seen in real-world applications including power distribution system fault diagnosis. For

* This work is supported by National Science Foundation Grant ECS-0245383.

instance, when we analyze healthy and faulty conditions of a power system, we will have much more healthy data than faulty data simply because power systems are operational most of the time. Even for the power system faulty conditions, some outages are rare events (e.g., tornado caused power outages in North Carolina), whose records are much fewer than those of major outage causes such as trees.

Most of the commonly used classification algorithms aim to minimize the overall error rate. When imbalanced data are presented, these algorithms inevitably tend to prioritize different classes in favor of the class with more data in order to achieve a high overall accuracy. This biased favor may sacrifice the performance on identifying minority class, and achieve a high accuracy on the majority class but a very low, sometimes unacceptable, accuracy on the minority class.

In authors' previous works, the data mining based fuzzy classification algorithm (Ishibuchi and Yamamoto 2002) has been extended to *E*-algorithm (Xu et al. 2006) for imbalanced data; the immune system inspired algorithm, Artificial Immune Recognition System (AIRS) (Watkins 2001, 2002, Watkins and Timmis 2002), has been investigated for its capability for imbalanced data classification (Xu et al. 2006). Both algorithms demonstrate good capabilities in classifying imbalanced data comparing with the widely used feed forward neural network (Xu and Chow 2006). *E*-algorithm can produce linguistic rules to explain the reasoning of fault cause identification, but it is computational demanding. AIRS can quickly generate a small set of non-redundant patterns representing the training data through immunology mechanisms; however, it does not present its reasoning directly. In order to utilize the searching mechanism of AIRS and the rule generation capability of *E*-algorithm, a hybrid algorithm for power distribution system outage cause identification, named as Fuzzy Artificial Immune Recognition System (FAIRS), is proposed in this work.

This paper starts with a brief introduction of *E*-algorithm and AIRS, followed by the description of the hybrid algorithm FAIRS. The application of the FAIRS to power distribution system fault diagnosis is then presented, with its performance compared with its two base algorithms, *E*-algorithm and AIRS.

2 *E*-Algorithm

E-algorithm is extended from a data mining based fuzzy classification method (Ishibuchi and Yamamoto 2002) for imbalanced data. *E*-algorithm utilizes the statistical information revealed by two data mining concepts, *support* and *confidence* (Ishibuchi and Yamamoto 2002), to determine fuzzy set partitions and select fuzzy rules. By this mean, necessary information is extracted from the data so as to fulfill classification tasks even without sufficient domain knowledge available a priori.

Since fuzzy algorithms deal with approximate reasoning rather than precisely deduction, compatibility grade is used to represent the matching degree of a specific case with a fuzzy rule. For a fuzzy rule expressed as Eq. (2.1), the compatibility grade of a data sample with a fuzzy rule is expressed as in Eq. (2.2)

$$\mathbf{A}_k \Rightarrow B_k \tag{2.1}$$

where $\mathbf{A}_k=(A_{1,k}, \ldots, A_{m,k})$; $A_{i,k}$, $i=1,\ldots,m$, is the fuzzy set for the i^{th} input attribute x_i in rule R_k, $k=1,\ldots,K$; K is the number of rules; B_k is the fuzzy set for the output attribute y in rule R_k (only one output attribute is considered in this paper).

$$\mu_{\mathbf{A}_k}(\mathbf{x}_l)=\mu_{A_{1,k}}(x_{1,l})\times\cdots\times\mu_{A_{m,k}}(x_{m,l}) \tag{2.2}$$

where $\mathbf{x}_l=(x_{1,l},\ldots, x_{m,l})$ is the l^{th} data sample, $x_{i,l}$ is the i^{th} data attribute, $l=1,\ldots,N$, N is the number of data samples; $\mu_{A_{i,k}}(\cdot)$ represents the membership function of the antecedent fuzzy set $A_{i,k}$ of rule R_k.

Support as shown in Eq. (2.3) indicates the magnitude of the total compatibility grade of a fuzzy rule R_k with all the data from the rule's corresponding class B_k; *confidence* as shown in Eq. (2.4) presents the proportion of the total compatibility grade of a fuzzy rule R_k with all the data from its corresponding class B_k. Different from Ishibuchi's original algorithm (Ishibuchi and Yamamoto 2002), the number of data in different classes, expressed as N_{B_k}, has been incorporated into the equations in order to take the data imbalance issue into consideration.

$$s(\mathbf{A}_k \Rightarrow B_k)=\frac{\dfrac{\sum_{l\in B_k}\mu_{\mathbf{A}_k}(\mathbf{x}_l)}{N_{B_k}/N}}{N} \tag{2.3}$$

$$c(\mathbf{A}_k \Rightarrow B_k)=\frac{\dfrac{\sum_{l\in B_k}\mu_{\mathbf{A}_k}(\mathbf{x}_l)}{N_{B_k}/N}}{\sum_{l=1}^{N}\mu_{\mathbf{A}_k}(\mathbf{x}_l)} \tag{2.4}$$

Four fuzzy set partitions for each attribute, as shown in Fig.1, are simultaneously used in the algorithm such that fuzzy rules can be generated even without appropriate fuzzy sets determined from domain knowledge a priori (Ishibuchi and Yamamoto 2002). With this approach, each antecedent attribute is initially associated with 15 fuzzy sets (14 sets generated by these 4 partitions as well as a "don't care" set).

E-algorithm firstly enumerates all possible combinations of antecedent fuzzy sets $\mathbf{A}_k=(A_{1,k}, \ldots, A_{m,k})$ and assigns a consequent part B_k to each combination. B_k is the class achieving the maximum *confidence* value given the combination:

$$B_k=\arg\max_{p\in\{1,2,\cdots,M\}} c(\mathbf{A}_k \Rightarrow p) \tag{2.5}$$

where M is the number of classes.

The product of *support* and *confidence* is then used as a measure to select N_s (a user-defined number) best rules from each class and form a fuzzy classification rule base S, which is responsible to make final classification decisions. For any test data sample $\mathbf{x}_r$, the best matching rule from the rule base S is identified in order to

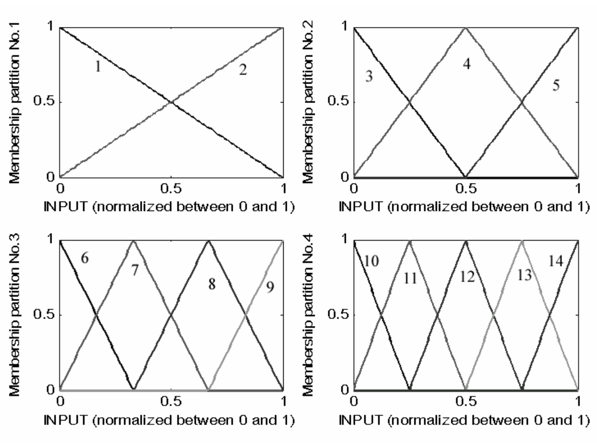


Fig. 1. Four fuzzy partitions for each attribute membership function

determine the category to which $\mathbf{x}_r$ belongs. The selection makes use of single winner rule method (Ishibuchi et al. 1999) which employs the product of the compatibility grade $\mu_{A_k}(\mathbf{x}_r)$ and the certainty grade of k^{th} rule, CF_k, as the criterion

$$CF_k = c(\mathbf{A}_k \Rightarrow B_k) - c_{\text{sec}} \tag{2.6}$$

where c_{sec} is the second largest *confidence* value given $\mathbf{A}_k$; for a two-class problem, c_{sec} is simply the *confidence* value for the other class given $\mathbf{A}_k$.

The data mining based fuzzy classification *E*-algorithm can fulfill classification tasks when sufficient domain knowledge is not available for determining fuzzy set and generating fuzzy rule; however, this algorithm is computational demanding because of the enumeration of all possible antecedent fuzzy set combinations, which is the key step to prevent the requirement for a priori knowledge. The total number of possible fuzzy set combinations for *m* attributes increases exponentially with the attribute number *m*. As a result, the rule length in this work is limited to three antecedent attributes in order to reduce the computational demand; which is also a common practice in Ishibuchi *et al.*'s work (Ishibuchi and Yamamoto 2002).

3 Artificial Immune Recognition Systems (AIRS)

Immune systems guard our bodies against infections due to pathogens through complicated mechanisms. AIRS is a biological immune system inspired classification algorithm. The algorithm related aspects of immunology are briefly introduced in this section; more detailed overview of immune systems can be found in various references such as (Castro and Timmis 2002).

3.1 Natural Immune Systems

When antigens invade our bodies, the surface receptors of different B-cells (one kind of lymphocyte) are able to recognize specific antigens through complementary pattern matching, called *binding*. The stronger the binding is, the higher the receptor-antigen affinity is. The recognitions activate the hosting B-cells; and then the activated B-cells start to proliferate. Some of the clones differentiate into plasma cells that secrete large amount of antibodies to neutralize the pathogens; while some clones experience somatic hypermutation to create diverse antibodies such that the immune system can adapt its response to new threats.

Antibodies have to compete for limited resources. The ones fitting the antigens better survive. Some survived antibodies become memory cells and remain in the system for a long period such that the system response to the same or similar antigens in the future can be improved in terms of speed and accuracy.

3.2 AIRS Algorithm

AIRS is based on B-cell mechanism. As a classification algorithm, the input data to be categorized are considered as antigens, and different class patterns are expressed as antibodies. Both antigens and antibodies are usually represented as vectors. The Euclidean distance between antigens and antibodies are utilized to measure the affinity of the antigen-antibody binding.

In the algorithm, antibodies are included within B-cells, which contain additional information such as the resource assigned to antibodies. A set of identical B-cells is represented as an artificial recognition ball (*ARB*) in order to reduce duplication (Watkins et al. 2004).

During the initialization phase, AIRS randomly selects a certain number of training data to form an initial set of *ARB*s, S_{ARB}, and an initial set of memory cells, S_{MC}, for each class.

The training phase of AIRS is a one-shot incremental procedure. For each training antigen, the best matching cell is first identified from the current memory cell population S_{MC} of the antigen's class and denoted as mc_{match}. This mc_{match} gets a chance to generate offspring and expand its affiliated S_{ARB} through clonal expansion and somatic hypermutation. Each *ARB* in the updated S_{ARB} is examined for its affinity with the antigen and assigned a finite amount of resources accordingly. The total system resource available is considered as constant, the weakest *ARB*s (i.e., has low affinity value with antigens) are eliminated until the total resource limitation is met. The survived *ARB*s further produce offspring through immunology mechanisms (clonal expansion and somatic hypermutation) and the newly generated *ARB*s compete for the limit resource likewise until the average affinity value for all the *ARB*s in this current S_{ARB} reaches a pre-set threshold (or other user defined stopping criteria are satisfied). The best matching *ARB* is then nominated as a memory cell candidate for its affiliated class, denoted as $mc_{candidate}$. This $mc_{candidate}$ will only be added to S_{MC} and become a long-lived memory cell if it matches the antigen better than mc_{match} already does.

This procedure repeats for all the training antigens. The final memory cell sets generated during the training phase serve as the class prototypes, by which the class of any test antigen is determined through *k*-Nearest Neighbor (*k*-NN) approach.

Through immunology mechanisms and resource competition AIRS can quickly search for a small set of class prototypes for classification tasks. However, since the class patterns are presented as antibodies (usually in vector forms), AIRS is lack of the capability of directly presenting its reasoning for classification as *E*-algorithm does.

4 Fuzzy Artificial Immune Recognition System (FAIRS)

In order to take advantage of the strengths of both *E*-algorithm and AIRS and suppress their weakness, a hybrid algorithm, Fuzzy Artificial Immune Recognition System (FAIRS), has been developed. FAIRS aims to quickly develop inference rules (with sufficient flexibility in rule length) for classification tasks, based on the way how *E*-algorithm extracts rules using the statistical information revealed by data mining measures and the way how AIRS searches for representative memory cells.

4.1 Antigen and Antibody

Same as in AIRS, the input data to be categorized are considered as antigens, denoted as **Ag**:

$$\mathbf{Ag} = [Ag_1, \cdots, Ag_r] \tag{4.1}$$

where Ag_i, i=1,..., r is the i^{th} gene of the antigen (corresponding to the i^{th} input attribute), r is the number of genes in an antigen (i.e., the number of input attributes).

In AIRS, antibodies take on the same vector format as antigens in order to calculate the affinity of antigen-antibody binding, which is the Euclidean distance. In FAIRS, antibodies **Ab** represent fuzzy classification rules; therefore a different format is used:

$$\mathbf{Ab} = [I_1, \cdots, I_r \vdots \mathbf{A}_1, ..., \mathbf{A}_r] \tag{4.2}$$

where $I_i \in \{0,1\}$, i=1,..., r, is the i^{th} antecedent attribute inclusion indicator; $\mathbf{A}_i$ represents the sequence number of the fuzzy set to which the i^{th} antecedent attribute belongs.

The antecedent attribute inclusion indicator I_i represents the attribute status in a rule: 1 indicates the inclusion of the i^{th} gene Ag_i (or the i^{th} antecedent attribute) and 0 indicates its exclusion (i.e., the "don't care" case). In order to determine the fuzzy set partition, the "simultaneous-four-fuzzy-set-partition" approach is utilized. Since the use of antecedent attribute inclusion indicator embeds one possible fuzzy set ("don't care"), each attribute of a fuzzy rule is only associated with 14 fuzzy sets here instead of 15 sets as in *E*-algorithm. A four-bit binary array $\mathbf{A}_i$ between 0001 and 1110 can present any value from 1 to 14, it is utilized to represent the 14 possible fuzzy sets for each attribute; in other words, each $\mathbf{A}_i$ corresponds to a fuzzy set whose sequence number shown in Fig.1 is the same as the decimal value of $\mathbf{A}_i$.

The antigen-antibody affinity A_f is the compatibility grade of a fuzzy rule with a data sample, calculated as the product of the memberships of an included attribute to the corresponding fuzzy set as shown in Eq. (4.3):

$$A_f(\mathbf{Ab}, \mathbf{Ag}) = \prod_{i=\{1,\cdots,r\}, I_i \neq 0} \mu_a(Ag_i) \tag{4.3}$$

$$a = dec(\mathbf{A}_i) \tag{4.4}$$

where $\mu_a(\cdot)$ represents the membership function of a^{th} antecedent fuzzy set, a is the decimal value of $\mathbf{A}_i$ calculated by the conversion function $dec(\cdot)$.

4.2 Algorithm

Due to the change of antibody format, the S_{ARB} and the S_{MC} cannot be initialized by randomly selecting training data as antibodies. Instead, both S_{ARB} and S_{MC} are initially set as empty and to be expanded during the training.

The incremental training procedure of FAIRS is similar to AIRS.

Each class of data has its own S_{ARB} and S_{MC}; for example, $S_{ARB.c}$ and $S_{MC.c}$ are the ARB set and the memory cell set for class c, respectively. Given a specific training antigen **Ag** from class c, the best matching cell mc_{match} from its own memory cell set $S_{MC.c}$ is identified as in Eq. (4.5). If $S_{MC.c} \equiv \varnothing$, then FAIRS randomly generates an antibody as mc_{match} with class c.

$$mc_{match} = \arg\max_i \left[A_f(mc_i, \mathbf{Ag}) \right] \tag{4.5}$$

where mc_i, i=1,...,T, refers to the i^{th} memory cell in $S_{MC.c}$, T is the number of memory cells included in the current $S_{MC.c}$.

The mc_{match} generates new ARBs to expand $S_{ARB.c}$ through clonal expansion and somatic hypermutation as in AIRS. The number of new ARBs that each mc_{match} can generate is proportional to its affinity with **Ag**.

$$N_{clone} = \alpha \cdot A_f(mc_{match}, \mathbf{Ag}) \tag{4.6}$$

where N_{clone} is the number of new ARBs to be generated; α is the clonal rate.

The somatic hypermutation is currently based on random mechanism. Any binary digit in **Ab** may be converted from 1/0 to 0/1 with certain mutation rate.

Each ARB is allocated a finite number of resources based on its affinity with **Ag**. If the total resources allocated exceed the system capability, the weakest ARBs will be eliminated until the total resource allocated meets the system limit.

The expansion of ARB set $S_{ARB.c}$ and the competition of resources repeat until the average ARB-antigen affinity value over the entire ARB set reaches a pre-set threshold β:

$$\frac{\sum_{ARB_j \in S_{ARB}} A_f(ARB_j, \mathbf{Ag})}{|S_{ARB}|} \geq \beta \tag{4.7}$$

The best ARB from $S_{ARB.c}$ is nominated as the $mc_{candidate}$ and is added to $S_{MC.c}$ if its affinity with **Ag** is higher than the mc_{match}.

After all the antigens are presented, *support* and *confidence* of every generated rule (i.e., memory cells in S_{MC}) with every training antigen are calculated using Eq. (2.3) and Eq. (2.4). Different from E-algorithm that chooses the final rules based on calculated *support* and *confidence* of the entire possible fuzzy rule combination, FAIRS only calculates *support* and *confidence* for fuzzy rules already chosen. This approach significantly reduces the computing time but may cause conflicts between individually selected rules and statistically supported rules. For instance, a class-A rule developed during the training phase actually has measures *support* and *confidence* in favor of class B, which means the certainty grade CF for this class-A rule as calculated in

Eq. (2.6) is negative since $c(\mathbf{A} \Rightarrow B) > c(\mathbf{A} \Rightarrow A)$. In this instance, the individually selected rules will be deleted, giving way to the statistically supported rules.

The final S_{MC} contains all the fuzzy classification rules for different classes. For every test antigen, the rule with the maximum product of compatibility grade and certainty grade becomes the winner rule to determine the class of the test data. This step is the same as in the *E*-algorithm.

The process of FAIRS for one antigen is summarized as a brief flowchart in Fig 2.

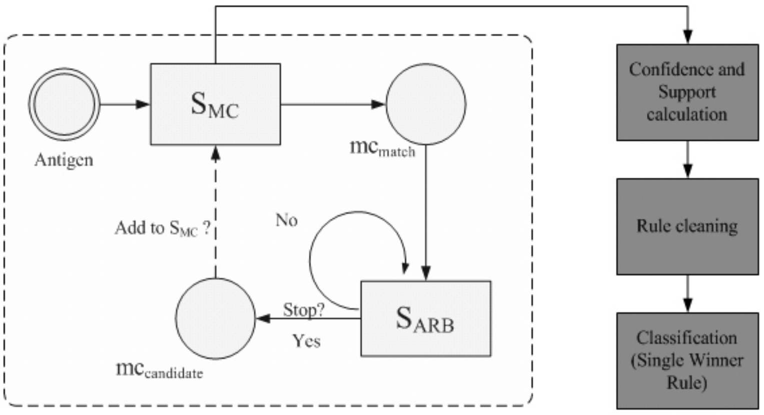


Fig. 2. Brief flowchart of FAIRS

5 Power Distribution Outage Data

FAIRS algorithm has been applied to Duke Energy distribution outage data from 1994 to 2002 for fault cause identification in order to examine its performance.

Every time an outage is detected in the system as a result of the activation of protective devices (e.g. a circuit breaker, a fuse); the relative information is recorded into the data collection system. Each outage record contains 33 information fields, of which six are considered as the most essential and influential factors based on domain experts' suggestions as well as the statistical significance tests (Xu et al. 2003). These six influential factors are: *circuit ID, weather, season, time of day, phases affected*, and *protective devices activated*. The attribute *cause* entered by crew after finding out the actual outage cause is the class label. Among all the possible causes, three top customer interruption causes for most utilities (Brown 2002) including Duke Energy: animal contact, tree, and lightning strike, are used as prototypes.

All these six influence factors are categorical variables; they are transformed into numerical variables using likelihood measures (Chow and Taylor 1995) so that they can be passed on to the algorithm. The likelihood measure shown in Eq. (5.1) represents the conditional probability of an outage caused by a specific fault given certain condition (e.g., an outage caused by tree given icy weather condition).

$$L_{i,j} = \frac{N_{i,j}}{N_j} \tag{5.1}$$

where *i* refers to outage cause (e.g., tree, lightning), *j* refers to outage related event or condition (e.g. lightning weather, fuse activated, morning), $N_{i,j}$ is the number of outages caused by fault *i* under condition *j*, N_j is the number of outages under condition *j*, and $L_{i,j}$ is the likelihood measure of fault *i* given condition *j*.

Likelihood measures provide useful information for fault cause identification; they are logically used as the algorithm inputs (i.e., the element of antigen vectors). As shown in Eq. (5.1), the likelihood measure depends on both fault type *i* and influential condition *j*; different likelihood values are calculated with respect to different fault causes even under the same influential condition. As a result, FAIRS algorithms always process one outage cause at a time. For example, the algorithms aim to distinguish tree caused faults from non-tree caused faults (which can be caused by animal, lightning, or others).

Based on Duke Energy senior distribution engineers' suggestions, seven of Duke Energy's thirty-two service regions in North Carolina and South Carolina are selected as representations considering geographical features and system status: Chapel Hill (CH), Clemson (CS), Durham (DH), Greenville (GV), Hickory (HC), Lancaster (LC), and Winston-Salem (WS). These seven regions cover metropolitan areas, cities, towns, rural areas and wooded areas, and also embody both old systems and new systems.

6 Result and Discussions

6.1 Performance Measures

The performance of FAIRS is compared with its two base algorithms: *E*-algorithm and AIRS. The commonly used performance measure for classification algorithm checks the overall accuracy, i.e., the percentage of correctly classified cases. This measure can be misleading when the data are imbalanced. For instance, the majority class *M* in a two-class imbalanced data set *Q* contains 95% of the data and the minority class *N* contains the remaining 5% data; a classifier that indiscriminantly categorizes every individual case as the majority class *M* still achieves an overall accuracy as high as 95%. This is certainly undesirable from the classification viewpoint. Kubat *et al.* have proposed the g-mean (Kubat et al. 1998) to measure the imbalanced data classification performance. This measure is based on the confusion matrix as shown in Table 1.

Assuming tree/animal/lightning caused faults as positive classes and non-tree/non-animal/non-lightning caused faults as negative classes; true positive rate (Acc^+) indicates the classification accuracy on tree/animal/lightning caused faults and the true negative rate (Acc) indicates the classification accuracy on non-tree/non-animal/non-lightning caused faults, as shown in Eq. (6.1) and Eq. (6.2) respectively. G-mean shown in Eq. (6.3) examines the classification accuracies on both positive and negative classes, and punishes large disparities between the accuracies on both classes: the g-mean is high when both Acc^+ and Acc are large and the difference between them is

Table 1. Confusion Matrix

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	True Positive (TP)	False Negative (FN)
Actual Negative Class	False Positive (FP)	True Negative (TN)

$$Acc^{+} = \frac{TP}{TP + FN} \tag{6.1}$$

$$Acc^{-} = \frac{TN}{TN + FP} \tag{6.2}$$

$$\text{g-mean} = \sqrt{Acc^{+} \times Acc^{-}} \tag{6.3}$$

small. The classifier in the aforementioned example of imbalanced data set Q only gets a g-mean value of 0, although it has a 95% overall accuracy. Therefore, g-mean is more suitable to measure classification performance when imbalance data are involved.

6.2 Results and Discussions

6.2.1 Accuracy

The outage records from each representative region are divided into training data and test data by year: the data from 1994 to 1999 form the training set and the remaining data (2000-2002) form the test set. For example, the region of HC has 10,030 training data and 5,617 test data.

Due to the randomness involved in AIRS and FAIRS, both algorithms are performed 30 runs for each case in order to generate statistically representative results. The average values of g-mean over the 30 runs for both FAIRS and AIRS as well as the g-mean value from E-algorithm are presented in Fig. 3-5 for lightning caused faults, animal caused faults, and tree caused faults, respectively. For clear presentation, the sequence of three algorithms appeared in the figures is arranged accordingly.

Two-sample tests of hypothesis are performed to compare the performance of different approaches, in which the null hypothesis:

$$H_0\text{: g-mean}_{\text{FAIRS}} = \text{g-mean}_{\text{AIRS/E-algorithm}} \tag{6.4}$$

is tested against the alternative hypothesis:

$$H_1\text{: g-mean}_{\text{FAIRS}} > \text{g-mean}_{\text{AIRS/E-algorithm}} \tag{6.5}$$

The p-value of a test indicates the probability of obtaining the existing sample data given the null hypothesis (Walpole et al. 2002), so a low p-value leads to the rejection of the null hypothesis. The commonly used level of significance 0.05 is applied in this paper, i.e., a p-value under 0.05 will reject the null hypothesis in favor of the

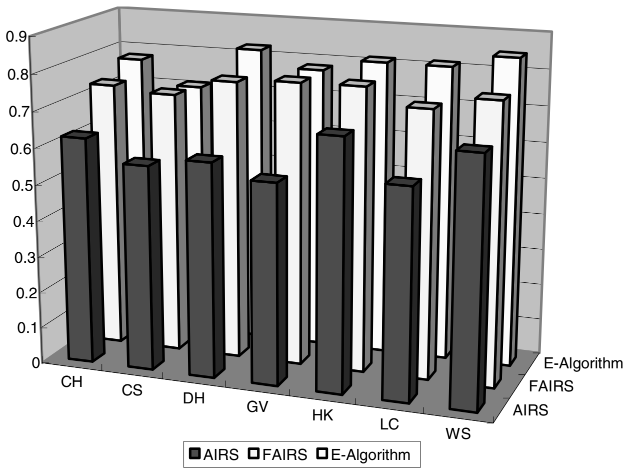


Fig. 3. G-means for lightning caused fault identification

Table 2. P-values of two-sample tests of hypothesis for lightning caused faults

	FAIRS v.s. *E*-algorithm	FAIRS v.s. AIRS
CH	1.0000	<0.0001
CS	<0.0001	<0.0001
DH	1.0000	<0.0001
GV	0.6283	<0.0001
HK	1.0000	<0.0001
LC	1.0000	<0.0001
WS	1.0000	<0.0001

alternative hypothesis. The p-values of the hypothesis tests on g-mean from three algorithms are presented in Table 2-4 (for lightning caused faults, animal caused faults, and tree caused faults, respectively).

Fig.3 shows that *E*-algorithm in general has the highest g-mean values on lightning caused faults and AIRS has the lowest; the sample mean values of g-mean achieved by FAIRS are in between for most of the seven regions. Based on the p-values of hypothesis tests shown in Table 2, it can be concluded with high significance that FAIRS has better performance than AIRS in all the regions since all the p-values are less than 0.0001. On the other hand, *E*-algorithm has larger g-mean than FAIRS in five out of seven regions but smaller in CS; the similar conclusion cannot be drawn for GV at the level of significance 0.05.

From the comparison of animal caused faults identification results shown in Fig.4 and Table 3, FAIRS has a larger g-mean than both AIRS and *E*-algorithm in region LC but a smaller g-mean in region DH, while achieves a g-mean value in between for

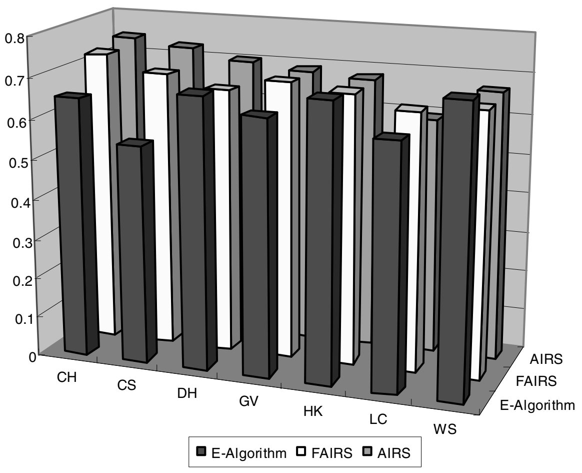


Fig. 4. G-means for animal caused fault identification

Table 3. P-values of two-sample tests of hypothesis for animal caused faults

	FAIRS v.s. *E*-algorithm	FAIRS v.s. AIRS
CH	<0.0001	0.9892
CS	<0.0001	1.0000
DH	0.9989	1.0000
GV	<0.0001	0.4330
HK	0.8917	0.6129
LC	0.0004	<0.0001
WS	1.0000	0.8545

CH and CS. In the remaining three regions, the conclusion about the performance comparison on animal caused faults identification cannot be drawn at the significance level of 0.05, except that FAIRS has a larger g-mean in GV but a smaller g-mean in WS than *E*-algorithm.

Fig.5 and Table 4 show that FAIRS has better performance than both AIRS and *E*-algorithm in three regions (CS, GV, and WS) but worse in two regions (DH and HK).

Based on the comparisons shown in Fig.3-5 and Table 2-4, FAIRS in most of the cases has a comparable performance in terms of g-mean with both AIRS and *E*-algorithm. In only 3 out of all the 21 cases, FAIRS obtains smaller g-mean values than both algorithms (in region DH and HK for tree caused faults and in region DH for animal caused faults). This fact shows that the hybrid algorithm maintain its capability of classifying imbalanced data as its two base algorithms.

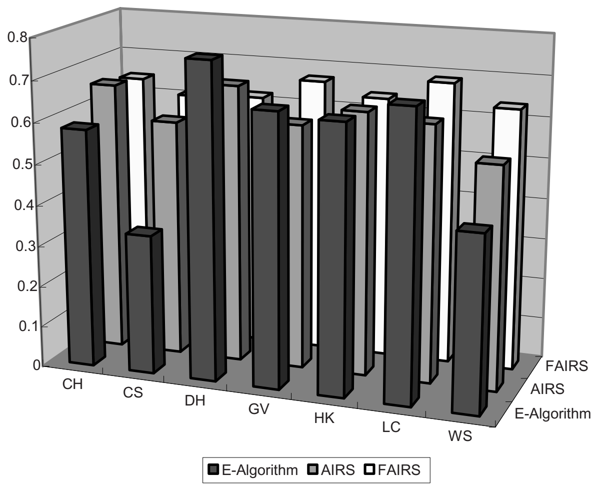


Fig. 5. G-means for tree caused fault identification

Table 4. P-values of two-sample tests of hypothesis for tree caused faults

	FAIRS v.s. *E*-algorithm	FAIRS v.s. AIRS
CH	<0.0001	0.9491
CS	<0.0001	<0.0001
DH	1.0000	1.0000
GV	0.0001	<0.0001
HK	1.0000	0.9987
LC	0.9525	<0.0001
WS	<0.0001	<0.0001

6.2.2 Computational Requirement

While the focus has not been on algorithmic complexity analysis for this current paper, it is worthwhile to compare the running time of three algorithms so as to have a taste of their computational demand. The test results using tree caused faults identification in region CH as an example are shown in Table 5. The computing time test is based on a PC with 1.8GHz Intel(R) Pentium(R) M processor and 512MHz RAM.

It is quite clear that *E*-algorithm takes much longer running time than the other two algorithms. As discussed in Section 2, *E*-algorithm needs to generate a large initial set of fuzzy rules (which contains 57,904 rules in this case) and further calculate the compatibility grade of each fuzzy rule with all the training data. In this example, 3,120 training data from the region CH leads to 180,660,480 compatibility grade calculations. Most of the calculations are wasted since only a few rules (30 rules as in (Xu et al. 2006)) are finally used for classification tasks.

As discussed in Section 2, the total number of possible fuzzy set combinations increases exponentially with the number of attributes included (or rule length). The

Table 5. Comparisons of running times of three algorithms

	FAIRS	AIRS	*E*-algorithm
Average running time for each run	64 seconds	56 seconds	47 minutes 47 seconds

Table 6. The number of rules generated by FAIRS

	CH	CS	DH	GV	HK	LC	WS
Lightning	27	27	25	27	31	30	30
Animal	35	32	32	28	33	24	32
Tree	31	35	37	39	35	36	39

current heavy computation load from *E*-algorithm is based on already restricted rule length. On the other hand, FAIRS uses the quick search mechanism from immune system to generate a small set of fuzzy rules for classification with significantly reduced computing time (the numbers of rules generated by FAIRS algorithm are shown in Table 6). Furthermore, the rule length is not limited by using the antecedent attribute inclusion indicator.

6.2.3 Fuzzy Rules

As one of the motivations for developing this algorithm, FAIRS integrates rule extraction capability into the immune system algorithms so that the fault diagnosis inference mechanism can be presented linguistically. Two rules for identifying lightning caused faults are listed as examples.

1. IF the likelihood measure of Time of Day is low (according to membership partition No.1 in Fig.1), THEN the outage is NOT likely to be caused by lightning.
2. IF the likelihood measure of Weather is medium low (according to membership partition No.3 in Fig.1) AND the likelihood measure of Protective Devices Activated is medium low (according to membership partition No.3 in Fig.1) AND the likelihood measure of Season is high (according to membership partition No.1 in Fig.1) AND the likelihood measure of Time of Day is high (according to membership partition No.4 in Fig.1), THEN the outage is likely to be caused by lightning.

Rule 2 contains four antecedent attributes (weather condition, protective devices activated by the outage, season, and time of day); in other words, its rule length is 4. The presence of this rule demonstrates the rule length flexibility of FAIRS: the algorithm can generate the rules with arbitrarily length as necessary rather than limit the rule length in order to reduce the computational demand as in *E*-algorithm.

7 Conclusions

Power distribution system fault diagnosis is important for maintaining the system availability and reliability. However, real-world data imperfections such as data imbalance have been affecting the performance of many fault diagnosis methods. *E*-algorithm and AIRS have demonstrated good capabilities in imbalanced data classifications as investigated in our previous works.

E-algorithm can produce inference rules but is computational demanding; AIRS is able to quickly search class patterns but is lack of rule extraction capability. In order to utilize both advantages, a hybrid algorithm of *E*-algorithm and AIRS, Fuzzy Artificial Immune Recognition System (FAIRS), has been proposed in this paper. FAIRS has been applied to Duke Energy historical outage data which involve a large degree of imbalance to demonstrate its performance on real world imperfect data.

The proposed FAIRS algorithm is compared with both its base algorithms using g-mean as the performance measure. The results show that FAIRS achieves comparable performance with AIRS and *E*-algorithm, but it significantly reduces the computation time (compared with *E*-algorithm) while keeps the capability of extracting linguistic rules to explain the inference. At the same time, the rule length flexibility is improved.

References

Brown, R.E.: Electric Power Distribution Reliability. Marcel Dekker, New York (2002)

Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Approach. Springer, London (2002)

Chow, M.Y., Taylor, L.S.: Analysis and prevention of animal-caused faults in power distribution systems. IEEE Transaction Power Delivery 10(2), 95–1001 (1995)

Han, J., Kamber, M.: Data Mining: Concepts and Techniques. San Francisco. Academic Press, Morgan Kaufmann Publishers, London, San Francisco (2001)

Ishibuchi, H., Nakashima, T., Murata, T.: Voting in fuzzy rule-based systems for pattern classification problems. Fuzzy Sets and Systems 103(2), 223–238 (1999)

Ishibuchi, H., Yamamoto, T.: Comparison of heuristic rule weight specification methods. In: Proc. IEEE International Conference on Fuzzy Systems, pp. 908–913 (2002)

Kubat, M., Holte, R., Matwin, S.: Machine Learning for the Detection of Oil Spills in Radar Images. Machine Learning 30, 195–215 (1998)

Walpole, R.E., et al.: Probability and Statistics for Engineers and Scientists, 7th edn. Prentice Hall, Englewood Cliffs (2002)

Watkins, A.: AIRS: A Resource Limited Artificial Immune Classifier. M.S. Thesis, Mississippi State University (2001)

Watkins, A.: Artificial Immune Recognition System (AIRS) C++ source code (2002)

Watkins, A., Timmis, J.: Artificial Immune Recognition System (AIRS): Revisions and Refinements. In: Proc. 1st International Conference on Artificial Immune Systems (ICARIS), pp. 173–181 (2002)

Watkins, A., Timmis, J., Boggess, L.: Artificial Immune Recognition System (AIRS): An Immune Inspired Supervised Machine Learning Algorithm. Genetic Programming and Evolvable Machines 5(1) (2004)

Xu, L., Chow, M.Y., Taylor, L.S.: Analysis of Tree-Caused Faults in Power Distribution Systems. In: Proc North American Power Symposium, pp. 122–127 (2003)

Xu, L., Chow, M.Y., Taylor, L.S.: Data Mining Based Fuzzy Classification Algorithm for Imbalanced Data. In: Proc. IEEE World Congress on Computational Intelligence, pp. 4216–4221 (2006)

Xu, L., et al.: On the Investigation of Artificial Immune Systems on Imbalanced Data Classification. In: Proc. IEEE World Congress on Computational Intelligence, pp. 1629–1634 (2006)

Xu, L., Chow, M.Y.: A Classification Approach for Power Distribution Systems Fault Cause Identification. IEEE Transaction on Power Systems 21(1), 53–60 (2006)

Detection of Phishing Attacks: A Machine Learning Approach

Ram Basnet, Srinivas Mukkamala, and Andrew H. Sung

New Mexico Tech, New Mexico 87801, USA
{ram,srinivas,sung}@cs.nmt.edu

1 Introduction

Phishing is a form of identity theft that occurs when a malicious Web site impersonates a legitimate one in order to acquire sensitive information such as passwords, account details, or credit card numbers. Though there are several anti-phishing software and techniques for detecting potential phishing attempts in emails and detecting phishing contents on websites, phishers come up with new and hybrid techniques to circumvent the available software and techniques.

Phishing is a deception technique that utilizes a combination of social engineering and technology to gather sensitive and personal information, such as passwords and credit card details by masquerading as a trustworthy person or business in an electronic communication. Phishing makes use of spoofed emails that are made to look authentic and purported to be coming from legitimate sources like financial institutions, ecommerce sites etc., to lure users to visit fraudulent websites through links provided in the phishing email. The fraudulent websites are designed to mimic the look of a real company webpage.

The phishing attacker's trick users by employing different social engineering tactics such as threatening to suspend user accounts if they do not complete the account update process, provide other information to validate their accounts or some other reasons to get the users to visit their spoofed web pages.

Why is it important to tackle the problem of phishing? According to the Anti-Phishing Working Group, there were 18,480 unique phishing attacks and 9666 unique phishing sites reported in March 2006. Phishing attacks affect millions of internet users and are a huge cost burden for businesses and victims of phishing (Phishing 2006). Gartner research conducted in April 2004 found that information given to spoofed websites resulted in direct losses for U.S. banks and credit card issuers to the amount of $1.2 billion (Litan 2004). Phishing has become a significant threat to users and businesses alike.

Over the past few years, much attention has been paid to the issue of security and privacy. Existing literature dealing with the problem of phishing is scarce. Fette et al proposed a new method for detecting phishing emails by incorporating features specific to phishing (Fette et al. 2006).

We applied different methods for detecting phishing emails using known as well as new features. We employ a few novel input features that can assist in discovering phishing attacks with very limited a-prior knowledge about the adversary or the method used to launch a phishing attack. Our approach is to classify phishing emails by incorporating key structural features in phishing emails and employing different

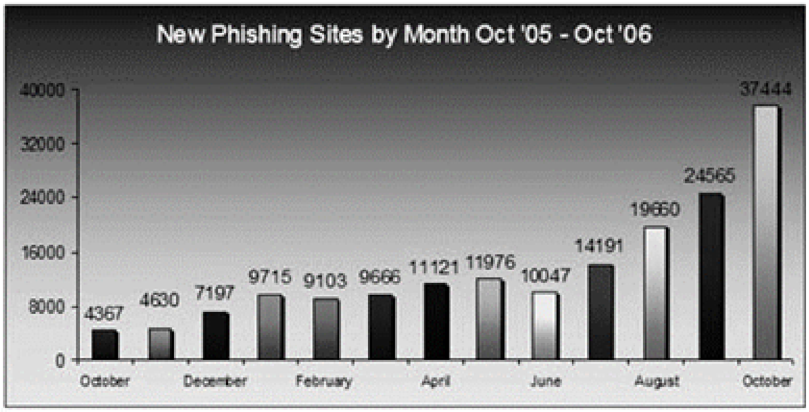


Fig. 1. Unique phishing site URLs rose 757 percent in one year

machine learning algorithms to our dataset for the classification process. The use of machine learning from a given training set is to learn labels of instances (phishing or legitimate emails). Our paper provides insights into the effectiveness of using different machine learning algorithms for the purpose of classification of phishing emails.

Soft Computing techniques are increasingly being used to address a gamut of computational problems. Clustering is a type of unsupervised learning; unsupervised learning assumes that there is no previous knowledge about the class membership of the observations, i.e., class labels of data is unknown. The purpose of using unsupervised learning is to directly extract structure from a dataset without prior training. Although, supervised learning provides for a much better accuracy, unsupervised learning provides for a fast and reliable approach to derive knowledge from a dataset.

This paper is organized as follows: In section 2 we provide an overview of features used in our experiments. In section 3 we present the data used. In section 4 we describe and present the different experiments conducted and also present the performance results of various machine learning algorithms. Finally, in section 5 we provide concluding remarks.

2 Features Used

There exist a number of different structural features that allow for the detection of phishing emails. In our approach, we make use of sixteen relevant features. The features used in our approach are described below.

- HTML Email: HTML-formatted emails are mainly used for phishing attacks, because plaintext emails do not provide for the scale of tricks afforded with HTML-formatted emails. Hyperlinks are active and clickable only in html-formatted emails. Thus, a HTML-formatted email is flagged and is used as a binary feature.

- IP-based URL: One way to obscure a server's identity is achieved through the use of an IP address. Use of an IP address makes it difficult for users to know exactly where they are being directed to when they click the link. A legitimate website usually has a domain name for its identification. Phishers usually use some zombie systems to host phishing sites. When a link in an email contains a link whose host is an IP address (for example, *http://81.215.214.238/pp/*) we flag the email and is used as a binary feature.
- Age of Domain Name: The domain names (if any) used by fraudsters are usually used for a limited time frame to avoid being caught. We can thus use this feature to flag emails as phishing based on the fact that the domain is newly registered and set a criteria of being new if it is less than 30 days old. This can be achieved by performing a WHOIS query on the domain name in the link. A WHOIS query provides other information such as the name or person to which the domain is registered to, address, domain's creation and expiration dates etc. This feature is a binary.
- Number of Domains: We make use of the domain names in the links that we extract and do a count of the number of domains. Two or more domain names are used in an URL address to forward address from one domain to the other. *http://www.google.com/url?sa=t&ct=res&cd=3&url=http%3A%2F%2Fwww.antiphishing.org%2F&ei=-0qHRbWHK4z6oQLTm-BM&usg=uIZX_3aJvESkMveh4uItI5DDUzM=&sig2=AVrQFpFvihFnLjpnGHVsxQ* for instance has two domain names where google.com forwards the click to URL antihphishing.org domain name. The number of domains we count is considered a continuous feature.
- Number of Sub-domains: Fraudsters make use of sub domains to make the links look legitimate. Having sub domains means having an inordinately large number of dots in the URL. We can make use of this feature to flag emails as phishing emails. For instance, *https://login.personal.wamu.com/verification.asp?d=1* has 2 sub domains. This is a continuous feature.
- Presence of JavaScript: JavaScript is usually employed in phishing emails, because it allows for deception on the client side using scripts to hide information or activate changes in the browser. Whenever an email contains the string "JavaScript", we flag it as a phishing email and use it as a binary feature.
- Presence of Form Tag: HTML forms are one of the techniques used to gather information from users. An example below shows the use of form tag in an email. An email supposedly from Paypal may contain a form tag which has the action attribute actually sending the information to *http://www.paypal-site.com/profile.php* and not to *http://www.paypal.com*. The email used for collecting user's info has form tag *<FORM action=http://www.paypal-site.com/profile.php method=post>* for example.
- Number of Links: Most often phishing emails will exploit the use of links for redirection. The number of links in email is used as a feature. A link in an email is one that makes use of the "href" attribute of the anchor tag. This feature will be continuous.
- URL Based Image Source: To make the phishing emails look authentic, images and banner of real companies are used in the emails. Such images are usually linked from the real companies' web pages. Thus, if any of the emails make use of such URL based images we flag it as a phishing email. This feature is binary.

- Matching Domains (From & Body): We make use of the information from the header of the email and match it with the domains in the body of the email. Most phishing emails will have different domains in the header and the body part. We will thus flag emails that have mismatching domain information. For example: The 'From' information in the header part of the email will show the email originating from "someone@paypal-site.com", while the body will have actual (*"http://www.paypal.com"*) company's domain for an authentic look. This feature is binary.
- Keywords: Phishing emails contain number of frequently repeated keywords such as suspend, verify, username, etc. We use word frequency (Count of keyword divided by total number of words in an email) of a handful of most commonly used keywords by phishers. This feature is continuous.
- Some handful of keywords if present in emails are counted and normalized. Group of words with similar meaning or synonyms are used as a single feature. We use six groups of keywords as six separate features. Six groups of keywords that we have used as features are listed below:
  - Update, Confirm
  - User, Customer, Client
  - Suspend, Restrict, Hold
  - Verify, Account
  - Login, Username, Password
  - SSN, Social Security

3 Data Used

To implement and test our approach, we have used two publicly available datasets i.e., the ham corpora from the SpamAssassin project as legitimate emails and the emails from PhishingCorpus as phishing emails (Phishing 2006, Spam 2006). The total number of emails used in our approach is 4000. Out of which 973 are used as phishing emails and 3027 as legitimate (ham) emails. The entire dataset is divided into two parts for testing and training purpose. A total of 2000 emails are considered as training samples and the remaining are considered for testing purpose. The tabular form of different samples used:

Table 1. Data used for experiments

Total samples	4000
Total phishing emails	973
Total legitimate emails	3027
Total training samples	2000
Total testing samples	2000

We used Python and Java scripts to parse the phishing and legitimate (ham) emails and extract the features mentioned above in section 3. We have a total of sixteen attributes for each email relation.

4 Experiments

To evaluate our implementation, we used different machine learning methods and a clustering technique on our phishing dataset. We used Support Vector Machines (SVM, Biased SVM & Leave One Model Out), Neural Networks, Self Organizing Maps (SOMs) and K-Means on the dataset described in section 3.

4.1 Model Selection of Support Vector Machines (SVMs)

In any predictive learning task, such as classification, both a model and a parameter estimation method should be selected in order to achieve a high level of performance of the learning machine. Recent approaches allow a wide class of models of varying complexity to be chosen. Then the task of learning amounts to selecting the sought-after model of optimal complexity and estimating parameters from training data (Chapelle 1999, Cherkassy 2002, Lee 2000).

Within the SVMs approach, usually parameters to be chosen are (i) the penalty term C which determines the trade-off between the complexity of the decision function and the number of training examples misclassified; (ii) the mapping function Φ; and (iii) the kernel function such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{1}$$

In the case of RBF kernel, the width, which implicitly defines the high dimensional feature space, is the other parameter to be selected (Chapelle 1999).

We performed a grid search using 5-fold cross validation for each of the faults in our data set. We achieved the search of parameters C and γ in a coarse scale.

4.2 Biased Support Vector Machines (BSVMs)

Biased support vector machine (BSVM), a decomposition method for support vector machines (SVM) for large classification problems (Chan 2004). BSVM uses a

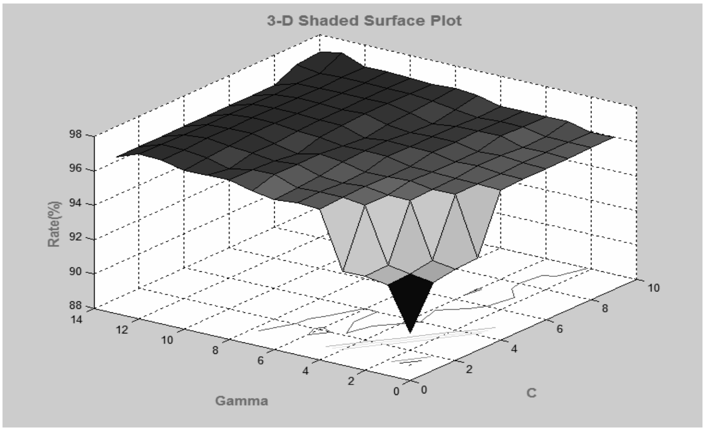


Fig. 2. 3-D view of accuracy for different Gamma and C pairs

decomposition method to solve a bound-constrained SVM formulation. BSVM Uses a simple working set selection which leads to faster convergences for difficult cases and a bounded SVM formulation and a projected gradient optimization solver which allow BSVM to quickly and stably identify support vectors.

Leave-one-out model selection for biased support vector machines (BSVM) is used for automatic model selection (Chan 2004). Model selection results BSVM using LOOMS are given in figure 2.

4.3 Neural Networks

Artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. A neural network conducts an analysis of the information and provides a probability estimate that it matches with the data it has been trained to recognize. The neural network gains the experience initially by training the system with both the input and output of the desired problem. The network configuration is refined until satisfactory results are obtained. The neural network gains experience over a period as it is being trained on the data related to the problem. Since a (multi-layer feedforward) ANN is capable of making multi-class classifications, a single ANN (Scaled Conjugate Gradient), is employed for classification, using the same training and testing sets.

4.3.1 Scaled Conjugate Gradient Algorithm

The scaled conjugate gradient algorithm is an implementation of avoiding the complicated line search procedure of conventional conjugate gradient algorithm (CGA). According to the SCGA, the Hessian matrix is approximated by

$$E^{''}(w_k)p_k = \frac{E^{'}(w_k + \sigma_k p_k) - E^{'}(w_k)}{\sigma_k} + \lambda_k p_k \tag{2}$$

where E' and E'' are the first and second derivative information of global error function $E\ (w_k)$. The other terms p_k, σ_k and λ_k represent the weights, search direction, parameter controlling the change in weight for second derivative approximation and parameter for regulating the indefiniteness of the Hessian. In order to get a good quadratic approximation of E, a mechanism to raise and lower λ_k is needed when the Hessian is positive definite (Moller 1993).

We ran experiments for 17 times with the same neural network settings. The first experiment was run using all the 16 features and we got 97.8% accuracy. Then for each experiment we removed one feature. So, each 16 experiment had one less feature (total 15) starting from feature no. 1, no. 2 so on and so forth. The following graph shows the result of accuracy for each experiment with one less feature. The motivation for doing so was to see which set of 15 features produces the highest accuracy which in turn might help us do some sort of feature selection. Results using different features are summarized in figure 3.

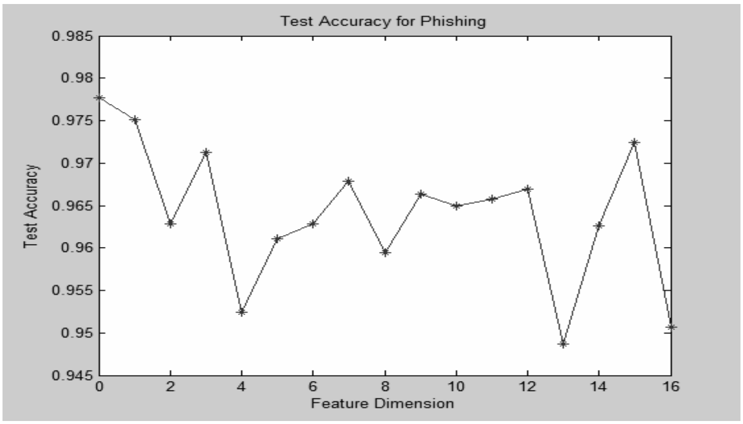


Fig. 3. Neural network results using different features

4.3.2 Self Organizing Maps (SOMs)

Self-Organizing Map (SOM) is an unsupervised algorithm that performs clustering of input data and maps it to a two-dimensional map. In the map, the similar data items will be mapped to nearby locations on the map (Vesanto 1999).

SOM-based analysis can be done using a visualization technique called the U-matrix, which shows the cluster structure of the map. High values of the U-matrix indicate a cluster border. While, uniform areas of low values indicate the clusters themselves. The component planes are also useful for visualizing the different components in the reference vectors. For visualizing the SOM of the phishing data, we took a sample of 200 emails (consisting of 50% legitimate and 50% phishing emails).

From the U-matrix we can see that the top rows of the SOM form a clear cluster. From the labels we can see that this corresponds to the legitimate emails (as represented by 'h'). The phishing emails form the other visible cluster on the lower part of the SOM. This is also indicated by the labels as phishing emails (as represented by 'p').

From the 16 component plane figures of Figure 4, we can visualize the clustering patterns for the different features. The highlights can be summarized as following:

1. Html Email: Almost all phishing emails are HTML-based
2. IP based URL: High values for phishing emails
3. Age of domain name: A small cluster concentrated on phishing emails
4. No of Domains: Very few of the emails (both phishing and legitimate had URL directed)
5. Max Sub Domains: This feature had prominence in phishing emails
6. Presence of JavaScript: Small cluster concentrated on phishing emails
7. Presence of Form Tag: Small cluster distributed evenly between phishing and legitimate emails
8. Number of Links: This feature had prominence in phishing emails
9. Image Source URL: This feature had prominence in phishing emails
10. Domain matching (From and Body): This feature had prominence in phishing emails.

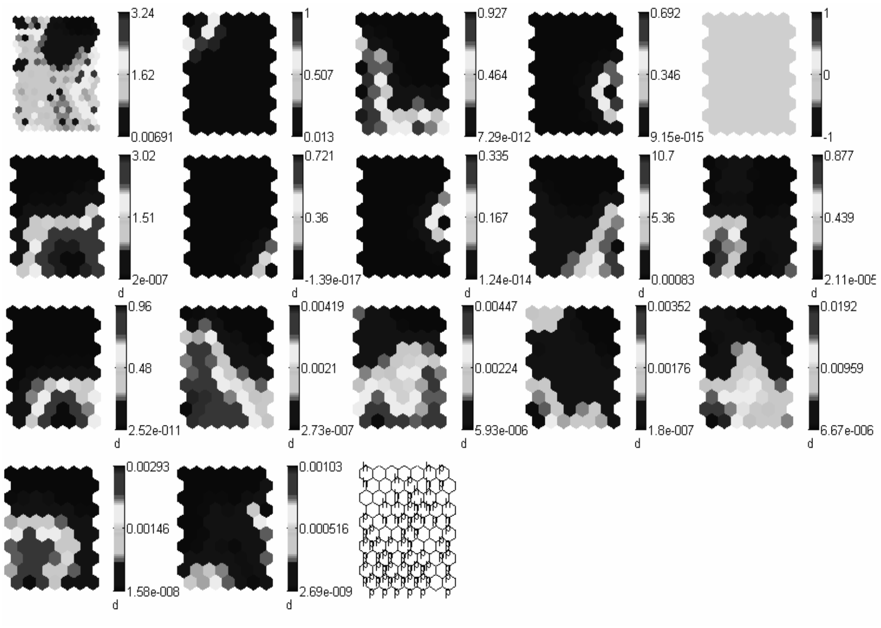


Fig. 4. Visualization of the SOMs for phishing data

11. Features 11 to 16 Keywords: The keywords features had very high prominence in phishing emails

The map unit labels also clearly show the clustering patterns. This complements the cluster pattern seen in the U-matrix. The U-matrix is shown on the top left. The other 16 figures following the U-matrix are the component planes. The bottom right figure is the map unit labels.

4.4 K-Means

K-means clustering is an unsupervised non-hierarchal clustering. This attempts to improve the estimate of the mean of each cluster and re-classifies each sample to the cluster with nearest mean. Practical approaches to clustering use an iterative procedure, which converges to one of numerous local points. These iterative techniques are sensitive to initial starting conditions. The refined initial starting condition allows the iterative algorithm to converge to a "better" local point. The procedure is being used in k-means clustering algorithm which being used for both discrete and continuous data points. Let us consider a n example feature vectors $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ all from the same class, and we know that they fall into k compact clusters, $k < n$. Let $\mathbf{m}_i$ be the mean of the vectors in Cluster I. If the clusters are well separated, we can use a minimum-distance classifier to separate them. That is, we can say that $\mathbf{x}$ is in Cluster i if $\| \mathbf{x} - \mathbf{m}_i \|$ is the minimum of all the k distances (Witten 2005).

Each cluster then creates a centroid frequency distribution. Each instance is then iteratively reassigned to the cluster with the closest centroid. When instances stop moving between clusters, the iteration process also stops.

K-Means aims at minimizing the objective function below (Anderberg 1973):

$$J = \sum_{j=1}^{k} \sum_{i=1}^{x} \| x_i^{(j)} - C_j \|^2 \tag{3}$$

where $\| x_i^{(j)} - C_j \|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ Cluster center C_j, is an indicator of the distance of the *n* data points from Ftheir respective cluster centers.

Table 2. Accuracies of K-Means on phishing dataset

Clustering Technique	Incorrectly Clustered Instances	Incorrectly Clustered Instances %	Accuracy %
K-means	635.0	9.2109 %	90.7891%

5 ROC Curves (SVMs)

ROC is a graphical plot between the sensitivity and specificity. The ROC is used to represent the plotting of the fraction of true positives (TP) versus the fraction of false positives (FP).

The point (0,1) is the perfect classifier, since it classifies all positive cases and negative cases correctly. Thus an ideal system will initiate by identifying all the

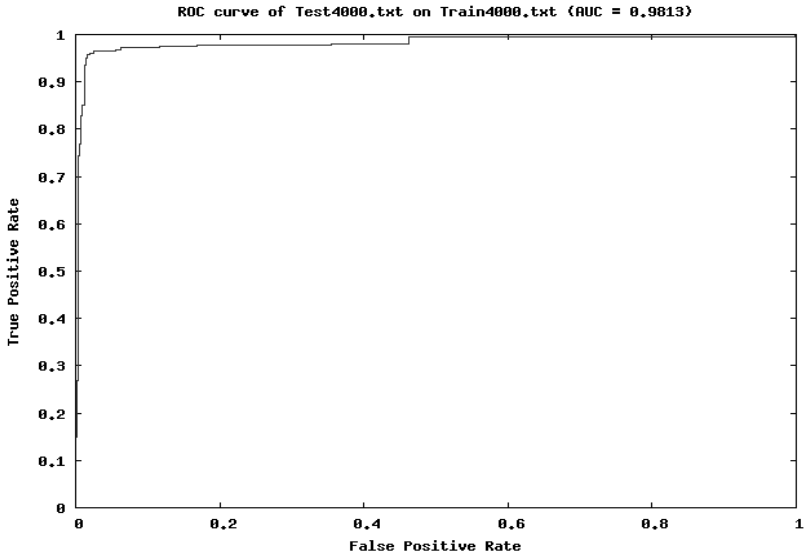


Fig. 5. Phishing attack detection accuracy using SVMs

positive examples and so the curve will rise to (0,1) immediately, having a zero rate of false positives, and then continue along to (1,1). Detection rates and false alarms are evaluated for the phishing data set and the obtained results are used to form the ROC curves. In each of these ROC plots, the x-axis is the false alarm rate, calculated as the percentage of normal emails considered as phishing attacks; the y-axis is the detection rate, calculated as the percentage of phishing attacks detected. A data point in the upper left corner corresponds to optimal high performance, i.e, high detection rate with low false alarm rate (Egan 1975).

The accuracy of the test depends on how well the test classifies the group being tested into 0 or 1. Accuracy is measured by the area under the ROC curve (AUC). An Area of 1 represents a perfect test and an area of .5 represents a worthless test. In our experiment, we got an **AUC of 0.9813** as shown in Figure 5.

6 Summary and Future Work

Although the performance of six different machine learning methods used is comparable, we found that Support Vector Machine (LIBSVM) achieved consistently the best results. Biased Support Vector Machine (BSVM) and Artificial Neural Networks gave the same accuracy of 97.99%.

We have added new features to what researchers have published in literature. The classifiers used in this paper showed comparable or better performance in some cases when compared to the ones reported in the literature using the same datasets. Our results demonstrate the potential of using learning machines in detecting and classifying phishing emails. As a future work we plan to use more machine learning algorithms to compare accuracy rates. We also plan to do a thorough feature ranking and selection on the same data set to come up with the set of features that produces the best accuracy consistently by all the classifiers.

Acknowledgements

Support for this research received from ICASA (Institute for Complex Additive Systems Analysis, a division of New Mexico Tech), and DOD IASP Capacity Building grant is gratefully acknowledged.

References

Anti-Phishing Working Group (2006) Phishing Activity Trends Report. http://www.antiphishing.org/reports/apwg_report_mar_06.pdf

Litan A (2004) Phishing Attack Victims Likely Targets for Identity Theft. Gartner Research

Fette I, Sadeh N, Tomasic A (2006) Learning to Detect Phishing Emails. Technical Report CMU-ISRI-06-112. Institute for Software Research International, Carnegie Mellon University

Phishing Corpus (2006) http://monkey.org/~jose/wiki/doku.php?id=PhishingCorpus

Spam Assassin (2006) http://spamassassin.apache.org/

Anti-Phishing Working Group, Phishing Activity Trends Report (2006), http://www.antiphishing.org/reports/apwg_report_mar_06.pdf

Litan, A.: Phishing Attack Victims Likely Targets for Identity Theft. Gartner Research (2004)

Fette, I., Sadeh, N., Tomasic, A.: Learning to Detect Phishing Emails. Technical Report CMU-ISRI-06-112. Institute for Software Research International, Carnegie Mellon University (2006)

Phishing Corpus (2006), http://monkey.org/~jose/wiki/doku.php?id=PhishingCorpus

Spam Assassin (2006) http://spamassassin.apache.org/

Chapelle, O., Vapnik, V.: Model Selection for Support Vector Machines. Advances in Neural Information Processing Systems 12

Cherkassy, V.: Model Complexity Control and Statistical Learning Theory. Journal of Natural Computing 1, 109–133 (2002)

Lee, J.H., Lin, C.J.: Automatic Model Selection for Support Vector Machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University (2000)

Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. Department of Computer Science and Information Engineering, National Taiwan University (2001)

Chan, C.H., King, I.: Using Biased Support Vector Machine to Improve Retrieval Result in Image Retrieval with Self-organizing Map. In: Proceedings of International Conference on Neural Information Processing, pp. 714–719. Springer, Heidelberg (2004)

Moller, A.F.: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Neural Networks 6, 525–533 (1993)

Vesanto, J., et al.: Self Organizing Map (SOM) Toolbox. In: Proceedings of Mat lab DSP Conference, Finland, pp. 35–40 (1999)

Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

Anderberg, M.: Cluster Analysis for Applications. Academic Press, London (1973)

Egan, J.P.: Signal Detection Theory and ROC Analysis. Academic Press, New York (1975)

Author Index