IJCI Menofyia University
Faculty of Computers & Information

International Journal of Computers and Information

Implementation of a Real-Time Simulator for Dynamic Systems

H. M. Abdull-Kader,
Menoufia University, Faculty of Computers and Information, Information Systems Dept., Egypt,
hatem6803@yahoo.com

A. B. El-Sisi,
Menoufia University, Faculty of Computers and Information, Computer Science Dept., Egypt

K. A. Mostafa,
Menoufia University, Faculty of Computers and Information, Information Technology Dept., Egypt

Imbaby I. Mahmoud
Engineering Dept., Atomic Energy Authority, Cairo, Egypt,
imbabyisma@frcu.eun.eg

Abstract

Real-time systems refer to systems that have real-time requirements for interacting with a human operator or other agents with similar timescales. An efficient simulation of real-time systems requires a model that is accurate enough to accomplish the simulation objective and is computationally efficient. In this paper a real time modeling system for dynamic systems will be studied. Normally, Real time modeling can be classified into hardware and software systems, but this work focuses on the software techniques and systems. Finally a demonstration example for real time simulator has been simulated for complex dynamic system, namely a small nuclear fusion device (Egyptor Tokamak). The obtained results agree well with published work. Such simulator can be considered an imperative requirement for predicative control tasks.

1. Introduction

Real-time simulation has been used for many years for operators training and design or testing of hardware and software in situations in which it is inconvenient to use the real system [1-4]. Examples include flight simulators for pilot training, plant simulators for operator training, and a wide range of applications in which the simulator is used to test hardware and embedded software in the loop. One of the key parameters of a real-time simulator is the frame rate. Many real time simulators, are used for operator training, perform satisfactorily with frame times in the range 10 to 100ms [5]. All the calculations needed to advance the simulation by one frame must be completed, along with all necessary data transfers within one frame. In some applications real-time simulation is used as a test environment for real hardware or embedded software referred to as the system under test (SUT) [6]. In some cases much shorter frame times (<10µs) are required because of the high-frequency dynamics of both the simulated system and (SUT). Such applications are found, for example, in aerospace, automotive and power electronic systems [7], [8]. The techniques described here focus on power electronic systems, but they are equally relevant to other applications requiring similar frame times. In the literatures, many dynamic systems applications can be found such as robotics, industrial production systems, and nuclear power plants [9-12].

In this paper we will focus on the nuclear power plant application. This type of application can be classified in two main categories namely nuclear fusion experiments and fission reactors. The structure of this paper is as follows. Section 2 gives an overview about real-time modeling. In section 3 the details of case study is introduced. Section 4 describes the proposed simulator for our case study. The result of the proposed simulator is shown in section 5. Finally, some conclusion are put forward in Section 6.

2. Modeling Techniques of Real-Time Systems

Real-time systems differ from traditional data processing systems in that they are constrained by certain nonfunctional requirements (e.g., dependability and timing). Although real-time systems can be modeled using the standard structured design methods, these methods lack explicit support for expressing the real-time constraints. Standard structured design methods incorporate a life cycle model in which the following activities are recognized:

(1) Requirements definition. An authoritative specification of the system's required functional and nonfunctional behavior is produced.
(2) Architectural design. A top-level description of the proposed system is developed.
(3) Detailed design. The complete system design is specified.
(4) Coding. The system is implemented.
(5) Testing. The efficacy of the system is tested.

Real time simulation systems can be implemented via software or hardware. For hard real-time systems, this has the significant disadvantage that timing problems will be recognized only during testing, or worse, after deployment. Researchers have pointed out that the time requirements should be addressed in the architectural design phase [13]. The architectural design activities are defined as the logical architecture design activity, and the physical architecture design activity. The logical architecture embodies commitments that can be made independently of the constraints imposed by the execution environment, and is primarily aimed at satisfying the functional requirements. The physical architecture takes these functional requirements and other constraints into account, and embraces the non-functional requirements. The physical architecture forms the basis for asserting that the application's nonfunctional requirements will be met once the detailed design and implementation have taken place [14]. The physical architecture design activity addresses timing (e.g., responsiveness, orderliness, temporal predictability and temporal controllability) and dependability requirements (e.g., reliability, safety and security), and the necessary schedulability analysis that will ensure that the system once built will function correctly in both the value and time domains. Appropriate scheduling paradigms are often integrated to handle nonfunctional requirements.

3. Methods for Modeling dynamic systems

Methods for modeling dynamic systems can be classified into different categories. The first one is mathematical models and the second is heuristics models. Also there are some methods which use a combination between the previous two methods called hyperid methods.

3.1.1. Mathematical Methods. In this method the system can be governed by a set of equations called system equations. Before discussing dynamic models, let us recall that time-invariant dynamic systems are in general modeled by static functions, by using the concept of the system's state. Given the state of a system and given its input, we can determine what the next state will be. In the discrete-time setting we can write

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

where x(k) and u(k) are the state and the input at time k, respectively, and f is a static function, called the *state-transition function*.

Dynamic models of different types can be used to approximate the state-transition function. As the state of a process is often not measured, *input-output* modeling is usually applied. The most common is the NARX (Nonlinear Auto Regessive with exogenous input) model:

$$y(k+1) = f(y(k), y(k-1)..., y(k-ny+1), u(k), u(k-1),...,u(k-nu+1)) \tag{2}$$

Here $y(k), \&, y(k-ny+1) and\ u(k), \&, u(k-nu+1)$ denote the past model outputs and inputs respectively and n_y, n_u are integers related to the model order (usually selected by the user).

In this sense, we can say that the dynamic behavior is taken care of by external dynamic filters added to the dynamic system. In equation (2), the input dynamic filter is a simple generator of the lagged inputs and outputs, and no output filter is used. Since the dynamic models can approximate any smooth function to any degree of accuracy, models of type, given by equation (2), can approximate any observable and controllable modes of a large class of linear or discrete-time nonlinear systems.

3.1.2. Heuristics Methods. In this method we can not describe the behavior of the system by a known set of equations. But we need fitting methods to get approximation. So we need information about the system from the field of this system. Two common sources of information for building dynamic models are the prior knowledge and data process measurements. The prior knowledge can be of a rather approximate nature (qualitative knowledge, heuristics), which usually originates from "experts", i.e., process designers, operators, etc. In this sense, dynamic models can be regarded as simple dynamic expert systems. For many processes, the data that are available as records of the process operation or special identification experiments can be designed to obtain the relevant data. Building dynamic models from data involves methods based on dynamic logic and approximate reasoning, but also ideas originating from the field of neural networks, data analysis and conventional systems identification. The acquisition or tuning of dynamic models by means of data is usually termed dynamic identification.

In the literature there are two main approaches to the integration of knowledge and data in a dynamic model. These approaches can be distinguished as:

(1) The expert knowledge expressed in a verbal form is translated into a collection of If–Then rules. In this way, a certain model structure is created. Parameters in this structure (membership functions, consequent singletons or parameters) can be fine-tuned using input output data [15].

(2) No prior knowledge about the system under study is initially used to formulate the rules, and a dynamic model is constructed from data. It is expected that the extracted rules and membership functions can provide a posteriori interpretation of the system's behavior. Related to which one of the previous an expert can confront this information with his own knowledge, can modify the rules, or supply new ones, and can design additional experiments in order to obtain more informative data. These techniques, of course, can be combined, depending on the particular application

4. Selection of model Structure and parameters

With regard to the design of dynamic models, two basic items are distinguished: the structure and the parameters of the model. The structure determines the flexibility of the model in approximation (unknown) mappings. The parameters are then tuned (estimated) to fit the data at hand. A model with a rich structure is able to approximate more complicated functions, but, at the same time, has worse *generalization* properties. Good generalization means that a model fitted to one data set will also perform well on another data set from the same process. In dynamic models, structure selection involves the following choices:

1. *Input and output variables.* With complex systems, it is not always clear which variables should be used as inputs to the model. In the case of dynamic systems, one also must estimate the order of the system. For example the input-output NARX model, this means to define the number of input and output lags n_y and n_u, respectively. Prior knowledge, insight in the process behavior and the purpose of modeling are the typical sources of information for this choice. Sometimes, automatic data-driven selection can be used to compare different choices in terms of some performance criteria.

2. *Structure of the rules*. This choice involves the model type and the antecedent form . Important aspects are the purpose of modeling and the type of available knowledge.

3. *Number and type of membership functions for each variable.* This choice determines the level of detail (granularity) of the model. Again, the purpose of modeling and the detail of available knowledge, will influence this choice [16].

4. *Type of the inference mechanism, connective operators.* These choices are restricted by the type of dynamic model [17].

To facilitate data-driven optimization of dynamic models (learning), differentiable operators (product, sum) are often preferred to the standard min and max operators. After the structure is fixed, the performance of a dynamic model can be fine-tuned by adjusting its parameters. Tunable parameters models are the parameters of antecedent and consequent membership functions (determine their shape and position) and the rules (determine the mapping between the antecedent and consequent dynamic regions).

One of the most parameter to select the model is *model accuracy* which define the ability of a model to capture the system at the right level of detail and to achieve the simulation objective within an allowable error bound. Computational efficiency involves the satisfaction of the real-time requirements to simulate the system, in addition to the efficiency of model computation. In existing applications, it is a user's responsibility to construct the model appropriate for the simulation task.

5. Case Study: Small Nuclear Fusion Device Construction

Many fusion-related plasma experiments use magnetic field confinement to contain the charged plasma. A toroidal device called a Tokamak, which is first developed in Russia, is the most successful device yet found for magnetic confinement of plasma [18], [19]. In this device a combination of two magnetic fields is used to confine and stabilize the plasma, a strong toroidal field (TF), is produced by the current in the windings, and a weaker "poloidal" field, is produced by the toroidal current (current in the plasma). In addition to confining the plasma, the toroidal current is used to heat it. The resultant helical field lines spiral around the plasma and keep it from touching the walls of the vacuum chamber. There are many small fusion experiments used as hardware simulator for large fusion projects like (TEXT, TEXTOR, ASDEX) [20]. The Egyptor tokamak is one from these experiments [21-23].

The Egyptor is a small tokamak device of noncircular cross section (R=30 cm, a=10 cm) intended primarily to study of plasma-wall interaction. Figure (1) shows a block diagram for Egyptor tokamak. Studies can be carried out in this small machines , the results of these studies are key in making predictions for larger devices [24]. The detailed balance of (particle production and loss), and of (power input and loss) determine the state of the plasma in a tokamak. The particle balance equation [25] is basically single-ion-species plasma, setting $n_e = n_i = n$, is given by

$$\frac{\partial n}{\partial t} = n n_n \langle s v \rangle_i + \frac{1}{r}\frac{\partial}{\partial r}\left[r D_A \frac{\partial n}{\partial r} \right] \quad (\mathrm{m^{-3}.s^{-1}}) \tag{3}$$

where $< sv >_i$ is ionization rate of the neutral particles that fuel the plasma, n_e is the electron density, n_i is the ion density, n is the particle density, n_n is the neutral density, r is the minor radius, and D_A is the diffusion coefficient. At steady state the above equation is equal to 0, which can be stated in the following form:

$$\frac{d}{dr}[(\frac{r}{n})(\frac{dn}{dr})] + [\frac{e^2 V_e^2}{K(T_i + T_e)}] nr = 0 \tag{4}$$

where $\boldsymbol{T_i}$ and $\boldsymbol{T_e}$ are ion and electron temperatures, **e** is electron charge, **K** Boltzman's constant, and $\boldsymbol{V_e}$ is electron velocity. Substituting $[(e^2 V_e^2)/K(T_i + T_e)] = b$, the equation becomes

$$\frac{d}{dr}[(\frac{r}{n})(\frac{dn}{dr})] + bnr = 0 \tag{5}$$

The power balance may be written separately for each species [25] as following.

i- For the electrons, the local power balance is given by

$$\frac{\partial}{\partial t}\left[\frac{3}{2} n_e e T_i\right] = \frac{1}{r}\frac{\partial}{\partial r} r\left[n_e e c_e \frac{\partial T_e}{\partial r} + \frac{3}{2} T_e e D_A \frac{\partial n_e}{\partial r} \right] + P_{ie} - P_b - P_R - P_c + P_\Omega + P_{ae} + P_{ae} (\mathrm{w.m^{-3}}) \tag{6}$$

ii- For the ions, the local power balance is given by

$$\frac{\partial}{\partial t}\left[\frac{3}{2} n_i e T_i\right] = \frac{1}{r}\frac{\partial}{\partial r} r\left[n_i e c_i \frac{\partial T_i}{\partial r} + \frac{3}{2} T_i e D_A \frac{\partial n_i}{\partial r} \right] + P_{ei} - P_{cx} + P_{ai} + P_{ai} (\mathrm{w.m^{-3}}) \tag{7}$$

The parameters c_e and c_i are the electron and ion thermal diffusivities, respectively, and $\boldsymbol{P}$ ($\mathrm{W.m^{-3}}$) is the power density of the various mechanisms indicated by the subscripts:

ie ion-electron collision, *W* ohmic heating, ***b*** bremsstrahlung radiation, *a* alpha power, ***R*** radiation, ***a*** auxiliary power, ***c*** cyclotron radiation, ***cx*** charge exchange, and ***ei*** electron-ion collision.

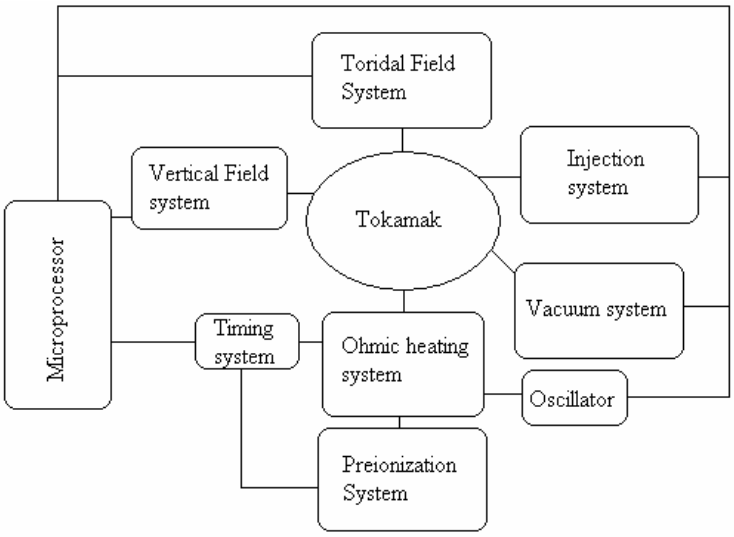


Figure 1 Block Diagram for Egyptor Tokamak

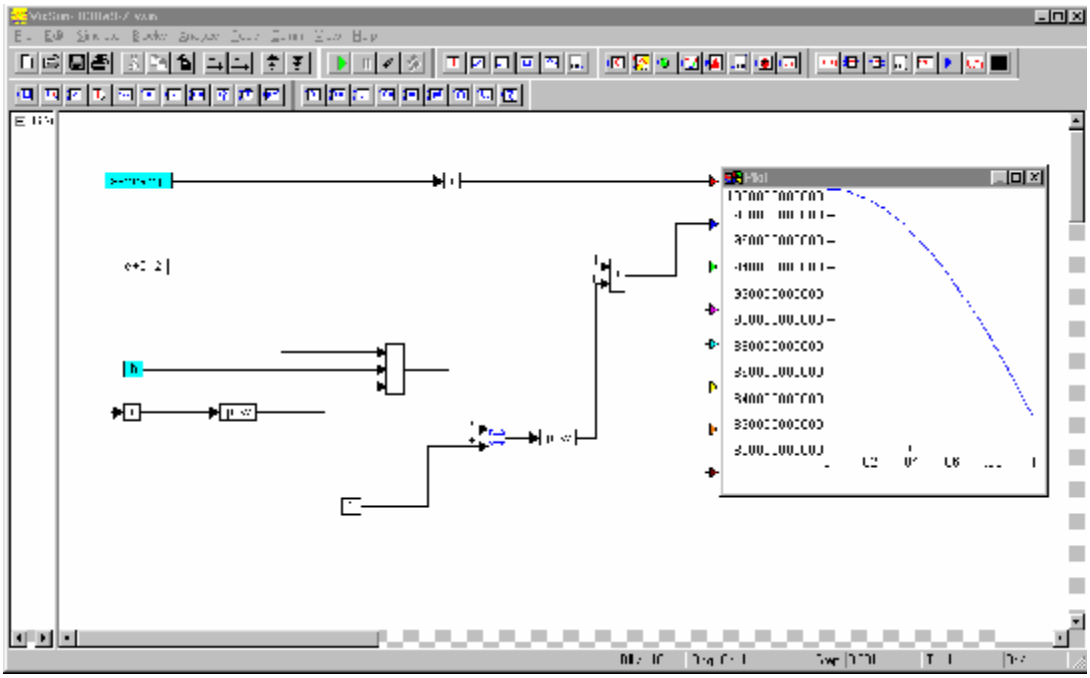


Figure 2-a. Implemented Simulator Program for Bennett Distribution.

6. Proposed Simulator for Tokamak Device

A proposed program for modeling and simulation of Tokamak is designed and implemented in **VisSim** environment, where **VisSim** is a visual block diagram language for nonlinear dynamic simulation. A block API allows users to create their own blocks in C/C++. Add-ons allow real-time analog and digital I/O for real-time simulation, embedded system C code generation, optimization, neural nets, OPC, frequency domain analysis, scaled fixed point, IIR and FIR filter design. In this environment, the system is modeled by the graphical interconnection of function blocks [26]. For flexibility, variables are used to denote system parameters and then are assigned values in a separate compound block. Data analysis is also

included in the program. The program can be distributed with **VisSim** viewer or through generated C code from **VisSim** block diagram, which means it does not depend on the **VisSim** environment. Differentiation and Integration Blocks are adjusted to suit the nature of the Tokamak equations, which have two variables radius (r) and time (t) as described in the pervious section. Equation (5) is modeled using VisSim environment at first. Secondly, the diffusion rate is taken into account in the model. For comparison purposes, the Bennett distribution of the form $n = n_0/(1+n_0*b*r^2)^2$ [27], which is applied as an approximated solution of the particle balance equation is also modeled for verification purposes of the simulator. Figure (2-a) shows the implemented simulator program for Bennett distribution and figure (2-b) gives Bennett distribution.

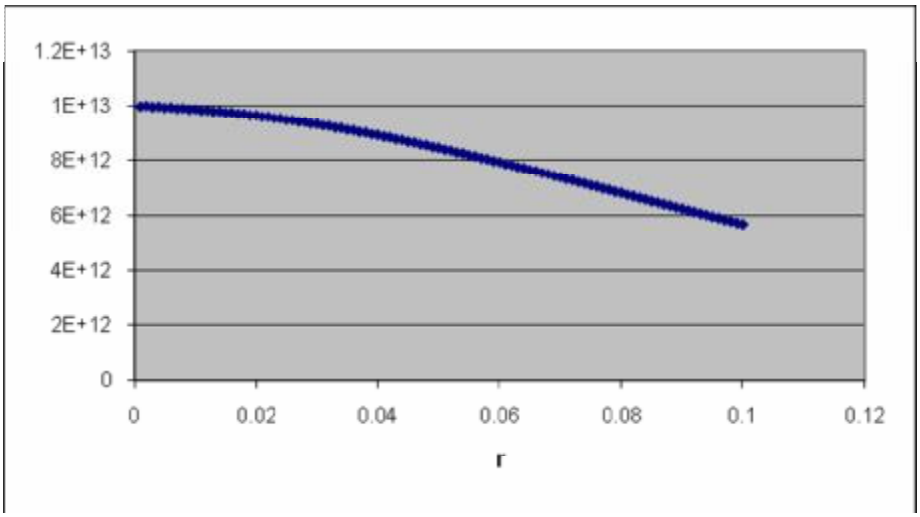


Figure 2-b. Bennett Distribution

7. Simulation Results

In this section the simulation results of our case study Small Nuclear Fusion Device (Egyptor Tokamak) using ViSim is presented. Figure (3-a) shows the implemented simulator program for power balance equation (ion temperature), and figure (3-b) shows radial ion temperature distribution.

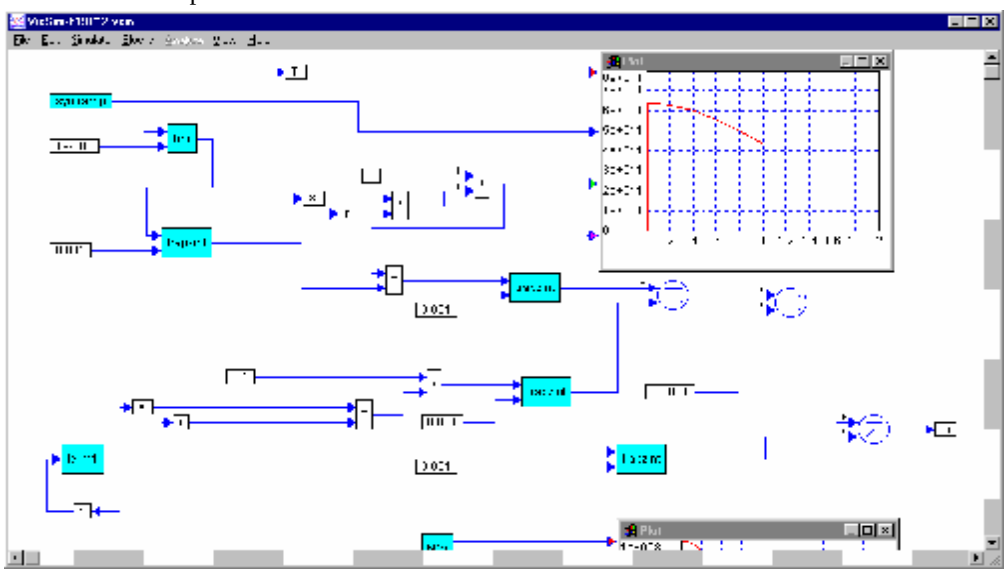


Figure 3-a. Implemented Simulator Program for Ion Temperature.

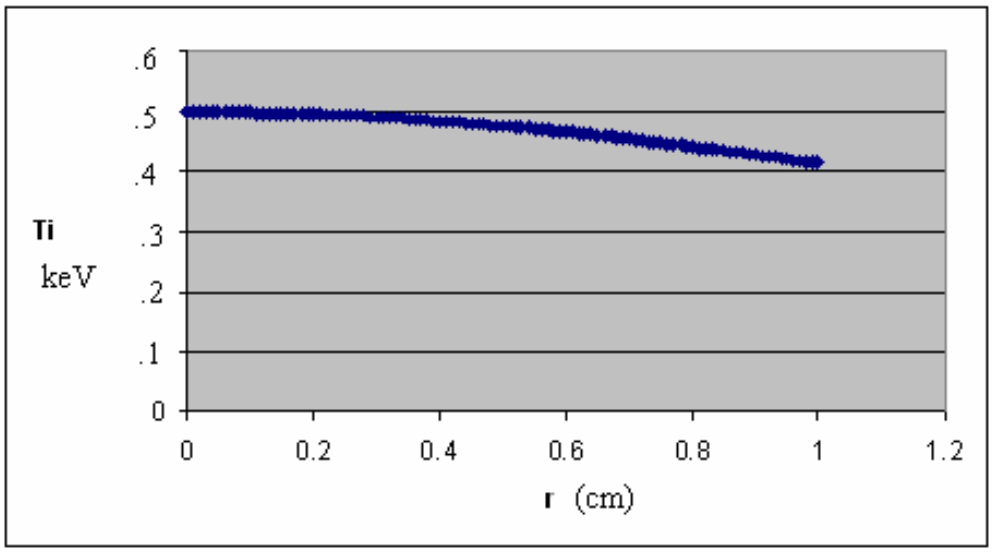


Figure 3-b. Radial Ion Temperature Distribution.

Finally figure (4-a) the implemented simulator program for particle balance equation is given, and figure (4-b) shows the radial density distribution inside plasma. The height and width of curves are controlled by Tokamak parameters (b and internal parameters in figure (2-a)).

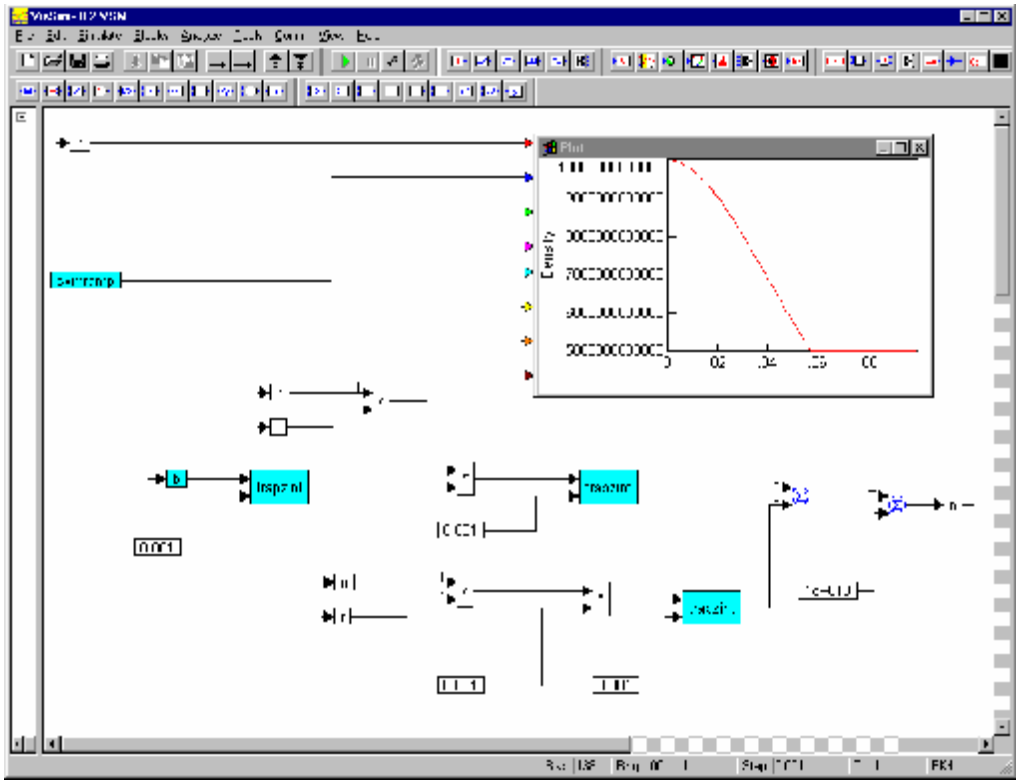


Figure 4-a. Implemented Simulator Program for Particle Balance.

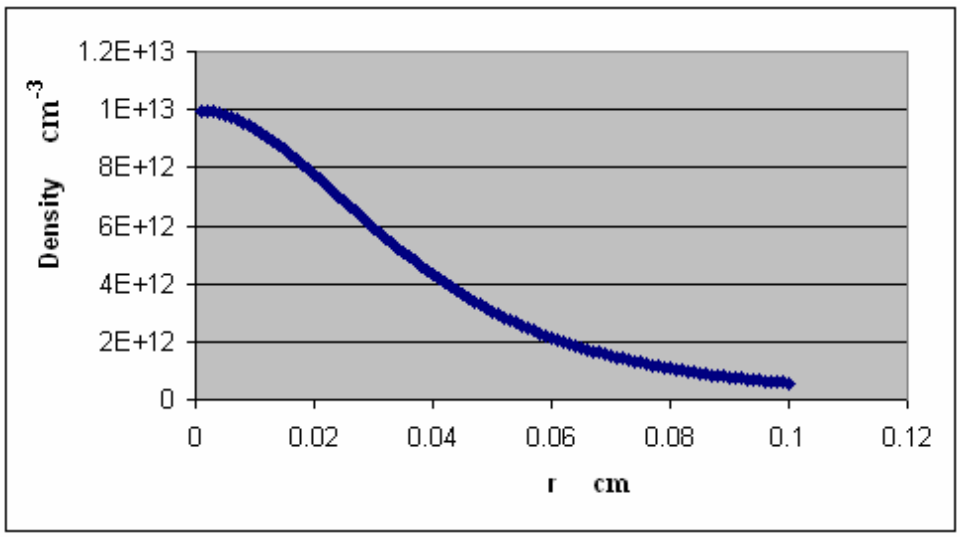


Figure 4-a. Implemented Simulator Program for Particle Balance.

Comparing with published experimental and analytical results in [28] as shown in figure (5), we obtain good agreement. Moreover, any enhancements can be easily added to the simulator in a block diagram form. Simulation of the Tokamak system can be used in conjunction with control system of Tokamak [22] using the same software. Hence, Model Predictive Control (MPC), a method which continuously updates the controller and is able to predict and act during power supply saturation can be applied. In this case, we used the real-time mode for hardware-in-the-loop control part of the whole system. Simulating in real-time mode has the effect of retarding a simulation so that one simulation second equals one second in real time.

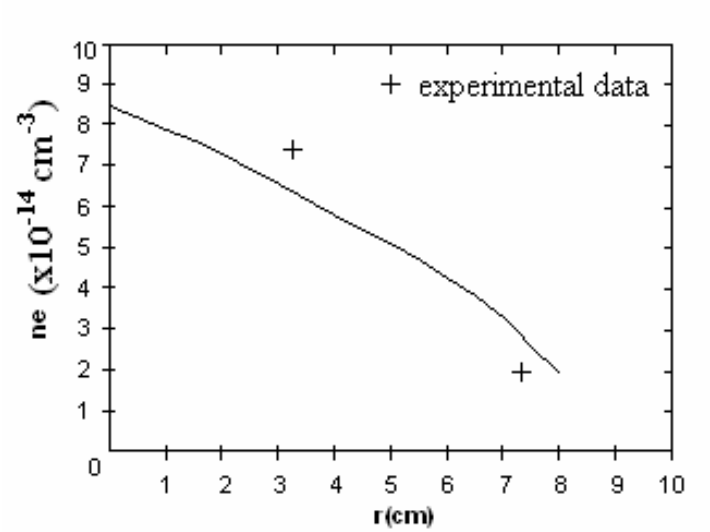


Figure 5. The electron density profile in [28].

8. Conclusions

In this paper studied a real time modeling system for dynamic systems has been focused for the software real time modeling techniques and systems. A simulator for Tokamak Egyptor is designed and implemented. It allows the introduction of the techniques of PC interfacing with industrial and scientific systems using easy to manipulate environment (e.g. block diagram method) to engineers involved in the nuclear field. The obtained results of comparison with published

work is good. Using the same environment, Model Predictive Control (MPC) methods can be used. Moreover, any enhancement to the model can be added easily due to the nature of block diagram programming.

9. References

[1] Kangsun Lee, Paul A. Fishwick, "OOPM/RT: A Multimodeling Methodology for Real-Time Simulation", ACM Transactions on Modeling and Computer Simulation, Vol. 9, No. 2, April 1999, pp. 141–170.

[2] J. M. Giron-Sierra, J. A. Gomez-Pulido, B. Andres-Toro, "X-Windows Simulation of Steam Power Plants Based on Physics Principles" Proceedings of the Winter Simulation Conference, 1993.

[3] T. G. Kirner, C. Kirner, "Simulation of Real-Time Systems: An Object-Oriented Approach Supported by a Virtual Reality-Based Tool", IEEE computer society, Proceedings of the 38th Annual Simulation Symposium 2005.

[4] W. B. Rouse, "Human-Computer Interaction in the Control of Dynamic Systems", Computing Surveys, Vol. 13, No. 1, March 1981

[5] R. Crosbiea, J. Zenora, R. Bednara, and D.Worda, "High-Speed, Scalable, Real-Time Simulation Using DSP Arrays", IEEE computer society,Proceedings of the 18th Workshop on Parallel and Distributed Simulation, 2004, pp. 52-59.

[6] R. Vincent, B. Horling, V. Lesser, and T. Wagner, "Implementing Soft Real-Time Agent Control", *AGENTS'01,* Montr´eal, Quebec, Canada, May 28-June 1, 2001.

[7] H. Kopetz, "Software *Engineering* for Real-Time:A Roadmap", ACM 2000, pp. 203-211.

[8] S. Abourida, C. Dufour, J. Bélanger, G. Murere, N. Léchevin, and B. Yu "Real-Time PC-Based Simulator of Electric Systems and Drives", APEC 2002, pp. 433-438.

[9] B.G. Penaflor, et al, "Real-time control of DIII-D plasma discharges using a Linux alpha computing cluster", Fusion Engineering and Design, 56-57, 2001, pp. 739-742.

[10] J. Sousa, et al, "A distributed real-time system for event-driven control and dynamic data acquisition on a fusion plasma experiment", Fusion Engineering and Design, 48, 2000, pp. 31-36,.

[11] J.A. Romero, et al., "Real time profile control at JET", Fusion Engineering and Design, 43, 1998, pp. 37-58,.

[12] K. Nishimura, et al, "Real-time plasma control system for LHD", Fusion Engineering and Design,39-40, 1998, pp. 169-172,.

[13] Lecture Notes, "Seventh College on Microprocessor-Based Real-Time Systems in Physics", Abdus Salam ICTP, Trieste, Italy, 2002.

[14] M. Dinkel, U. Baumgarten, "Modeling Nonfunctional Requirements: A Basis for dynamic Systems Management", ACM, 2005.

[15] A.Abreu and LuisCustodio Carlos Pinto-Ferreira, "Fuzzy Modeling: a Rule Based Approach", Proceedings of the fifth IEEE International Conference on Fuzzy Systems, 1996, pp. 162-168.

[16] C.-W. Xu, Y.-Z. Lu, "Fuzzy model Identification and self learning for dynamic systems", IEEE Transactions on Systems, Man and Cybernetics, pp. 683-689, 1987.

[17] C.-C. Wong, N.-S. Lin, "Rule extraction for fuzzy modeling", Fuzzy Sets and Systems, no. 88, 1997, pp. 23-30.

[18] Artsimovitch L. A., Nuclear Fusion, 12, 215 (1972).

[19] Kadomtseve B. B. "Tokamak Plasma: A Complex Physical System" Translation editor P.Ewlaing, Institute of Physics Publishing Bristol and Philadelphia (1992).

[20] Wesson J. with contributions " TOKAMAK" Clarendon Press Oxford (1987).

[21] A. El-Sisi and M. Masoud, " Egyptor Tokamak Reconstruction, Development and Operation" *Internal Report, SIDC*, EAEA, 2001.

[22] R. Flohr t. al., "The Tokamak Experiment " UNITOR"-A Device for Plasma-Wall Interaction Research" Physica 104C, 423-433, 1981.

[23] A. B. El Sisi; H. Hegazy "EGYPTOR Tokamak: Modification of the Original Design Using Permanent Compensation Coils and First Results of the Breakdown Discharge, Journal of Fusion Energy, Volume 22, Number 3, September 2003, pp. 191-194(4).

[24] Annual Report of the EURATOM/UKAEA Fusion Programme 1997/98.

[25] J. Sheffield , "Tokamak Start-Up," Edited by H. Knoepfel, Plenum, New York 1986.

[26] Imbaby I. Mahmoud and S. A. Kamel, " Using a Simulation Technique for Switched-mode High Voltage Power Supplies Performance Study", *IEEE Trans. IAS*, Vol. 34, No. 5, Sept.-Oct. 1998, pp.945-952.

[27] Masoud M. M., Ph. D Thesis, Faculty of Science, Cairo Univeresty, 1971.

[28] M. H. Hughes, "Numerical Calculations of Transport in ALCATOR" Princeton Plasma Physics Lab. PPPL-1411, Jan., 1978.

A Combined Particle Swarm Optimization Algorithm Based on the Previous Global Best and the Global Best Positions

Mahmoud M. El-Sherbiny
Operations Research Dept, Institute of Statistical Studies and Research (ISSR),
Cairo University, Egypt.
Email. m_sherbiny@yahoo.com

Abstract

This paper introduces a combined algorithm to particle swarm based optimization and discusses the results of experimentally comparing the performances of its three versions with the performance of the particle swarm optimizer. In the combined algorithm, each particle flies and is attracted toward a new position according to its previous best position and the point resulted from the combination of the previous global best position and the global best position. The variants of the combined algorithm and the particle swarm optimizer are tested using a set of multimodal functions commonly used as benchmark optimization problems in evolutionary computation. Results indicate that the algorithm is highly competitive and can be considered as a viable alternative to solve the optimization problems.

Keywords*: Particle swarm optimization; Convergence; Evolutionary computation;*

1. Introduction

The particle swarm algorithm, which is frequently called particle swarm optimizer, is a new evolutionary algorithm, where the population is now called a swarm and each individual is called a particle [1]. It is inspired by the behavior of bird flocking and fish schooling. A large number of birds or fish flock synchronously, change direction suddenly, and scatter and regroup together. Each particle benefits from the experience of its own and that of the other members of the swarm during the search for food.

Particle Swarm Optimization (PSO) algorithm was proposed by Eberhart and Kennedy in 1995 [1], and had been applied to evolve weights and structure for artificial neural networks by Shi and Eberhart in 1998 [2], manufacture and milling by Tandon in 2000 [3], reactive power and voltage control by Abido M.A. in 2002 [4] and Jiang Chuanwenet in 2005 [5], and state estimation for electric power distribution systems by Shigenori et al. in 2003 [6]. The convergence and parameterization aspects of the PSO have also been discussed in [7, 8, 9].

PSO has been successfully used as an alternative to other evolutionary algorithms in the optimization of D-dimensional real functions. Particles move in a coordinated way through the D-dimensional search space towards the optimum of the function. Their movement is influenced not only by each particle own previous experience, but also by a social compulsion to move towards the best position found by its neighbours. To implement these behaviours, each particle is defined by its position and velocity in the search space. In each iteration, changes resulting from both influences in the particle's trajectory are made to its velocity. The particle's position is then updated accordingly to the calculated velocity. The PSO, its main variants and the structural model behind it are extensively discussed in [10].

This paper aims to introduce a combined algorithm of PSO and discuss the results of experimentally comparing the performance of its versions with the particle swarm optimizer (PSO)[8]. In the combined algorithm, each particle flies and is attracted toward a new position according its own best position and the point resulted from the combination of the the point resulted from the combination of the previous global best position and the global best position.

The rest of the paper is organized as follows: in section 2, the PSO method is described. In section 3, the combined algorithm and its versions are exposed. Test functions and test conditions are presented in sections 4 and 5. In section 6, optimization test experiments are illustrated. In section 7, the experimental results are reported, and are discussed in section 8. Finally, conclusion is reported in section 9.

2. Particle Swarm Optimization

The particles evaluate their positions relative to a goal (fitness) at every iteration, and particles in a local neighborhood share memories of their "best" positions, then use those memories to adjust their own velocities and positions as shown in equations (1) and (2). The PSO formula define each particle as a potential solution to a problem in the *D*-dimensional space, with the i^{th} particle represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle also remembers its best position, designated as X_{p_i}, and its velocity $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ [11].

In each generation (iteration) *t*, the velocity of each particle is updated, being pulled in the direction of its own best position (X_{p_i}) and the best of all positions (X_g) reached by all particles until the preceding generation. After finding the two best values, the particle updates its velocity and positions according to equations (1) and (2).

$$V_i(t) = aV_i(t-1) + b_1 r_1 \left(X_{p_i} - X_i(t)\right) + b_2 r_2 \left(X_g - X_i(t)\right) \quad \textbf{(1)}$$

$$X_i(t) = cX_i(t-1) + dV_i(t) \quad \textbf{(2)}$$

At iteration *t*, the velocity $V_i(t-1)$ is updated based on its current value affected by a momentum factor *a* and on a term which attracts the particle towards previously found best positions: its own previous best position (X_{p_i}) and globally best position in the whole swarm (X_g). The strength of attraction is given by the average of the own and the social attraction coefficients b_1 and b_2. The particle position $X_i(t)$ is updated using its current value and the newly computed velocity $V_i(t)$, affected by coefficients *c* and *d*, respectively and they can be set to unity without loss of generality [8]. Randomness useful for good state space exploration is introduced via the vectors of random numbers r_1 and r_2. They are usually selected as uniform random numbers in the range [0, 1].

The original PSO formula developed by Kennedy and Eberhart [1] were combined by Shi and Eberhart [2] with the introduction of an inertia parameter, ω, that was shown empirically to improve the overall performance of PSO.

Several interesting variations of the PSO algorithm have recently been proposed by researchers in [12], [13], [14], [15], [16], [17]. Many of these PSO improvements are essentially extrinsic to the particle dynamics at the heart of the PSO algorithm and can be applied to augment the new algorithm presented in this paper. By contrast to most other PSO variations, this paper proposes a significant modification to the dynamics of particles in PSO, moving each particle towards a new position according its own best position and the point resulted from the combination between the previous global best position and the global best position instead of the global best position that used in the standard particle swarm. This is in addition to the terms in the original PSO update equations.

3. The Combined Algorithm

In the standard PSO algorithm, in each generation *t*, the velocity of each particle is updated, being pulled in the direction of its own previous best position (X_{p_i}) and the best of all positions (global position) (X_g) reached by all particles until the preceding generation. Whereas in the combined PSO algorithm, in each generation *t*, the velocity $V_i(t-1)$ of particle *i* is updated based on its own best position (X_{p_i}) and the point (X_c) resulted from the combination of the global best position (X_g) and the previous global best position (X_{2g}) as illustrated in equation (3). Where R_1 and $R_2 \in [0,1]$ are uniform random variables defined as the combination weights.

$$(X_c) = R_1 (X_g) + R_2 (X_{2g}) \quad \textbf{(3)}$$

In other words, after finding a new global best position for the (X_g) its old position will be assigned to (X_{2g}) and the particle updates its velocity according to equation (4) and updates its positions according to equation (2).

$$V_i(t) = aV_i(t-1) + b_1 r_1 \left(X_{p_i} - X_i(t)\right) + b_1 r_1 \left(\left(R_1 X_g + R_2 X_{2g}\right) - X_i(t)\right) \quad \textbf{(4)}$$

In order to study the effects of the parameters R_1 *and* R_2 in (4) on the performance of the Combined Particle Swarm Optimization algorithm (CPSO), three variants were used in the experiments denoted as CPSO1, CPSO2, and CPSO3.

In the CPSO1 version, the particle updates its velocity according to equation 3 with equal random weight combination between the global best position (X_g) and the previous global best position (X_{2g}). i.e. $R_1 = R_2 = R$.

In the CPSO2 version, the particle updates its velocity according to equation 3 with random weight combination between the global best position (X_g) and the previous global best position (X_{2g}). i.e. R_1 and R_2 are two different random variables.

In the CPSO3 version, the particle updates its velocity according to equation 3 with random linear combination between the global best position (X_g) and the previous global best position (X_{2g}). i.e. $R_2 = (1- R_1)$.

4. Test Functions and Conditions

In order to know how competitive the combined algorithm is and the effects of the combination weights R_1 and R_2 , we decided to compare its three versions against the PSO algorithm that is represented in [8]. Five benchmarking functions were selected to investigate the performance of the three versions CPSO algorithm and PSO. The considered test functions were used in [6], [7] and [8]. The functions, the number of dimensions (D), the admissible range of the variable (x), and the goal values are summarized in Table 1.

Two parameter sets (Eqs. (1), (2) and (4)) a and $b = b_1 = b_2$ were selected to be used in the test based on the suggestions in other literature where these values have been found, empiricaly, to provide good performance [10, 7, 9]. and used in testing the PSO by I.C. Trelea [8].

Parameter set 1 (a = 0.6 and b = 1.7) was selected by the author in the algorithm convergence domain after a large number of simulation experiments [7].

Parameter set 2 (a = 0.729 and b = 1.494) was recommended by Clerc [18] and also tested in [7] giving the best results published so far known to the author. All elements of c and d were set to 1 as used in [8].

A more detailed study of convergence characteristics for different values of these parameters exists in [19].

5. Optimization Test Experiments

In order to test the performance of the three versions of CPSO and PSO algorithms two sets of experiments were used with the above mentioned test conditions and the two parameter sets.

In the first set of experiments, the maximum iteration number was fixed to 2000. Each optimization experiment was run 20 times with random initial values of x and v in the range $[x_{min}, x_{max}]$ indicated in Table 1. Population sizes of N = 15, 30 and 60 particles were tested. The number of iterations required to reach the goal was recorded. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations, for each test function are calculated and presented in Tables 2-6.

In the second set of experiments, Each optimization experiment was run 20 times for 1000 iterations with population sizes of N = 30 particles. The averages of the best values in each iteration were calculated and plotted in figures 1- 5.

During the optimization process the particles were allowed to "fly" outside the region defined by $[x_{min}, x_{max}]$ and also the velocity was not restricted.

6. The Experimental Results

This section compares the various algorithms to determine their relative rankings using both robustness and convergence speed as criteria. A "robust" algorithm is one that manages to reach the goal consistently (during all runs) in the performed experiments [20]. Tables 2–6 present the following information: Average number, median, minimum, maximum number of iterations required to reach a function value below the goal. Also, success rate of required iterations, and expected number of function evaluations. The "success rate" column lists the number of runs (out of 20) that managed to attain a function value below the goal in less than 2000 iterations, while the "Ex. # of Fn. Evaluation" column presents the expected number

of function evaluations needed on average to reach the goal, calculated only for the succeeded runs using the following formula.

Ex. # of Fn. Evaluation = (Average number of iterations) x (number of particlsin the swarm)/ (success rate)

Table 2 shows that the CPSO1 and CPSO2 algorithms reached the goal during all the runs for solving the Sphere function (F_0) with both parameter sets. While CPSO3 and PSO algorithms failed to reached the goal during some runs with parameter set 1.

Table 1. Test functions [8]

Name	Formula	Dim. D	Rang [xmin, xmax]	Goal for F
Sphere	$F_0(\vec{x}) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]^D$	0.01
Rosenbrock	$F_1(\vec{x}) = \sum_{i=1}^{D-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	30	$[-30, 30]^D$	100
Rastrigin	$F_2(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	30	[-5.12, 5.12]D	100
Griewank	$F_3(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^D$	0.1
Schaffer's	$F_6(\vec{x}) = 0.5 - \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	2	$[-100, 100]^2$	10^{-5}

The optimal solution for all functions is equal to 0

Also, as illustrated in figure 1, all the algorithms, except CPSO3 and PSO with the parameter set1 and 15 particles, have reached a function value below the goal of the sphere function with both parameter sets. That means the number of particles affects the convergancy of the CPSO3 and PSO algorithms for such problems type.

None of the algorithms, with the exception of the CPSO3 with both parameter sets and 15 particles and PSO with parameter set2 and 15 particles, had any difficulty reaching the goal of the Rosenbrock function (F_1) during any of the runs. Table 3 shows the expected number of function evaluations for all the algorithms required for solving the Rosenbrock function (F_1) with the CPSO1 algorithm requiring the fewest function evaluations overall. Also, fig. 2 illustrates that the CPSO1 and CPSO2 reached a value below the function goal while CPSO3 and PSO stacked near the function goal.

Table 4 shows that the CPSO1 and CPSO2 algorithms perform admirably on the Rastrigin function (F_2), but the CPSO3 and PSO algorithms are less robust on the same function.

Note that the CPSO1 algorithm is doing very well on this problem, delivering the best overall performance for the Rastrigin function where it reached the goal on approximately 60 iterations and reached the optimal solution in approximately less than 400 iterations as illustrated in fig. 3.

Concerning the effects of the parameter sets on the algorithms performance on Rastrigin function (F_2), there is no significant difference between the algorithms performance with both the parameter sets except the performance of CPSO2 algorithm with parameter set1 is a little mach better than its performance with parameter set2 as shown in figure 3.

Table 2. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations, for the test function F_0

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F_0	15	CPSO1	125	168	126	173	69	102	161	216	1	1	1874	2516
		CPSO2	320	471	316	457	249	358	373	613	1	1	4804	7065
		CPSO3	632	720	635	699	461	500	903	1313	0.45	1	21070	10796
		PSO	769	764	722	731	523	586	1377	1275	0.40	1	28838	11460
	30	CPSO1	131	180	127	179	103	137	161	221	1	1	3917	5396
		CPSO2	300	404	296	396	259	296	352	528	1	1	9006	12126
		CPSO3	311	358	306	360	246	315	499	434	1	1	9323	10739
		PSO	344	395	333	395	266	330	457	572	1	1	10320	11850
	60	CPSO1	118	157	120	159	90	123	132	185	1	1	7083	9423
		CPSO2	264	346	254	346	221	301	302	389	1	1	15816	20760
		CPSO3	224	281	222	285	197	245	254	310	1	1	13431	16854
		PSO	252	314	252	313	214	269	309	368	1	1	15120	18840

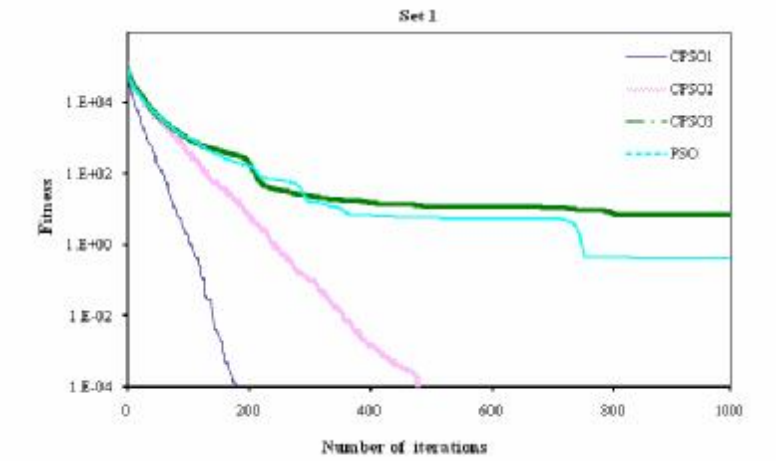


Figure 1. Average best fitness curves for sphere function (Fo)

Griewank's function (F_3) proves to be hard to solve for all the algorithms except CPSO1 CPSO2 algorithms, as can be seen in Table 5. Only the CPSO1 and CPSO2 algorithms consistently reached the goal during all runs with both parameter sets and they are candidate to reach the optimal solution (see fig. 4) while PSO and CPSO3 did not reach the goal during some runs (see Table 5).

The CPSO3 failed almost completely to reach the goal for solving Griewank function (F_3) with parameter set1 and 15 particles, as can be seen in Table 5 while it reached the goal in almost 11 out of 20 runs. Fig. 4. illustrates the effects of the parameter sets on the performance of CPSO3 and PSO algorithms. Note that both the CPSO3 and PSO algorithms failed almost completely to reached the goal with parameter set1 in 1000 runs but they did with parameter set2 in less than 350 runs, while the CPSO1 and CPSO2 algorithms managed to solve the same function consistently with both the parameter sets. That means CPSO3 and PSO algorithms are very fast in algorithming the solution but making a bridging – zigzagging- near the optimal solution.

Table 3. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations, for the test function F1

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F_1	15	CPSO1	38	112	39	105	52	63	119	160	1	1	1318	1673
		CPSO2	288	484	276	463	199	281	516	760	1	1	4324	7260
		CPSO3	587	753	490	666	325	416	1147	1403	0.50	0.80	17601	14118
		PSO	531	1430	523	729	413	452	695	9476	0.50	1	15930	21450
	30	CPSO1	84	105	81	105	57	74	110	127	1	1	2520	3149
		CPSO2	257	392	258	395	204	253	336	771	1	1	7701	11771
		CPSO3	450	477	401	417	212	249	961	1051	1	1	13500	14315
		PSO	614	900	383	408	239	298	3718	4642	1	1	18420	27000
	60	CPSO1	77	102	76	101	54	80	89	124	1	1	4623	6138
		CPSO2	227	309	218	298	174	234	468	430	1	1	13590	18561
		CPSO3	263	356	230	280	170	215	531	775	1	0.85	15768	25138
		PSO	337	611	284	311	189	219	916	4450	1	1	20220	36660

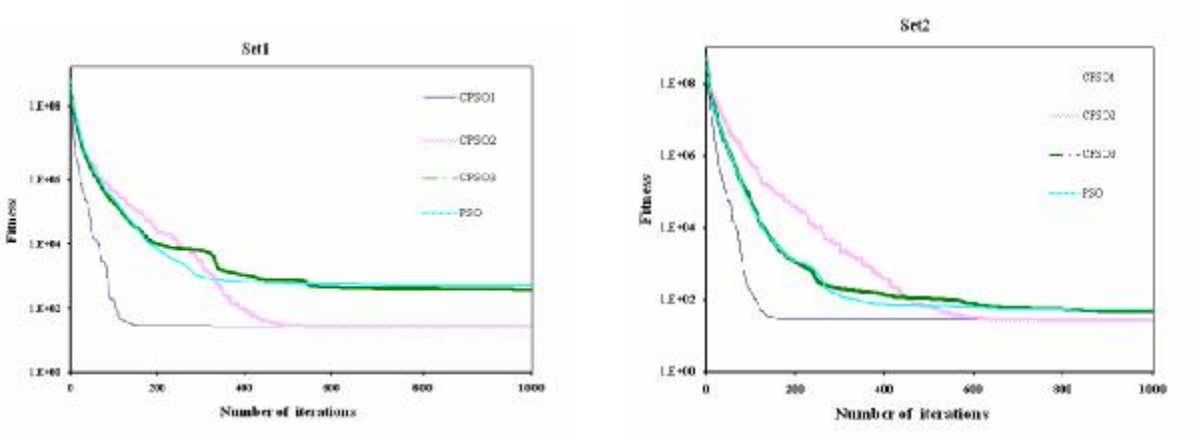


Figure 2. Average best fitness curves for Rosenbrock function (F1)

Table 4. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations , for the test function F_2

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F_2	15	CPSO1	57	81	51	71	23	47	139	147	1	1	857	1217
		CPSO2	264	548	220	517	157	145	698	1357	0.90	1	4402	8214
		CPSO3	122	169	120	156	80	105	194	309	0.85	1	2145	2671
		PSO	172	299	147	292	102	123	208	299	0.35	0.8	7371	5606
	30	CPSO1	48	68	44	63	32	37	83	145	1	1	1451	2054
		CPSO2	281	443	233	348	121	159	857	955	0.95	1	8862	13283
		CPSO3	106	155	100	148	66	104	192	212	0.90	1	3522	4641
		PSO	140	182	128	174	104	123	208	299	0.90	0.95	4667	5747
	60	CPSO1	60	69	55	62	39	34	132	127	1	1	3603	4152
		CPSO2	265	517	209	414	144	215	633	1526	1	1	15906	31026
		CPSO3	91	127	87	120	62	80	136	193	1	1	5457	7614
		PSO	122	166	116	164	84	119	168	214	0.95	1	7705	9960

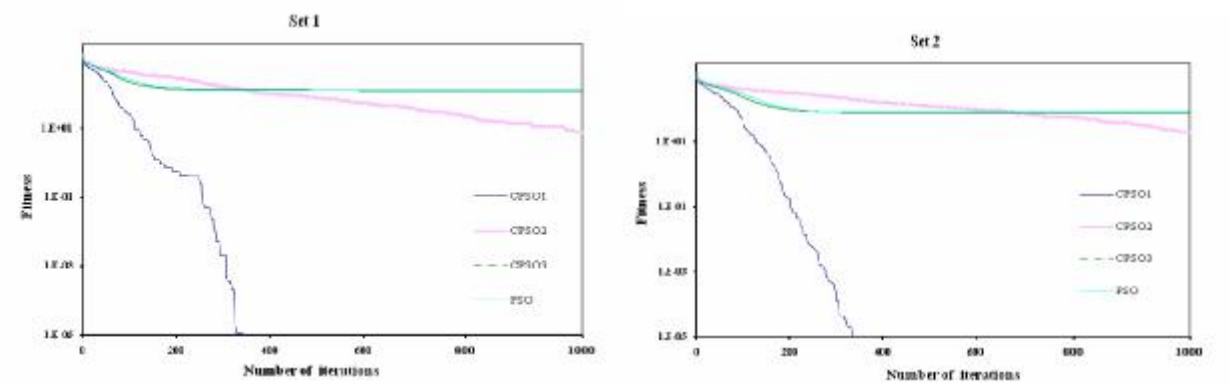


Figure 3. Average best fitness curves for Rastrigin function (F2)

Concerning the Schaffer's function (F_6): Table 6 illustrates that only the CPOS1 and CPSO2 algorithms reached the goal in all runs with both parameter sets, while CPSO3 and PSO algorithms had some difficulties in reaching the goal.

Table 5. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations, for the test function F_3

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F_3	15	CPSO1	132	215	131	205	84	89	191	470	1	1	1982	3218
		CPSO2	364	581	314	522	254	329	731	1403	1	1	5465	8714
		CPSO3	-	508	-	487	0	438	0	702	-	0.55	-	13845
		PSO	689	755	580	608	443	470	1589	1755	0.35	0.6	29529	18875
	30	CPSO1	131	168	130	165	79	57	229	293	1	1	3929	5046
		CPSO2	342	440	293	431	227	330	720	667	1	0.95	10257	13910
		CPSO3	266	312	265	301	235	270	327	505	0.85	0.90	9386	10387
		PSO	313	365	304	361	257	319	401	455	0.90	0.90	10433	12167
	60	CPSO1	113	152	109	148	72	115	199	198	1	1	6753	9117
		CPSO2	325	421	268	370	188	301	353	718	1	1	19494	25245
		CPSO3	211	251	208	248	178	216	240	286	1	1	12681	15084
		PSO	226	287	224	280	202	266	250	238	0.95	1	14274	17220

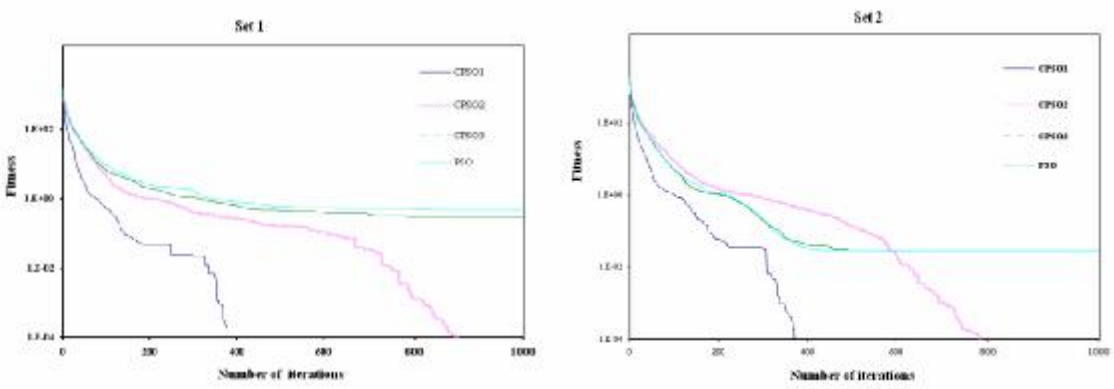


Figure 5. Average best fitness curves for Schaffer function F6

Note that, only the MPOS1 and CPSO2 algorithms reached below the goal and the optimal values in less than 350 iterations on average with both parameter sets while CPSO3 and PSO algorithms had stacked before the goal with both parameter sets in 1000 iterations (see fig. 5).

7. Discussion

Overall, as far as robustness is concerned, the CPSO1 algorithm appears to be the winner, since it achieved a perfect score in all the test cases as represented in boldface (see Tables 2-6).

The CPSO2, algorithm is less robust, followed closely by the CPSO3 and PSO algorithms. The standard PSO algorithm was fairly unreliable on this set of problems.

As a result, the PSO must be executed several times to ensure good results, whereas one run of CPSO1 and the CPSO2 usually sufficient.

Note that in the set1 case, there is a little difference between the performance of the algorithms with parameter set1 and with parameter set2 of where the algorithms' conversances are faster with parameter set1 than with parameter set2.

CPSO3 and PSO are more sensitive to parameter changes than the other algorithms. When changing the problem, one probably needs to change parameters as well to sustain optimal performance.

Regarding convergence speed, CPSO1 is always the fastest followed by CPSO2, whereas the CPSO3 or PSO are always the slowest. Especially on the all functions, CPSO1 has a very fast convergence (2-5 times faster than PSO). This may be of practical relevance for some real-world problems where the evaluation is computationally expensive and the search space is relatively simple and of low dimensionality.

Overall, CPSO1 is clearly the best performing algorithm in this study. It finds the lowest fitness value for most of the problems, which emphasized in boldface, see figures 1-5.

Regarding the parameter sets: in general, the performance of all algorithms are best with parameter set1 than the performance with parameter set2, while all of them need less number of iterations to reach the specified goal with set1 than with parameter set2. That means parameter set2 slows the algorithms and don’t make a bredging phenomina while parameter set1 accelerate the algorithms but somewhile make a bredging phenomina.

Looking at the number of function evaluations, the CPSO1was in the lead, followed by the CPSO2 algorithm, as shown in boldface (see Tables 1-5)

Considering, the above mentioned point that CPSO1 had no difficulty in reaching the goal and all its solutions are below their corresponding goals more than the other algorithms. So, we can conclude that CPSO1 is more superior to the other algorithms. That means we can consider it as a best alternative algorithm for solving optimization problems.

Table 6. Average number, median, minimum, maximum, and success rate of required iterations, and expected number of function evaluations, for the test function F6

Fun.	# of Part. N	Algorithm	Number of algorithm iterations to achieve the goal								Success Rate		Ex. # of Fn. Evaluation	
			Average		Median		Minimum		Maximum					
			Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2	Set1	Set2
F6	15	CPSO1	198	232	183	201	57	63	473	498	1	1	2971	3476
		CPSO2	231	254	207	225	72	99	522	631	1	1	3461	3813
		CPSO3	286	307	177	273	64	114	704	613	0.40	0.55	10720	8366
		PSO	583	1203	138	126	63	91	3706	5853	0.45	0.4	19433	45113
	30	CPSO1	122	148	98	134	55	57	307	397	1	1	3668	4434
		CPSO2	144	163	126	163	57	87	439	298	1	1	4322	4893
		CPSO3	348	401	271	234	73	111	1252	1769	0.65	0.60	16047	20025
		PSO	161	350	120	157	74	102	595	1264	0.75	0.60	6440	17500
	60	CPSO1	93	112	76	110	42	73	206	193	1	1	5559	6708
		CPSO2	112	147	94	128	52	53	232	422	1	1	6726	8841
		CPSO3	402	305	217	146	72	75	1727	1378	0.85	0.95	28385	19257
		PSO	169	319	91	119	40	83	854	2361	0.90	0.95	11267	20147

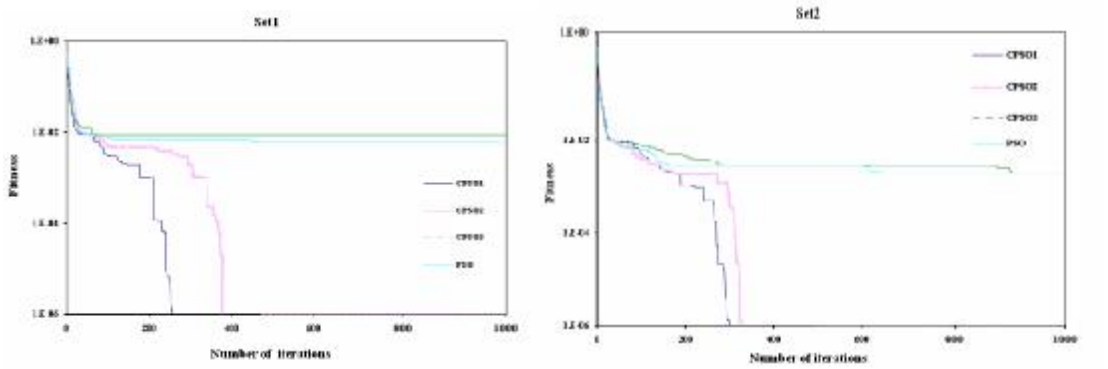


Figure5. Average best fitness curves for Schaffer function F6

8. Conclusion

This paper has proposed a new variation of the particle swarm optimization algorithm called a combined PSO, introducing a new term into the velocity component update equation: each particle is moved toward a new position according its best previous position and the point resulted from the combination of the best previous global position and the former best previous global position. The implementation of this idea is simple, based on storing the provious positions. The new algorithm outperfoms PSO on many benchmark functions, being less susceptible to premature convergence, and less likely to be stuck in local optima.

In this study, the CPSO1 has shown its worth on tested problems, and it outperformed CPSO2, CPSO3 and PSO on all the numerical benchmark problems as well. Among the tested algorithms, the CPSO1 can rightfully be regarded as an excellent first choice, when faced with a new optimization problem to solve.

To conclude, the performance of CPSO1 is outstanding in comparison to the other algorithms tested. It is simple, robust, converges fast, and finds the optimum in almost every run. In addition, it has few parameters to set, and the same settings can be used for many different problems.

Future work includes further experimentation with parameters of CPSO, testing the new algorithm on other benchmark problems, and evaluating its performance relative to Evolutionary Algorithms.

9. References

[1] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory”, in Proc. 6th Int. Symp. Micro Machine and Human Science, Nagoya, Japan, (1995) 39–43.

[2] Y. Shi and R. Eberhart, “A Combined Particle Swarm Optimizer”, In: Proceedings of IEEE World Congress on Computational Intelligence, (1998) 69–73.

[3] V. Tandon, “Closing The Gap Between CAD/CAM and Optimized CNC and Milling”, Master thesis, Purdue School of Engineering and Technology, Indiana University Purdue University Indianapolis, 2000.

[4] Abido M.A., “Optimal Power Flow Using Particle Swarm Optimization”, Electri Power and Energy Syst 2002; 24: 563–71.

[5] Jiang Chuanwen, Etorre Bompard, “A Hybrid Method Of Chaotic Particle Swarm Optimization and Linear Interior For Reactive Power Optimization”, Mathematics and Computers in Simulation 68 (2005) 57–65.

[6] N. Shigenori, G. Takamu, Y. Toshiku, F. Yoshikazu, “A Hybrid Particle Swarm Optimization For Distribution State Estimation”, IEEE Transactions on Power Systems 18 (2003) 60– 68.

[7] R. C. Eberhart and Y. Shi, “Comparing Inertia Weights and Constriction Factors In Particle Swarm Optimization” in: Proc. CEC, San Diego, CA, (2000) 84–88.

[8] Ioan Cristian Trelea, “The Particle Swarm Optimization Algorithm: Convergence Analysis And Parameter Selection”, Information Processing Letters 85 (2003) 317–325.

[9] R. Eberhart and Y. Shi, “Comparison between Genetic Algorithms and Particle Swarm Optimization”, The 7th Annual Conference on Evolutionary Programming, 1998, San Diego, USA.

[10] M. Clerc and J. Kennedy, “The Particle Swarm: Explosion, Stability, And Convergence In A Multi-Dimensional Complex Space”, IEEE Trans. Evol. Comput. 6,(2002) 58–73.
[11] A. Carlisle and G. Dozier, “Adapting Particle Swarm Optimization to Dynamic Environments”, Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, (2000) 429-434.
[12] S. Tsumoto et al. (Eds.), “A Guaranteed Global Convergence Particle Swarm Optimizer”, RSCTC, (2004) 762–767. Springer-Verlag Berlin 2004.
[13] Van den Bergh, F., Engelbrecht, A. P., “Effects of Swarm Size on Cooperative Particle Swarm Optimizers”, Genetic and Evolutionary Computation Conference, San Francisco, USA, 2001.
[14] M. Lovbjerg and T. Krink, “Extending Particle Swarm Optimizers with Self-Organized Criticality”, Proceedings of Fourth Congress on Evolutionary Computation, (2002) 1588-1593.
[15] Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang, “Hybrid Particle Swarm Optimizer with Mass Extinction”, International Conf. on Communication, Circuits and Systems (ICCCAS), Chengdu, China, 2002.
[16] Xiao-Feng Xie, Wen-Jun Zhang and Zhi-Lian Yang, “A Dissipative Particle Swarm Optimization”, IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 2002.
[17] Jacques Riget and Jakob S. Vesterstorm, “A Diversity-Guided Particle Swarm Optimizer - The ARPSO” , EVALife Technical Report no. 2002-02.
[18] M. Clerc, “The Swarm And The Queen: Towards A Deterministic and Adaptive Particle Swarm Optimization”, in: Proc. ICEC, Washington, DC, (1999), 1951–1957.
[19] F. van den Bergh, “An Analysis Of Particle Swarm Optimizers,” Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, 2002.
[20] Frans van den Bergh and Andries P. Engelbrecht, “A Cooperative Algorithm To Particle Swarm Optimization”, IEEE Transactions on Evolutionary Computation, 8(3), (2004) 225-239.

Modeling of Updating Moving Object Database Using Timed Petri net Model

F. A.Torky,
Prof. of Computer Science & Engineering and President of Kafer El-Sheekh University, Egypt
Fatorkey@yahoo.com

H. M. Abdulkader,
Lecturer of Information Systems Faculty of Computers and Information Menoufya Univ. Egypt.
Hatem6803@Yahoo.Com

N. Abd El-Wahed,
Dean of Faculty of Computers and Information, Menoufya University, Egypt
Nabil_is48@yahoo.com

Warda El-Kholy
Information Systems Faculty of Computers and Information Menoufya University Egypt
Warda_Elkoly@yahoo.com

Abstract

Tracking moving objects is one of the most common requirements for many location-based applications. The location of a moving object changes continuously but the database location of the moving object cannot update continuously. Modeling of such moving object database should be considered to facilitate study of the performance and design parameters. Such study is essential for selecting the optimal solution in order to minimize the implementation of the overhead cost. Location updating strategy for such type of database is the most important criteria. This paper proposed a timed Petri net model based on one of the most common updating strategies, namely the distance updating strategy. In addition, a method for estimating the time needed to update Moving Object Database (MOD) using the concept of the minimum cycle time in timed Petri nets is presented. This time is the main criterion, which can be used to study the overhead communication cost for MOD. A typical numerical example is given to demonstrate the advantages of proposed modeling technique.

Keywords*: Updating moving object database, Deterministic timed Petri net, Deviation update policy and tracking moving object database.*

1. Introduction

Recent advances in wireless communication systems and Global Position system (GPS) are the main issues that make position tracking of moving objects feasible. Tracking is an enabling technology for many location-based services. As a result, a wide interest of many new applications that depend on location management can be shown in the literature [1], [5]. Tourist services, mobile E-commerce and digital battlefield are examples of these applications [2]. Other application classes that will benefit from tracking include transportation, traffic control, mobile resource management, and mobile workforce. This brought to database researchers the new challenge in the area of MOD.

Traditional database management system (DBMS) is not equipped to handle continuously changing data such as the transient position of moving object. This means that traditional DBMS deals with static data attributes at a given time [3], leading to a rather discrete model. Therefore, in many MOD applications a continuous model for these dynamic objects will be essential in order to mange such moving objects [1], [3], [4]. In this case, an updating strategy for a moving object is required. The objective of this strategy is to accurate track the current location of moving object while minimizing the number of updates. It is obvious that the more often data is updated, the more accurate the data will be. However, the cost of updating data increases with the frequency updating the data. That is, there is a trade-off between updating cost and information accuracy in designing MOD systems. The most common approach is distance update policy, which updates the database every x units of distance. So it provides a certain error in response to a query about the location of any object (e.g.: retrieve the current location of an object?) The answer is within a circle of radius x centered at location L (which is provided in the last updating for database). This approach is used in many applications due to its simplicity [6], [9].

Petri nets are graphical and mathematical modeling tools applicable to many systems. They are promising tools for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic and or stochastic as a graphical tool [8]. In this paper, a timed Petri net is presented

for updating a MOD. The moving object in this model uses a deviation update policy to update its database location. Then, the Petri net method of the minimum cycle time is applied to estimate the time duration required to update the MOD [8]. The number of moving objects, the number of wireless communication agents, and the number of processors of the DBMS affect this time duration.

The rest of this paper is organized as follows. Moving object database architecture is presented in Section 2. Section 3 describes Petri net basics that are used in this paper. The model for updating the MOD using Petri net is presented in Section 4. Section 5 is an illustration example of the model and how the minimum cycle time method can be used to estimate the updating time. Section 6 discusses some applications of the model. Finally, conclusions and a proposal for future work are drawn in Section 7.

2. Moving object database (MOD) architecture and modeling

As shown in Figure (1) the MOD system modeled in this paper consists of: 1) A number of moving objects each of which is equipped with a GPS receiver, a processor for calculating the deviation of moving object based on deviation updating policy and a local database. 2) A database server with a number of processors, which controls a database for all moving objects, and can be centralized or distributed and 3) wireless agents that provide communication services between moving objects and the DBMS. The history of a moving object's location and time is stored in the database at the database server.

Figure 1. Architecture of the MOD updating system

When the number of available communication agents is limited and there is, more number of moving objects needs to update their location in the central database system. Thus, an overhead results from increasing both the number of update message and the number of wireless communication agents. Therefore, this paper proposed Timed Petri net model to decrease both the number of update messages and overhead of communication cost of the moving object database in an efficient manner. However, the following section explains some basic aspects of Petri net before we introduced the model.

3. Petri net Basics

A Petri net is a graphical and mathematical modeling tool. It consists of three types of object. These objects are places, transitions, and arcs that connect them. In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers). Where a K-weighted arc can be interpreted as the set K parallel arcs. Labels for unity weight are usually omitted. Input arcs connect places with transitions, while output arcs start at a transition and end at a place. There are other types of arcs, e.g. inhibitor arcs. Places can contain tokens; the current state of the modeled system (the marking) is given by the number (and type if the tokens are distinguishable) of tokens in each place.

Transitions are active components. They model activities, which can occur (the transition fires), thus changing the state of the system (the marking of the Petri net). Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled (there are enough tokens available in the input places). When the transition fires, it removes tokens from its input places and adds some at all of its output places. The number of tokens removed/added depends on the cardinality of each arc [8], [9], [10]. In modeling using the concept of conditions and events, places represent conditions, and transition represent events. For instance, input (output) places may represent preconditions (post-conditions) to an event (transition). Some typical interpretation of transitions and their input places and output places are shown in Table (1). A formal definition of a Petri net is given in table (2) [8].

Table 1Table 1. Some typical interpretations and places.

Output places	Transition	Input places
Post-conditions	Event	Preconditions
Conclusions	Clause in logic	Conditions
Input signals	Signal processor	Input signals
Buffers	processor	Buffers

Table 2. The definition of a Petri Net.

Formally a Petri net (PN) can be defined as follows
PN = (P ,T, I , O, M0) Where *P = { p1, p2, p3, ... , pm}* is a finite set of places *T = {t1, t2, t3,.....,tn}* is a finite set of transitions where p υ *T≠ Φ, and P∩T = Φ* *I : (P x T) → N* is an input function that defines the directed arcs from places to transitions, and N is a set of nonnegative integer *O : (P x T) → N* is an output function that defines the directed arcs from transitions to places , and *M0: P→ N* is the initial marking.

The classical Petri nets do not include any notion of time; in order to use the Petri net formalism for the quantitative analysis of the performance and reliability of system versus time, a class of Timed Petri net has been introduced. The time delay variables associated with the Petri net can be either deterministic variables (leading to the class of models called deterministic Petri net), or random variables (leading to the class of models called Stochastic Petri net) [8]. When time delay is associated with transitions, this type of net is called timed transition, Petri net [10]. Suppose there is a time delay associated with transition this means that when this transition is enabled tokens remain on the input places of a transition for a time at least equal to the time delay associated with enabled transition before their removal by firing this transition.

4. Model of MOD Updating System Using Timed Petri net

This section presents an application of timed Petri net model. The model of MOD system is shown in Figure (2) and this model consists of three phases:

Phase (1): Moving objects and GPS receivers. Each moving object is equipped with one GPS receiver (for collecting the current real location of the object), one processor (for calculation), and local database (to store the previous location of moving object and a threshold which are used to calculate the deviation of the moving object). The functionality of this part is that moving objects get the information on location and time, and applies distance update policy to generate an updating

message, if the deviation exceeds a specific threshold or if the moving object stops moving, which will be sent to the database server.

Phase (2): Communication Services. This part includes several wireless agents, which provide communication services. The functionality of this part is to provide the communication between moving objects and the database server.

Phase (3): Database server. The information of all moving objects is stored in a database and handled by the DBMS, which is equipped with a number of processors. The main functionality of this part is to update the database with the received messages and generate return messages to update the pervious location of the moving object. The operation of each transition, the tokens in each place and the meaning of each arc inscription in this model shown in Figure (2) are given in Tables 3, 4 and 5 respectively.

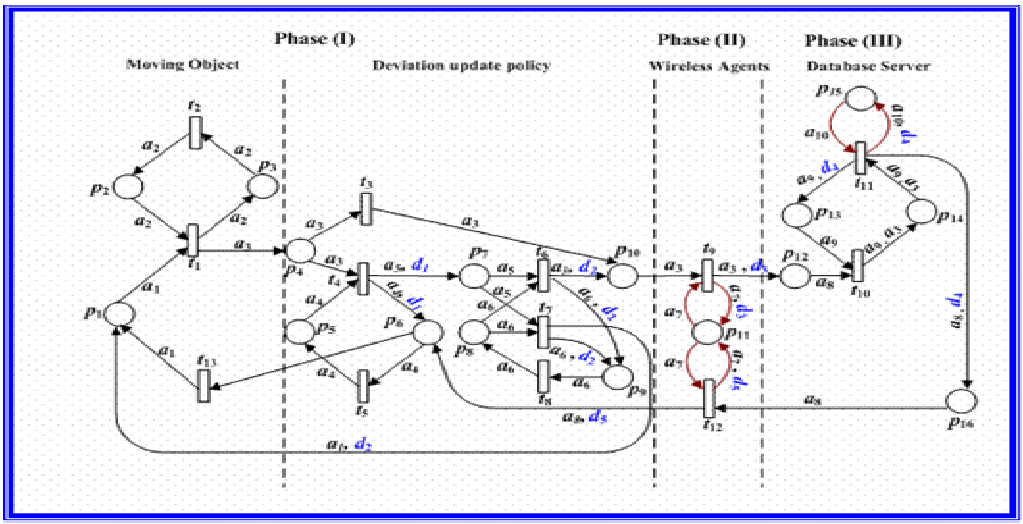


Figure 2. Petri net model for updating moving object database system.

Table 3. The operation of each transition.

Transition	Operation
t_1	A moving object gets its current location and time. Signal from satellites.
t_2	All GPS's receiving the signal from satellites.
t_3	The moving object (mid) stops moving, so sending < mid, cl, ct> to P10.
t_4	Calculating the deviation (did)for each moving object mid.
t_5	Passing all < mid, pl>'s from P6 to P5.
t_6	Comparing did with thid and the result is did > thid.
t_7	Comparing did with thid and the result is did ≤thid.
t_8	Passing all < mid, thid>'s from P9 to P8.
t_9	A wireless agent sends the message <mid, mcl, mct,> from the moving object mid to the database.
t_{10}	A moving object generates a transaction for updating the database.
t_{11}	A database processor executes the transactions for appending the received message <mid, mcl, mct> for the moving object mid.
t_{12}	A wireless agent Updating the location of the previous update for each moving object mid
t_{15}	Moving object gets its previous location and time after updating database

Table 4. The tokens of each place.

Place	Token
P1	<mid> All moving objects, waiting for current location and time. Initially all n moving objects are here.
P2	<gid, gcl, gct>, GPS receivers after collecting the information from satellites.
P3	All GPS's, n , waiting for receiving the signal from satellites
P4	<mid, mcl, mct, >, moving objects with the current location and time waiting for calculation or directly sending to the database server.
P5	<mid, plid>, waiting for sending pl to the moving objects. Initially all locations of the previous update for all n moving object are here.
P6	<mid, plid>, waiting for passing to P5.
P7	The set <mid, did, cl, ct> moving objects with the current location ,current time and deviation
P8	The thresholds for all moving objects, i.e. the set <mid, thid>, waiting for comparison. Initially all threshold values for all n moving object are here.
P9	The thresholds for all moving objects, i.e. the set <mid, thid>, waiting for passing < mid, thid > to P8.
P10	The set <mid , cl, ct>, waiting for sending to the database server.
P11	<wid> wireless agents. Initially there is r = the set of all wireless agents, waiting for receiving the information on moving objects.
P12	<mid, mcl, mct>, the messages of moving objects that are waiting for updating the database.
P13	<db, mid>, waiting for receiving the information on moving objects. Initially all n moving objects are here.
P14	<db, mid, mcl, mct>, transactions waiting for appending the information on the moving objects to the database db.
P15	<pid>, processors. Initially there is K = the set of all k processors managing the database of n moving objects
P16	<mid, cl>, waiting for updating the locations of the previous updates.

5. Illustration Example with Calculation of the Minimum Cycle Time

The minimum cycle time defined as the minimum time required to complete a firing sequence returning to the initial marking after firing each transition at least once [8]. This measure is used only for the timed net. The net shown in Figure (2) can be converted into a timed Petri net shown in Figure (3). A Petri net MATLAB toolbox, available at [12] is used to build the proposed model using MATLAB version 6.5. Since, we can move the delays d_1, d_2, d_3, d_4, d_5 of all the outgoing arcs of t_4, t_6, t_7, t_9, t_{11}, t_{12} to their corresponding transitions. We consider these delays are deterministic and the proposed model is deterministic timed Petri net model. Firings of transitions t_3 and t_4 represent the two cases: one (t_3) for which the moving object stop moving and the other (t_4) for which the moving object continue its motion and need to calculate the deviation. In addition, firings of transitions t_6 and t_7 represent the two cases: one (t_6) for which the deviation exceeds the threshold and the other (t_7) for which the deviation dose not exceed the threshold to simplify the analysis, it is assumed that these two cases occur with equal probabilities. In addition, the Self-loops (t_9-P_{11}, t_{12}-P_{11} *and* t_{11}-P_{15}) in Figure (2) are transformed into the loops as shown in Figure (3).

From studying the structural properties of the net in Figure (3) we can say that this net is bounded, conservative, repetitive and consistent. Now it is easy to find the following:

Incidence Matrix (A)

$$A = A_o - A_i \quad \textbf{(1)}$$

Where A_o = output matrix and A_i = input matrix. The entries of the incidence matrix are defined as follows: $aij = aij + - a_{ij}-$ where $a_{ij}+ = w\ (i, j)$ is the weight of the arc from transition i to its output place j and $a_{ij}- = w\ (i, j)$ is the weight of the arc to transition i from its input place j. When transition t_i fires, a_{ij} + represents the number of tokens deposited on it is output place p_j , a_{ij}- represents the of tokens removed from is input place p_j , a_{ij} represents the change in the number of tokens in place p_j. The following figures show the Petri net MATLAB toolbox to find the incidence matrix of the net mentioned above.

Table 5. The explanation of each arc inscription.

Arcs	Explanation
a1	<mid>, where mid is the identification number of a moving object.
a2	<gid, gcl, gct>, where gid is the identification number of a GPS receiver, gcl is current location of the moving object which has the same identification number as gid, and gct is the time for gcl.
a3	<mid, mcl, mct, >. A moving object mid has the current location and time (mcl, mct).
a4	<mid, pl> A set of locations of the previous update for all moving objects where mid = a moving object, *pl* = the location of the previous update for mid, *id = 1, 2, …, n*}
a5	<mid, did, cl, ct> (where mid = a moving object, did = the Euclidean distance between pl and cl, cl = the current location of mid, and ct = the time point when midis at cl *i = 1, 2, …, n*)
a5, d1	d1 is the time delay for calculation of the deviation of each moving object.
a4, d1	d1 is the time delay for calculation of the deviation of each moving object.
a6	<mid, thid>A set of thresholds for all moving objects, thid=the threshold for mid, *id = 1, 2, …, n}*
a3 ,d2	d2 is the time delay for Comparing did with thid for each moving object.
a7	<wid> The id of wireless communication agent wid
a6 , d2	d2 is the time delay for Comparing did with thid for each moving object.
a3 ,d3	d3 is the time delay associated with wireless communication for sending an update message
a7,d5	d5 is the time delay for wireless communication for sending Updating of the location of the previous update for each moving object mid
a8	<mid, cl> the moving objects current locations send to update the previous location where mid = a moving object ,cl = the current location of mid, *id = 1, 2, …, n*
a9	<db, mid>, where db is the name of the database handling the information of the moving object mid.
a9,a3	<db, mid, mcl, mct, >, a transaction of the database db for updating the current location and time (mcl, mct)
a8,d4	d4 is the time delay for the service provided by the processor pid in the database server that's the time to execute a transaction of the database db.
a10	<pid> The id of processor pid in the database server.
a9,d4	d4 is the time delay for the service provided by the processor pid in the database server, that's the time to execute a transaction of the database db.

Arcs	Explanation
a8,d5	d5 is the time delay for wireless communication for sending Updating of the location of the previous update for each moving object mid

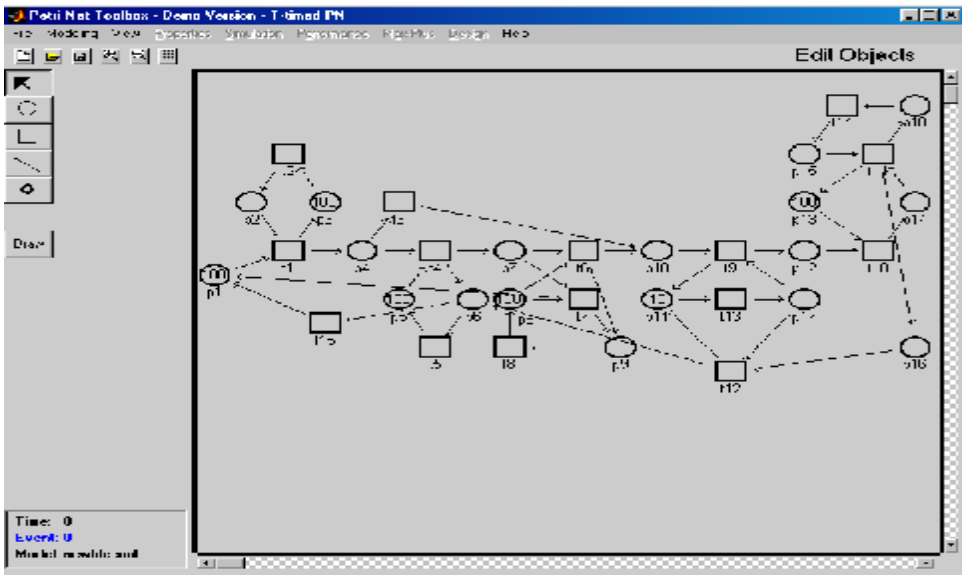


Figure 3. Deterministic timed Petri net model obtained from Figure 2.

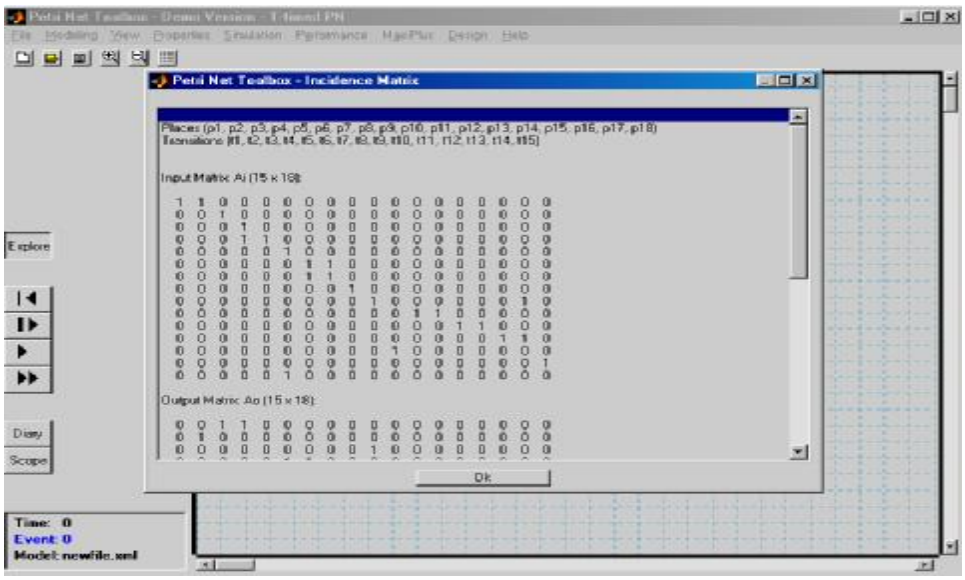


Figure 4-a. Shows the input Matrix of the net shown in Figure 3.

Two concepts are related to the incidence matrix. They are T-invariant and P-invariant, which are useful to find the minimum cycle time [8], [9],[10].

An integer solution x of A^T x = 0 is called T-invariant. The nonzero the corresponding transitions, which belong to a firing sequence transforming a marking M_0 back entries in a T-invariant represent the firing counts of to M_0. The T-invariant indicates the firing sequence transforming M_0 back to M_0 and the number of times these transitions appear in this sequence but does not specify the order of transition firing. In the net of Figure (3) there is a firing sequence from a marking

M back to the same marking M after firing each transition at least once. Such as $<<<t_1, t_1, t_4, t_5, t_7, t_8> t_2, t_1, t_3, t_{13}, t_9, t_{10}, t_{11}, t_{14}, t_{13}, t_{12}, t_{15}> t_2, t_1, t_4, t_5, t_6, t_8, t_{13}, t_9, t_{10}, t_{11}, t_{14}, t_{13}, t_{12}, t_{15}>$. The firing count vector x of this firing sequence is given by:

$$X = (3\ 3\ 1\ 2\ 2\ 1\ 1\ 2\ 2\ 2\ 2\ 2\ 4\ 2\ 2)^T \quad \textbf{(2)}$$

An integer solution y of Ay= 0 is called a P-invariant. This satisfies the following invariant property:

$$y^T M_i = y^T M_0 \quad \textbf{(3)}$$

Where M_0 is the initial marking and Mi is any marking reachable from M_0. The none zero entries in a p-invariant represent weights associated with the corresponding places so that the weight sum of tokens on these places is constant for all markings reachable from an initial marking. There are six (minimum, independent) P- invariants and they are given by:

$$\begin{aligned}
&P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7 P_8\ P_9\ P_{10} P_{11}\ P_{12}\ P_{13}\ P_{14} P_{15}\ P_{16}\ P_{17} P_{18}\\
y_1^T &= (0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0),\\
y_2^T &= (0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0),\\
y_3^T &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0),\\
y_4^T &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0),\\
y_5^T &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1) \text{ and}\\
y_6^T &= (1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0).
\end{aligned} \quad \textbf{(4)}$$

The minimum cycle time can be found by the following equation given in [8]:

$$Minimum\ Cycle\ Time = Max\{(y_k^T (A_i)^T DX / y_k^T M_0\} \quad \textbf{(5)}$$

where, for the net shown in Figure (3), x is the T-invariant given by Equation (2), yk's are six P-invariants given by Equation (4), and Ai = $[aij-]_{n \times m}$, D = the diagonal matrix of di (di is the time delay associated with transition ti , i = 1, 2,…, n) and M_0 = the initial marking are shown below:

$$\begin{aligned}
&P_1\ P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16} P_{17} P_{18}\\
M_0 &= (n\ 0\ n\ 0\ n\ 0\ 0\ n\ 0\ 0\ r\ 0\ n\ 0\ k\ 0\ 0\ 0)^T
\end{aligned}$$

Where n is the number of moving objects, GPS receivers, and tables in the local and server database, r is the number of wireless communication agents, and k is the number of database server processors. $yk^T M_0$ are found as follows:

$$\begin{aligned}
y_1^T M_0 &= n\\
y_2^T M_0 &= n\\
y_3^T M_0 &= r\\
y_4^T M_0 &= n\\
y_5^T M_0 &= k\\
y_6^T M_0 &= 2n
\end{aligned}$$

The input matrix A_i is given by the following matrix:

A_i =

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}
t_1	+1	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_2	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_3	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_4	0	0	0	+1	+1	0	0	0	0	0	0	0	0	0	0	0	0	0
t_5	0	0	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0
t_6	0	0	0	0	0	0	+1	+1	0	0	0	0	0	0	0	0	0	0
t_7	+1	0	0	0	0	0	+1	+1	0	0	0	0	0	0	0	0	0	0
t_8	0	0	0	0	0	0	0	0	+1	0	0	0	0	0	0	0	0	0
t_9	0	0	0	0	0	0	0	0	0	+1	0	0	0	0	0	0	+1	0
t_{10}	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0	0	0	0	0
t_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0	0	0
t_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	0
t_{13}	0	0	0	0	0	0	0	0	0	0	+1	0	0	0	0	0	0	0
t_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1
t_{15}	0	0	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0	0

The delay matrix D for the net in Figure (3) is the Diagonal matrix given by:

D =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
t_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_4	0	0	0	d_1	0	0	0	0	0	0	0	0	0	0	0
t_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_6	0	0	0	0	0	d_2	0	0	0	0	0	0	0	0	0
t_7	0	0	0	0	0	0	d_2	0	0	0	0	0	0	0	0
t_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_9	0	0	0	0	0	0	0	0	d_3	0	0	0	0	0	0
t_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{11}	0	0	0	0	0	0	0	0	0	0	d_4	0	0	0	0
t_{12}	0	0	0	0	0	0	0	0	0	0	0	d_5	0	0	0
t_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Thus

$$(A_i)^T\ DX\ =\ (\ 0\ \ 0\ \ 0\ \ 2d_1\ \ 2d_1\ 0\ \ 2d_2\ \ 2d_2\ \ 0\ \ 2d_3\ \ 0\ \ 0\ \ 0\ \ 2d_4\ \ 2d_4\ \ 2d_5\ \ 2d_3+2d_5\ \ 0\ )^T$$

and we find the following for Equation (5):

$$y_1^T(A_i)^T DX / y_1^T\ M_0 = 0$$
$$y_2^T(A_i)^T DX / y_2^T\ M_0 = 4d_2\ /n$$
$$y_3^T(A_i)^T DX / y_3^T\ M_0 = (2d_3+2d_5)\ /r$$
$$y_4^T(A_i)^T DX / y_4^T\ M_0 = 2d_4\ /n$$
$$y_5^T(A_i)^T DX / y_5^T\ M_0 = 2d_4/\ k$$
$$y_6^T(A_i)^T DX / y_6^T\ M_0 = (4d_1\ +2d_2+2(d_3+d_4+d_5))\ /2n$$

From Equation (5) the minimum cycle time of the net in Figure (3) can be given by:

$$The\ \min imum\ cycle\ time\ = Max\ \{0,\ 4d_2\ /n,\ 2(d_3+d_5)\ /r,\ 2d_4\ /n,\ 2d_4/\ k,\ (4d_1\ 2d_2+\ 2(d_3+d_4+d_5)\ /2n\} \qquad \textbf{(6)}$$

6. Application of the proposed Model

The minimum cycle time for the timed net in Figure (3) which is given by Equation (6) corresponds to the minimum time needed to check if the update is necessary for both stopped moving object or the moving object with deviation greater than a specific threshold and, if so, update once for each of n moving objects [11].

For example: consider the net in Figure (3) where we assume that the time delays as follows: d_1= 0.0002 time unit, d_2 = 0.0002 time unit, d_3= 0.01 time unit, d_4 = 1 time unit and d_5 = 0.01 time unit. Also assume that n= 100, r=10 and k=1. From equation (6) the minimum cycle time = $2d_4/k$ = 2 time unit.

If the GPS receiver collects the location information (current location of the moving object and current time) every three-time units, from Equation (6) each object can complete one update through 2 time units, so the system is safe. In other word the system is safe if the GPS receiver collects the location information every t time unit since t > minimum cycle time.

Also according to Equation (6), we can increase $4d_2/n$, $(2d_3+2d_5)/r$, $2d_4/n$ and $(4d_1 + 2d_2 + 2(d_3 + d_4 + d_5))/2n$ up to $2d_4/k$ without affecting the minimum cycle time. For example we can increase $(2d_3 + 2d_5)/r$ to $2d_4/k$ by decreasing the number of wireless agents' r =10 to 1 without affecting the performance of he system. The system with one agent allows each moving object to make its update to the database. This can reduce the cost paid for wireless communication services. Assuming that the wireless communication cost depends only on the time of communication, from Equation (6), we can also increase the number of moving objects without affecting the minimum cycle time. From this, we can deduce that a large number of moving objects can maintain their current locations in the database without needing to increase the number of wireless agents.

Suppose that the GPS receiver collects the location information more frequently, e.g., every 1 time unit, then the above minimum cycle time (2 time units) for each object may be too slow. That is, not all GPS signals can be recorded and the location information of some moving objects may be lost. In this case, it is necessary to decrease $2d_4/k$; by either decreasing d_4 or increasing k. Upgrading or improving the DBMS software so as to speed up transactions in the DBMS can reduce the delay d_4. Adding more DBMS processors can increase the value of k and reduce the minimum cycle time. For the DBMS with more than one processor, the database can be processed in parallel, reducing the time needed for update.

7. Conclusions and Future Work

This paper presents a timed Petri net model for updating moving object database system using distance-updating strategy. In addition, a method for calculation of the minimum cycle time for updating the database is proposed. A Petri net MATLAB toolbox used to study the structural properties of the proposed model. The presented model can be more complex by refining the presented model. For example, transitions t_9 and t_{12} can be refined in order to model wireless communication protocols. Transitions t_{10} and t_{11} also can be expanded to simulate a specific DBMS architecture. Another updating strategy such as a deviation updating strategy can be used instead of distance updating policy.

8. References

[1] O. Wolfson, L. Jiang, P. Sistla, S. Chamberlain, N. Rishe, and M. Deng, "Databases for Tracking Mobile Units in Real Time", Springer-Verlag Lecture Notes in Computer Science 1540, Proceedings of the Seventh International Conference on Database Theory (ICDT), Jerusalem, Israel, , Jan 1999, pp. 169-186.

[2] O. Wolfson, L. Jiang, S. Chamberlain, and S. Dao, "Location Management in Moving Object Databases.", Proceedings of The Second International Workshop on Satellite-Based Information Services (WOSBIS'97), Budapest, Hungary, October 1997.

[3] O. Wolfson , L. .Jiang, S. Chamberlain, and B. Xu, " Moving Object Databases: Issues and Solutions", Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98), Capri, Italy, July 1998, pp. 111-122.

[4] O. Wolfson, P. Sistla,, B. Xu, J. Zhou, S. Chamberlain, Y. Yesha, and N. Rishe, " Tracking Moving objects using database technology in DOMINO", proceedings of NGITS 99,the fourth workshop on next generation information technologies and systems, , July 1999, pp. 112-199.

[5] O. Wolfson , S. Chamberlain, , P. Sistla, B. Xu, and J. Zhou, "DOMINO: Databases fOr MovINg Objects tracking,", the Proceedings of the ACM-SIGMOD 1999, International Conference on Management of Data, Philadelphia, PA,

June 1999.
[6] QUALCOMMInc. Retrieved September 4, 2005, from: // http://www.qualcomm.com
[7] At Road Inc. Retrieved August 20, 2005, From: // http://www.atroad.com/
[8] T. Murata, "Petri Nets: Properties, Analysis and Application ", Proceedings of the IEEE, Vol. 77, No. 4, April 1989, pp. 541-580.
[9] O. Wolfson, and H. Yin, "Accuracy and resource consumption in tracking and location prediction", proceedings of 8Th international symposiums on spatial and temporal databases, July 2003. pp. 325-343.
[10] R. Zurawski, and M. Zhou, " Petri Nets and Industrial Application: A Tutorial", IEEE transaction on industrial electronics, VOL.41, NO.6, December 1994.
[11] T. Murata, J. Yim, H. Yin, and O. Wolfson " Fuzzy-Timing Petri-Net Model for Updating Moving Objects Database" Proceedings of the 2003 VIP Scientific Forum of International Conference on IPSI (Internet, Processing, Systems, and Interdisciplinaries), Sveti Stefan, Montenegro, Yugoslavia, October 2003, pp.1-7.
[12] C. Mahullea, M. Hanako, and O. Pasttravanu, Petri Net Toolbox for MATLAB. Retrieved August 20, 2005,
[13] From: // http://www.ac.tuiasi.ro/pntool .

An Intelligent Decision Support System for Faculty Evaluation

Hegazy Zaher
Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt.

W. F. Abd El-Wahed
Faculty of computers & Information, Shiben El-Kom, Menoufia University, Egypt.

Mahmoud M. El-Sherbiny,
Operations Research Dept, Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt.
m_sherbiny@yahoo.com

Haidy M.E. Matloub
Enrolled for MSc. Degree in Operations Research, (ISSR), Cairo University, Egypt

Abstract

This paper presents an intelligent decision support system (IDSSFPE) for faculty performance evaluation. The requirements of the faculty decision makers are identified through the analysis and the design of the IDSSFPE. Three models based of the IDSSFPE are illustrated. Such models are: i) The performance measurement model based on data envelopment analysis (DEA), ii) Number of applicants estimation model, and iii) Applicants classification model based on artificial neural network (ANN).

Keywords*: Faculty performance, Intelligence decision support system, Artificial neural network*

1. Introduction

Over last decade, new technologies like artificial neural networks (ANN) have seen a rapid acceptance for solving a wide range of business problems. Many papers have been reviewed ANN from various points of view. Simth and Gupta [28] review ANN applications and techniques for the operations researcher. Bel Raggad used ANN as a technology for knowledge resources management [3] and Bijayananda using it in the field of education to predict MBA students' success [4].

Also, Data envelopment analysis (DEA) is a methodology that has been used to evaluate the efficiency of entities. DEA used in various fields based on [14] such as, Jill Johnes measure teaching efficiency in higher education to graduates from UK universities [20]. J.Colin et al using computed DEA based efficiency scores for policy evaluations and possible funding guidance in UK higher education [18]. Jill Johnes explores the advantages and drawbacks of various methods for measuring efficiency in higher education context [19]. B.Casu and E.Thannssaulis evaluate cost efficiency in UK university control administration [2]. Colbert used DEA to determine the relative efficiency of 24 top ranked US MBA programs [7].

White in [31] points out that the trend of decision support system (DSS) will come into increased use in universities, as their administrators management scientists combine their managerial skills with quantitative techniques for organizing and presenting information. Elsa et al in [12] discusses a balanced scorecard approach for strategy and quality driven universities. Roger and Saud [27] develop a DSS to enable systematic exploration of educational database.

In literature the usage of DEA in a DSS is not a large scale. Such as, W.K. Wang describes knowledge–base DSS for measuring the performance of government real state investment using DEA models [33]. Marta et al used it as a tool for integrating heterogeneous data model applies fuzzy logic to decision support systems [21].

Some applications combined DEA and ANN models, such as, Parag and James in [25] used this combination to improve forecasting accuracy of neural networks under monotonicity assumption. Desheng, Zijiang and liang in [10] integrate DEA and ANN to examine the relative branch efficiency of a big Canadian bank. Parag in [26] used DEA to illustrate that non-linear variable return to scale (VRS) with empirical data and compare the performance of non-linear ANN with linear regression model.

This paper, combine the DSS, ANN and DEA for building an Intelligent Decision Support System for Faculty Performance Evaluation (IDSSFPE) and applying such system on El-Menoufia University. Such IDSSFPE integrates number of applicants admitted, evaluation performance for a faculty, and classify applicants to each department in a faculty. The number of applicants admitted to a faculty and to each department is estimated using ANN while Evaluation performance for a faculty is analyzed by applying DEA. The remaining parts of this paper are arranged as follows: Section 2 contains problem description. Section 3 the architecture of IDSSFPE is discussed. In section 4 discuses a case study applied IDSSFPE. Section 5 contains the analysis of results. Finally the conclusion is summarized in section 6.

2. Problem Description

Based on the way to ensure quality for higher education in Egypt, the IDSSFPE was build to help top management to take decisions concerning to the needs for measuring the performance of faculties, determine number of applicants admitted to any faculty, and applicants admitted to each department in a faculty.

Menoufia University is one of the higher education institutions in Egypt. Therefore, it is considered to the natural outlet for secondary school graduates seeking for university education form its location and other locations in Egypt. They wants to join the university based on his/her grades or as a best location university. Therefore, the IDSSFPE was used in order to evaluate the performance of faculties in Menoufia University.

3. The IDSSFPE Architecture

The Architecture of IDSSFPE developed for the use of faculty decision-making as outlined in Fig 1 which consists of the following parts:

- User Interface,
- Database, and
- Model -Base Module.

The main components and functions of each part will be illustrated in the following sections.

3.1. **The User Interface** The user interface of IDSSFPE for a faculty performance evaluation is a menu driven and user-friendly interface. It allows the DM to interact with other modules of the IDSSFPE. It has a capability to enter/open database implemented by FPE program. It permits a DEA and EMS software to be run as independent software. The UI includes a set of links to the IDSSFPE modules and accesses the DB and facilitates accessing to run the DEA modules, creating the classification and forecasting databases in addition to trains ANN to specify number of applicants admitted to a faculty and specify which applicants enter each department in a faculty.

3.2. The Database The database module divided into two parts. External database that can be generated out of program FPE to use in Frontier Analyst Software to measure performance, while internal database makes by program FPE and used for forecasting or classification in a faculty.

3.3. Model-Base Modules The model-base module incorporates multiple functionalities or tools into a loosely couple single package in such a way that they can be used independently. The major issues during the development of the IDSSFPE were the selection of model types and computing technology. The models developed to be included in the IDSSFPE are:

i. Forecasting Model
ii. Classification Model
iii. Performance assessment Model

The main components and the role of each model in the IDSSFPE will be illustrated in the following section.

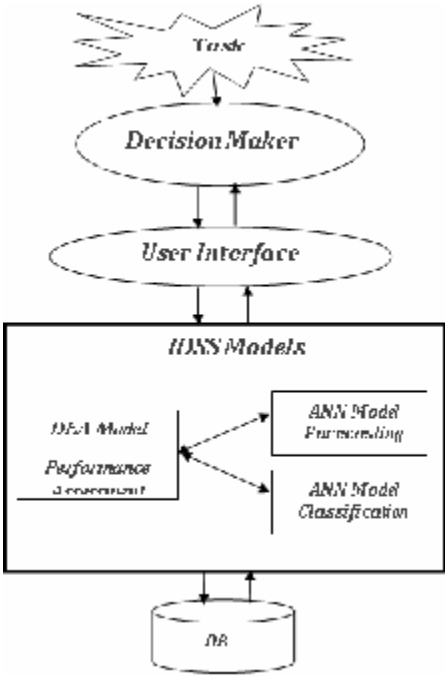


Figure 1. The Main Components Of The IDSSFPE

i. Forecasting Model

Forecasting models naturally highlight variable that have significant variation over the forecasting periods and can be forecasted well. In higher education two methodologies are used in enrollment forecasting. The first is econometric model and the second is trend model. Econometric forecasting Models used for identifying the determinants of enrollment rates rather than forecasting the enrollment Trend Models while the methodology of trend model is straightforward [9]. These models can be based on predicting the path of the college enrollment rate or total enrollment. The models can be statistical trend fitting models or can be based on "expert" judgment as to future rate of change the forecast performance of the various trend models should be analyzed [1].

In recent years ANN is identified as model for forecasting. Many studies have shown that neural networks can be one of the very useful tools in forecasting. Neural networks have flexible nonlinear function mapping capability that can approximate any continuous function with arbitrarily desired accuracy [8,17]. ANN are also nonparametric data-driven models that impose little prior assumptions on the underlying processes from which data are generated. As such, ANN are less susceptible to the model misspecification problem than most parametric nonlinear methods [15]. Neural networks are trainable analytic tool that attempt to mimic information processing patterns in the brain [16]. Because of ANN do not necessarily require assumptions about population distribution, economists, mathematicians and statisticians are increasingly using it for data analysis [15]. ANN not only doesn't require assumption about underlying population but also it is a powerful forecasting tool that draws on the most recent development in artificial intelligence research. With the advent of modern computer technology and information science sophisticated information systems can built that make decisions or predictions based on information contained in available past data. Such systems are called learning systems and are currently used for the purpose of classification or predictions [32]. Tsoukalas and Uhrig [29] define a neural network as a data processing system consisting of a large number of simple highly interconnected processing elements (artificial neurons) in architecture by the structure of cerebral cortex of the brain. Approximately 95% of the reported neural network business applications used the multiplayer feed forward neural networks (MFNN) with the back propagation-learning rule [34]. This type of neural network is popular [28] because its abroad applicability to many problem domains of relevance to business: principally predictions, classification, and modeling. MFNN is appropriate for solving problems that involve learning the relationships between a set of inputs and known outputs. Based on the above-mentioned models' review used in forecasting,

the MFNN is selected in this paper to sole number of applicants' estimation. MFNN is a collection of interconnected homogenous processing unit, called neuron. A neuron activity is controlled by a continuous and differentiable mathematical function that aggregates input signals received from other neurons and produce output signals transmitted to other neurons. The flow of information is from left to right, with the input layer receives input data describing the decision domain passed through the network via the hidden layer of neurons. The number of nodes constituting the hidden layer depends on the complexity of pattern in input data. Then to the output layer, which is prepares the response representing the situation outcome. Fig 3 shows the general architecture of a MFNN used in the IDSSFPE. The output for algorithm of ANN known because the decision maker don't determine number of applicants admitted to faculty based on faculty needs only but this number specified from the high council of universities. Therefore, a neural network with back propagation learning rule approach is used for this situation. Based in input variable in decision domain a recommended number of applicants admitted to faculty would be obtained.

Many steps are followed in building a forecasting model using neural network as state in [30]. The development process for ANN application includes nine steps as shown in Fig.3. In the next sections some of these steps of flow diagram examined in more detail:

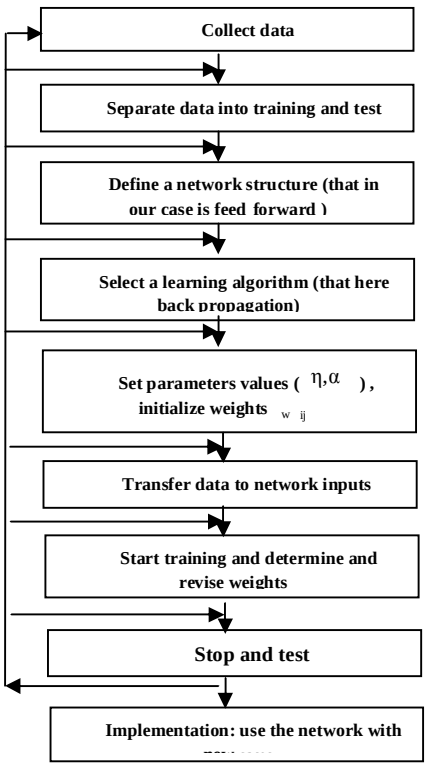
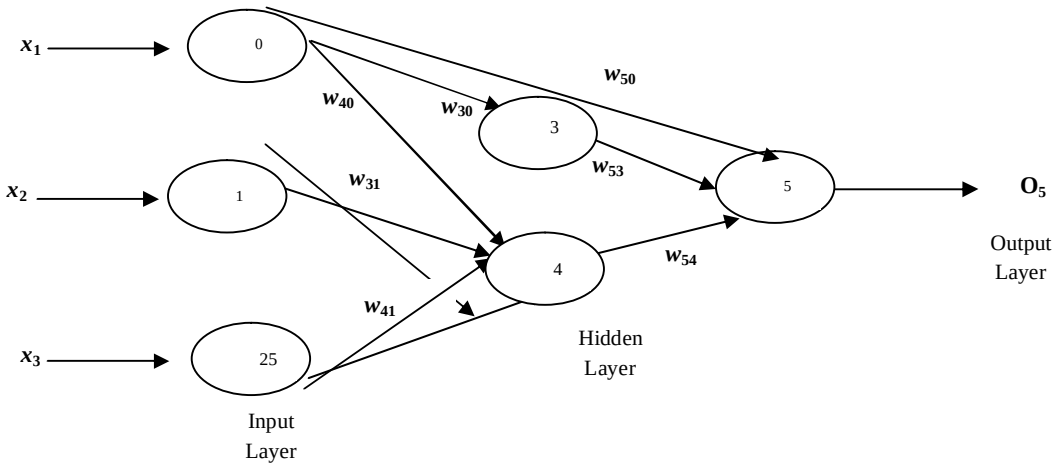


Figure 2. Flow diagram of the development process of an ANN

Collect Data

The data used in the IDSSFPE are collected from Menoufia University of the academic year 2002/2003. The data considered to be the inputs of the forecasting model are: the number of professors which is considered as an approximation of resources available for faculty teaching, the faculty supported from scientific search unity which is considered as sound indicator for the performance of a faculty and its research, and the cost of social solidarity which is considered as an approximation for expenses on applicant as aid.

While the output of that model are the number of enrollment applicants in a faculty which considered to be as an indicator for quality of student on admission.



Network Structure

One of the most critical decisions is the network architecture. That means the number of layers, the number of nodes in each layer, and how nodes are connected. In our situation a multiplayer feed forward neural network (MFFN) is chosen and then input nodes, output nodes, number of hidden layer, number of hidden nodes are determined. In network used by program chosen to be with three layer, input layer contains nodes based on the attributes of data set that user enter, number of hidden nodes increase by one of input node, and finally the output layer contains one node that represents the forecasted number of applicants to faculty.

Transfer data to network

The decisions made in this phase of development are critical to the performance of a network. The goal of data preparation is to reduce nonlinearity when its character is known and let the network resolve the hidden nonlinearities that not are understood.

Scaling data

While loading data from the database file, minimum value and maximum value for each column are determined. The values of inputs between minimum and maximum (called range Δ) must be scaled into the range 0.1 to 0.9 for the neural network input.

The equation for scale input is [29]:

$$y = (0.8/\Delta)x + [0.9 - (0.8x_{\max}/\Delta)]$$

Where**:**

y is the value of data scaled,

Δ is the range between maximum and minimum value ($\Delta = x_{\max} - x_{\min}$), and

x is the value of input will be transfer or scale.

Scaling variable between 0.1 and 0.9 is often used to limit the amount of sigmoid activation function used in the representation of variables in order to avoid "network paralysis". When the network is run, the output produced must be scaled.

Network training

Training the ANN is an iterative process that starts from random set of weights and gradually enhances the fitness of the network model and the known data set. The iteration continues until the error sum is converged to low acceptable level. In the back propagation algorithm, two parameter learning rate and moment adjusted to control the speed of reaching the minimum ratio of the different between calculated value and value of training cases. Taking these steps, the following parameters have been used for the MFNN design:

Threshold	0.1or less
Learning rate (h)	0.6
Moment (a)	0.9
Error, (E)	0.01

Uses $h^* = \frac{h}{1-a} = \frac{0.6}{1-0.9} = 0.25$ for as better learning rate and iterations 100000.

ii. Classification Model

The data collected from faculty of computers & information to be considered as the inputs of classification model are the grades of students in the following courses: Logic design (LD), Data Structure (DS), Operations Research (OR), Computer Software (CS). The output of the model is the applicants' classification to department that represents the admission policy to each department. The MFNN with back propagation learning rule is used to build this model for classifying applicants to each department in a faculty. The same nine steps used in forecasting model are followed in building the classification model.

iii. Performance assessment model:

Model for performance assessment was selected on the basis of the required output, which means measure the efficiency of a faculty based on its inputs and outputs.

Data Envelopment Analysis (DEA) initiated by Charnes, Cooper and Rhodes [5] is a linear programming–based technique for measuring the performance of administrative units. The performance of a unit is evaluated by comparing its performance with the best performing units of the sample. The measure of performance is expressed in the form of efficiency score. DEA is a non-parametric linear programming technique for measuring the relative efficiency of decision-making units (DMUs) that perform the same type of functions and have the identical goals and objectives. The notion of efficiency employed in the DEA approach is termed Pareto Efficiency, which is an extension of the social choice criterion of Pareto Optimality [23]. An advantage of DEA is that there is no preconceived functional form imposed on the data in determining the efficient units. That is, DEA estimates the production function of efficient DMUs using piecewise linear programming on the sample data instead of making restrictive assumptions about the underlying production technology. As an efficient frontier technique, DEA identifies the inefficiency in a particular DMU by comparing it to similar DMUs regarded as efficient. The efficiency of each DMU is measured relative to all other DMUs under the restriction that all DMUs lay on or below the efficient frontier. The DMUs indicated as efficient are only efficient in relation to others in the sample. The principal disadvantage of DEA is that it assumes data to be free of measurement error. While the need for reliable data is the same for all statistical analysis, DEA is particularly sensitive to unreliable data because the units deemed efficient determine the efficient frontier and, thus, the efficiency scores of those units under this frontier [24]. DEA appears as the most appropriate methodology for higher education evaluations.

DEA maximize the ratio of a weighted sum of outputs to a weighted sum of inputs where the attached weights to inputs and outputs are treated as variables that are to be optimized. The estimation procedure of the relative efficiency score for each DMU can be described as follows. Firstly, the best practice production function is generated by a set of DMU that

receive the maximum output amounts for a given level of inputs compared with the other DMUs, called relatively efficient. Secondly, for each department DEA produces an efficiency score by comparing the other non-efficient DMUs with the production frontier. These DMUs not lying on the frontier are called inefficient. Whereas the efficiency score for the frontier generating DMUs equals one, the distance to the efficiency frontier determines the level of inefficiency. Assume there is n DMUs that are denoted by index j. Each DMU uses m inputs to produce p outputs. DMU j uses the amount x_{ij} of input i to produce the amount y_{kj} of output k. The assigned weight to output k is u_k and the weight assigned to input i is v_i. Productivity is defined as the ratio of the weighted sum of outputs to the weighted sum of inputs. Thus, the relative efficiency of a DMU can be written as (2).

$$E_o = \frac{\sum_{k=1}^{p} u_k y_{ko}}{\sum_{i=1}^{m} v_i x_{io}} \qquad (2)$$

By determining the weights of the inputs and outputs endogenously, DEA permits a DMU to adopt a set of weights that will maximize its productivity ratio without exceeding 1 for other DMUs. Introducing this constraint converts the productivity ratio into a measure of relative efficiency as in (3).

The constraints mean that the ratio of virtual output to virtual input of the other DMUs does not exceed 1 for every DMU. Assuming that the DMUo is evaluated, the objective is to obtain weights u and v in such a way that the ratio of DMUo is maximized. Furthermore the non-negative constraint must hold for the weights.

$$\left.\begin{aligned} &\underset{u_k, v_i}{\text{maximize}}\ \mathrm{E_o} = \frac{\sum_{k=1}^{p} u_k y_{ko}}{\sum_{i=1}^{m} v_i x_{io}} \\ &\textit{subject to:} \\ &\frac{\sum_{k=1}^{p} u_k y_{kj}}{\sum_{i=1}^{m} v_i x_{ij}} \le 1 \quad ; j = 1, \ldots\ldots, n, \\ &\frac{u_k}{\sum_{i=1}^{m} v_i x_{ij}} > e \; ; \; k = 1, \ldots\ldots, p \\ &\frac{v_i}{\sum_{i=1}^{m} v_i x_{ij}} > e \quad ; i = 1, \ldots\ldots, m. \\ &e \ge 0 \end{aligned}\right\} \qquad (3)$$

The mathematical formulation of DEA is represented in (4) [22].

$$
\left.
\begin{aligned}
&\max\ E_o = \frac{\sum_{j=1}^{J} u_j y_{jm}}{\sum_{i=1}^{I} v_i x_{im}} \\
&s.t. \\
&\frac{\sum_{j=1}^{J} u_j y_{jn}}{\sum_{i=1}^{I} v_i x_{in}} \le 1 \quad \forall n = 1,.....o,...,N \\
&u_j \ge 0 \quad , v_i \ge 0 \\
&where \begin{bmatrix} j = 1,.....,J \\ i = 1,.....,I \end{bmatrix}
\end{aligned}
\right\} \qquad (4)
$$

where:

E_o is the efficiency of unit o.

y_{jn} is the observed quantity of output j produced by unit $n = 1,2,.....,N$

x_{in} is the observed quantity of input i produced by unit $n = 1,2,.....,N$

u_j is the weight (to be determined) given to output j by base unit o

v_i is the weight (to be determined) given to input i by base unit o

Remark: A fully rigorous development would replace $u_j, v_i \ge 0$ with $\frac{u_k}{\sum_{i=1}^{m} v_i x_{ij}}, \frac{v_i}{\sum_{i=1}^{m} v_i x_{ij}} > e \ge 0$ where e is a non-Archimedean elemant smaller than any positive real number.

The linear program solution technique will attempt to make efficiency of the unit as large as possible. The search procedure will terminate when some of the efficiencies hit 1.

The data considered to be the inputs of performance assessment model are collected from the university performance report. These data are :-

Inputs:

- Enrollment student: Number of applicants enrolled in a faculty for 1st year
- Students/teacher ratio: is the ratio between enrolled applicants and their teachers, is a sound indicator of level resources and effective level supervision in teaching programs.
- Master Registrations: Number of postgraduate applicant enrolled in Master degree in year *n*.
- PhD Registration: Number of postgraduate applicants enrolled in PhD degree in year *n*.

Outputs:

- Graduation number: Number of undergraduate applicant graduate in year *n*.
- Master graduates: Number of postgraduate applicant obtains the Master degree in year *n*.
- PhD graduates: Number of postgraduate applicant obtain the PhD degree in year *n*
- Number of conferences: Number of conferences proceeded by faculty in national & international domain.

4. Case Study

In this paper, the IDSSFPE with DEA and ANN technique applied to a case study in Menoufia University to assessment performance of 11 faculties in university, forecast enrollment applicant to them, and classify applicants to departments in faculty of computers & information.

Forecasting Model

In the case study the neural network using this model are Consist of three layers; which are input layer, one hidden layer and output layer (total three layers) was used.

- The input layer has three inputs which are Faculty Supported from scientific search unity (FSSSU), Social Solidarity (SS), and number of Professors (#of Prof) in the faculty.
- The output layer has only one neuron and one output, namely forecasting (number of applicant admitted).

The neural network model was built using FPE software developed by c#. The model was trained with 11 data points represented.

Classification Model

Faculty of computers & information used as a sample of faculties to classify student to each department. Neural network for this model with three layers was used, input layer has four inputs, which are grades of students in the following courses: logic design (LD), Data Structure (DS), Operations Research (OR), and Computer Science (CS). The output layer has only one neuron and one output, namely classification.

Performance assessment Model

In the IDSSFPE, DEA is conducted for measure performance of a faculty in Menoufia University. For faculty performance, the analysis is performing using the inputs and outputs mentioned in 3.3 (iii).

Selecting analysis options in DEA

The analysis options available in DEA are: Input minimization (also known as input orientation or contraction) instructs DEA to reduce the inputs as much as possible without dropping the output levels. Alternatively, when management's focus is on raising productivity without increasing the resource base, output maximization (also known as output orientation or expansion) could be specified. under output maximization, outputs are raised without increasing the inputs. It is worth noting that when none of the inputs is controllable by management (not the case in this study), one can only specify the output maximization model. The current study contributes to the DEA modeling investigations of MU faculties by computing an efficiency measures by making allowance for input measurement minimization and output maximization for radial adjustments. This model computed also assumes a convex technology. Another analysis option in DEA is a choice between constant returns to scale (CRS) and variable returns to scale (VRS). when computing the technical efficiency (TE) measures. Clearly the CRS assumption is only appropriate when all faculties are operating at an optimal scale. If this is not the case, the TE measures will be confounded by scale efficiencies. Consequently, in the latter case, the VRS assumption should be employed so as to compute TE measures that are devoid of these scale efficiency effects [6]. The VRS model will always envelop the data more closely than the CRS model, irrespective of whether variable returns to scale exist [11]. The appropriate returns to scale assumption is employed with respect to the data utilized in the current study, is a CRS assumption.

Sample to Faculty of Engineering (Sheben):

The mathematical model:

$$Max\, \mathrm{E}_{FoA} = \frac{571u_1 + 20u_2 + 4u_3 + 32u_4 + 9u_5}{4732v_1 + 23v_2 + 70v_3 + 10v_4}$$

s.t.

$\mathrm{E}_{\mathrm{FoA}} \le 1$

$\mathrm{E}_{\mathrm{FoEn}} \le 1$

..

.

.

$\mathrm{E}_{\mathrm{C\&I}} \le 1$ $\qquad u_j \ge 0$ j = 1,...,5

$v_i \ge 0$ i = 1,....,4

Transformation to LP for faculty of Engineering based on equation (4):

$$\max_{u,v} \; z = 517\,u_1 + 20\,u_2 + 4\,u_3 + 32\,u_4 + 9\,u_5$$

subject to :

$$517\,u_1 + 20\,u_2 + 4\,u_3 + 32\,u_4 + 9\,u_5 - 4732\,v_1 - 23\,v_2 - 70\,v_3 - 10v_4 \le 0$$
$$211\,u_1 + 23u_2 + 24u_3 + 13u_4 + 6u_5 - 873\,v_1 - 6\,v_2 - 27\,v_3 - 24v_4 \le 0$$
$$1953\,u_1 + 12\,u_2 + 6\,u_3 + 15\,u_4 - 9394\,v_1 - 208\,v_2 - 22\,v_3 - 10v_4 \le 0$$
$$421\,u_1 + 29\,u_2 + 8\,u_3 + 30\,u_4 + 3\,u_5 - 2361\,v_1 - 15\,v_2 - 65\,v_3 - 9v_4 \le 0$$
$$780\,u_1 + u_2 + 4\,u_3 + 9\,u_4 + 6\,u_5 - 10335\,v_1 - 272\,v_2 - 70\,v_3 - 10v_4 \le 0$$
$$314\,u_1 + 72\,u_2 + 45\,u_3 + 13\,u_4 + 22\,u_5 - 2204\,v_1 - 7\,v_2 - 120\,v_3 - 65v_4 \le 0$$
$$1005\,u_1 + 4\,u_3 + 8u_4 + 4\,u_5 - 15140\,v_1 - 330\,v_2 - 8v_4 \le 0$$
$$1404\,u_1 + 36\,u_2 + 4\,u_3 + 8\,u_4 + 2\,u_5 - 9114\,v_1 - 82\,v_2 - 228\,v_3 - 4v_4 \le 0$$
$$518\,u_1 + 33\,u_2 + 10\,u_3 + 11\,u_4 + 2\,u_5 - 1605\,v_1 - 39\,v_2 - 61\,v_3 - 27v_4 \le 0$$
$$517\,u_1 + 20\,u_2 + 4\,u_3 + 5\,u_4 - 1107\,v_1 - 158\,v_2 - 70\,v_3 - 10v_4 \le 0$$
$$644\,u_1 + 2\,u_4 - 315\,v_1 - 23\,v_2 \le 0$$
$$4732\,v_1 + 23\,v_2 + 70\,v_3 + 10\,v_4 = 1$$
$$u_j \ge 0 \; \mathrm{j} = 1,...,4$$
$$v_i \ge 0 \; \mathrm{i} = 1,....,4$$

There are mathematical formulations for each faculty of 11 faculties in Menoufia University, which are (agriculture, engineering, education, science, commerce, medicine, law, arts, home affairs, tourism & hotels, computers & information)

5. Analysis of Results

5.1. **Result of Forecasting Model:** As mentioned before that the MFNN model used to solve the forecasting model, which used for predict the number of applicant admitted to a faculty. The result is shown in Fig 4. The column called output is a recommendation for the decision maker, assist him to make decisions based on this recommendations.

Figure 4 Result of Forecasting Model

This recommendation may be different if the decision maker change of the setting of network such as number of iteration or learning rule or the both. The difference in recommendation based also on the change of decision domain meaning, the input variable to neural network.

In our case of faculties of Menoufia University the result with error 0.02 at learning rate 0.3 and 400 iterations. Then the DM uses these recommendations as an input for performance assessment model to notice the effect of number of applicants admitted to his faculty performance.

5.2. **Result of Classification Model:** The classification model used to classify applicant to each department in a faculty. The results show in Fig 5. The output columan is a recommendation for preferred department for this applicant. the decision maker make decisions based this recommendation. This recommendation may be different if the decision maker changes the setting of network such as number of iterations or learning rate or both.

The difference in recommendation based also on the change of decision domain meaning, the change in input variables to neural network.

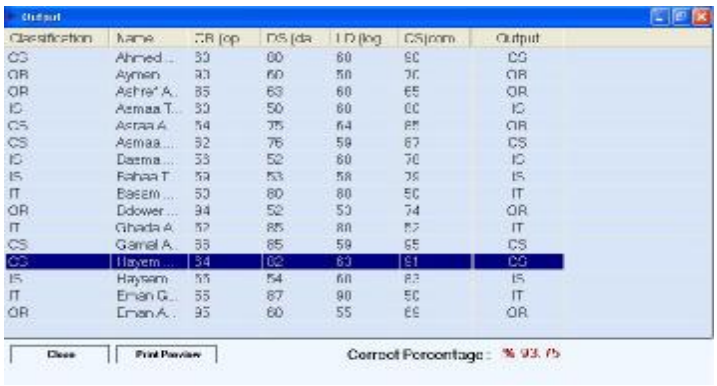


Figure 5 Result of Classification Model in Faculty of Computers & Information

In our case faculty of computers & information the classification ability of MFNN model was 93.75 at learning rate 0.25 and 500 iterations.

5.3. **Result of Performance assessment Model:** The performance assessment model that use a DEA based input and output variables specify up performance indicators of quality assurance in higher education [13].

Program IDSSFPE uses software Frontier Analyst to solve performance assessment model. Each software can be used independently, the software results are the following.

Table 1 Exhibit the performance score of each faculty based inputs and outputs mention in section 3.3 (iii). The two options classified the Faculty of Agriculture, Faculty of Engineering (Shbeen), Faculty of Science, Faculty of Medicine, Faculty of Arts, Faculty of Tourism & Hotels, and Faculty of Computers & Information as an efficient faculties where each faculty has 100% performance , while Faculty of Education, Faculty of Commerce, Faculty of Law, and Faculty of Home Affairs are classified as inefficient faculties where each faculty has less than 100% performance.

In order to improve the performance of inefficient faculties, some inputs may decrease and some output may increase. Table 2 exhibits the Potential Improvement of two options used concerning each inefficient faculty. The variable with sign (-) is an input variable that may be decrease with this percent to the target level, while variables with sign (+) is an output variables that may be increase with this percent from the target level.

The efficiency results suggest that the Menoufia faculties are operating at a fairly high level of efficiency relative to each other, although there is a chance for improvement in several faculties. The way to improve each faculty are illustrative in Table 2.based on these result of performance assesment model the decision maker can change of decision domain of applicants' classification that effect on graduation number and also in faculty performance.

Table 1. Faculty Performance

Faculties	CCR Input-Min	CCR Output-Max
Faculty of Agriculture	100%	100%
Faculty of Engineering(shpeen)	100%	100%
Faculty of Education	88.39%	88.39%
Faculty of Science	100%	100%
Faculty of Commerce	43.42%	43.42%
Faculty of Medicine	100%	100%
Faculty of Law	64.19%	64.19%
Faculty of Arts	100%	100%
Faculty of Home Affairs	95.48%	95.48%
Faculty of Tourism & Hotels	100%	100%
Faculty of Computers & Information	100%	100%

6. Conclusion

This paper has outlined the features of an IDSSFPE technique to evaluate performance of a faculty based on DEA and ANN techniques. The DEA is used as an efficiency tool to measure performance of a faculty while the ANN is used as an evaluation tool to forecast number of applicants admitted to the faculty and also, to classify applicants into departments of the faculty. Such IDSSFPE can be considered as a valuable tool to support and improve the decision making in a faculty.

Table 2. The Potential Improvement of each Faculty

Variable	Faculty of Commerce		Faculty of Law		Faculty of Education		Home Affairs	
	CCR Min-Input	CCR Max-Output	CCR Min-Input	CCR Max-Output	CCR Min-Input	CCR Max-Output	CCR Min-Input	CCR Max-Output
-Enrollment Student	-74.32%	-40.87%	-50.88%	-23.48%	-66.67%	-62.29%	-4.52%	0.0%
-Student / Teacher	-56.58%	0.0%	-80.17%	-69.11%	-11.61%	0.0%	-56.5%	-54.44%
-Master Registration	-56.58%	0.0%	-35.81%	0.0%	-11.61%	0.0%	-4.52%	0.0%
-PhD Registration	-56.58%	0.0 %	-35.81%	0.0%	-11.61%	0.0%	-4.52%	0.0%
+Graduation Number	324.21%	876.91%	15.34%	79.67%	164.27%	198.97%	0.0%	4.73%
+Master Graduates	754.41%	1867.63%	0%	55.87%	0.0%	13.13%	0.0%	4.73%
+PhD Graduates	0.0%	130.29%	22.51%	60.85%	34.34%	51.97%	-4.52%	95.99%
+ # of Conferences	0.0%	130.29%	27.55%	98.7%	80.51%	104.21%	87.14%	79.37%

7. References

[1] Armstrong, J. Scott, (1985) ," Long Range Forecasting" (New York: Wiley).

[2] B. Casu and E. Thanassoulis, "Evaluating cost efficiency in central administrative services in UK universities", Omega, 34(5) (2006) 417-426.

[3] Bel G. Raggad, "Neural network technology for knowledge resource management", Management Decision, 34 (2) (1996)20-24.

[4] Bijayananda Naik , Srinivsan Ragothaman , " Using neural networks to predict MBA student access" , College Student Journal (2004)

[5] Charnes, A., Cooper, W. and E. Rhodes, "Measuring the efficiency of Decision Making Units", European Journal of Operational Research, 26 (1978) 429-44.

[6] Coelli T, Prasada Rao DS, Battese GE. ,(1998), "An introduction to efficiency and productivity analysis. Boston/Dordrecht/ London: Kluwer Academic Publishers.

[7] Colbert, A., R.R. Levary, and M.C. Shaner, "Determining the relative efficiency of MBA programs using DEA", European Journal of operational Research, 125 (2000) 656-669.

[8] Cybenko, G. ,"Approximation by superpositions of a sigmoidal function.", Mathematical Control Signals Systems, 2 (1989) 303-314.

[9] Dennis Ahlburg, Michael McPherson, Morton Owen Schapiro, "Predicting Higher Education Enrollment in the United States: An Evaluation of Different Modeling Approaches ",(1994), http://www.williams.edu:/wpehe/DPs/DP-26.pdf

[10] Desheng(Dash) Wua, Zijiang Yang and Liang Liang, " Using DEA-neural network approach to evaluate branch efficiency of a large Canadian bank " , Expert Systems with Applications ,31(1) (2006) 108-115

[11] Dyson RG, Allen R, Camanho AS, Podinovski VV, Sarrico CS, Shale EA. ,"Pitfalls and protocols in DEA.", European Journal of Operational Research, 132 (2001)245–59.

[12] Elsa Cardoso, Maria Jose Trrigueiros , Patrciia Narciso, " A Balanced Scorecard Approach for Strategy – and Quality driven Universities" ,(2005) http://www.mc.manchester.ac.uk:80/eunis2005/medialibrary/papers/paper_174.pdf .

[13] François Tavenas ,(2003) , " Quality assurance : A Reference system for Indicators and evaluation Procedures " , European University Association , from web site: www.eua.be
[14] Gabriel Tavares, "A Bibliography of Data Envelopment Analysis (1978-2001) ", (2002) RUTCOR RESERCH REPORT, http://rutcor.rutgers.edu/~rrr.
[15] G. Peter Zhang, Christine X. Jiang, " An Evaluation of Neural Networks for Exchange Rate Forecasting" , Paper submitted to The 28th Annual Meeting of the Decision Sciences Institute San Diego, California (1997), http:// clt.astate.edu / jseydel / dsi97~1.doc.
[16] Gilbert , E.W., Krishnaswamy , C.R. , Pashley , M.M, " Neural Network Application in Finance : A Practical Introduction " (2000)
[17] Hornik, K., Stinchcombe, M., & White, H.," Multilayer feedforward networks are universal approximators.", Neural Networks,2 (1989) 359-366.
[18] J. Colin Glass, Gillian McCallion, Donal G. McKillop, Syamarlah Rasaratnam and Karl S. Stringer, "Implications of variant efficiency measures for policy evaluations in UK higher education", Socio-Economic Planning Sciences, 40 (2) (2006)119-142.
[19] Jill Johnes , " Data envelopment analysis and its application to the measurement of efficiency in higher education" , Economics of Education Review, 25(3) (2006)273-288.
[20] Jill Johnes , (In Press) , " Measuring teaching efficiency in higher education: An application of data envelopment analysis to economics graduates from UK Universities 1993 ", European Journal of Operational Research, In Press, Corrected Proof, Available online 17 May 2005.
[21] Marta Omero, Lorenzo D'Ambrosio, Raffaele Pesenti and Walter Ukovich, "Multiple-attribute decision support system based on fuzzy logic for performance assessment ", European Journal of Operational Research, 160(3) (2005) 710-725.
[22] Nevena .S, V. Angelova, "Measuring the efficiency of university libraries using Data Envelopment Analysis", INFORUM: 10th conferences on professional Information Resources, Prague , May (2004) 25-27.
[23] Nunamaker, T.R., "Using data envelopment analysis to measure the efficiency of non profit organizations: a critical evaluation" , Managerial and Decision Economics, 6 (1) (1985) 50-58.
[24] Othman Joumady and Catherine Ris.," Determining the relative efficiency of European Higher Education institutions using DEA " ,http://www.egss.ulg.ac.be/economie/ ris_appc.pdf
[25] Parag C. Pendharkar and James A., Rodger, "Technical efficiency-based selection of learning cases to improve forecasting accuracy of neural networks under monotonicity assumption" ,Decision Support Systems, 36(1) (2003) 117-136 .
[26] Parag C. Pendharkar," Scale economies and production function estimation for object-oriented software component and source code documentation size", European Journal of Operational Research ,172(3) (2006)1040-1050
[27] Roger Hartley, Saud M.Y. Almuhaidib, "User oriented techniques to support interaction and decision making with large educational databases" , Computers & Education, (2005).
[28] Smith, .J.N.D Gupta, "Neural Network in Business technique and applications for the operations researcher" , computers & operations research , 27 (2000)1023-1044.
[29] Tsoukalas , l.h ,and R.E.Uhrig , (1997) , " Fuzzy and Neural Approaches in Engineering", New York :Wiley and sons.
[30] Turban .E, Aron son J.E , Loang .T.P, (2005),"Decision Support Systems and intelligent systems " , 7th ed. , Prentic Hall.
[31] White G.P. , " A survey of recent management science applications in higher education administration", Interfaces, 17(2) (1987) 97-108.
[32] Weiss, S.M. and Kulikowski, C.A.(1991), " Computer Systems that learn", San Mateo, CA: Morgan Kaufman Publishers, Inc.
[33] W.K. Wang, " Knowledge-based decision support system for measuring the performance of government real estate investment ", Expert Systems with Applications, 29(4) (2005) 901-912 .
[34] Wong B.K, Bodnovich Ta, and Selvi .Y., "Neural Network application in business: a review and analysis of the literature (1988-1995)", Decision Support System, 19 (1997) 301-32.

A Hybrid Intelligent System for Arabic Handwritten Number Recognition

Reda M. Hussein
Faculty of computers & Information, Shiben El-Kom, Menoufia University, Egypt.
Reda3032@yahoo.com

W. F. Abd El-Wahed
Operations Research Dept. Faculty of computers & Information, Shiben El-Kom, Menoufia University, Egypt.

Fawzy Torkey *Operations*
Prof. of Computer Science & Engineering and President of Kafer El-Sheekh University, Egypt
Fatorkey@yahoo.com

Abstract

This paper shows how developments in the area of neural network combined with genetic algorithms can be used in the handwritten digit recognition. In this work, two approaches to the design of a feed-forward neural network that model the handwritten recognition system are discussed. The first approach focuses on constructing the network by using a trail-and-error method the second approach is responsible for determining the appreciate parameters of the neural network and its learning algorithm by the mean of genetic algorithms. Results show that using genetic algorithm for selecting the near optimal parameters of the neural network, is improving classification performance on handwritten digits.

Keywords*: Neural Networks; Genetic Algorithms; Handwritten numeral recognition*

1. Introduction

This work deals with neural network application in the field of pattern recognition and more specifically in handwritten recognition system. The application of NN in modeling non-linear systems, has a central drawback; the leak of a precise method to choose the appreciate topology, type of activation function, and parameters of learning algorithm. These tasks are usually based on a trail-and-error procedure performed by the developer of the model. In this approach, optimality or even near-optimality is not guaranteed, because the explored search space of the NN parameters is just a small portion of the whole search space and the type of search is rather random. To overcome this drawback, an automated method, based on the evolutionary properties of genetic algorithms (GAs), is developed. The role of GAs is to evolve several network architectures with different parameters so that the best possible combination is finally chosen.

Handwritten character recognition has been one of the most challenging tasks for artificial neural networks (ANN) designers. A number of researchers have recently applied neural network techniques to recognize the hand-written characters by using a genetic algorithm (GAs) approach. In [1] GA is used for optimally design the network architecture including number of hidden layers, number of neurons in each layer, connectivity and activation functions. And in [2] Except for the network architecture, the types of activation functions of the hidden and output nodes, as well as the type of the minimization approach of the back-propagation algorithm, are also included in the GA encoding. Also In [3] a genetic algorithm is used to obtain the optimal activation functions that vary according to some intermediate signals of the neural network. In [4], evolutionary programming was used for training neural networks, and in [5] a genetic algorithm was used for weight selection. Also In [6], changes in a neural network structure during training and operation were implemented by removing inactive synapses and inactive units. In [7] it is presented the evolution of neural networks for topology selection and weights through mutation although not directed to digit recognition.

In this paper improvements on the handwritten digit recognition accuracy are gained by a genetic selection of the parameters of the back-propagation learning algorithm (learning rate and momentum for all layers) in addition to connection weights.

The paper is organized as follow: next section presents the neural network architecture and its learning in with back-propagation algorithm. Section 3 introduces the Genetic Algorithms; Section 4 describes The Hybrid intelligent system used

for training the handwritten recognition system. Section 5 describes the dataset used for the handwritten digit recognition system. Section 6 presents a comparison to the two approaches and finally section 7 concludes the research study.

2. Neural Network

Consider a multi-layer feed-forward neural network with as shown in figure 1 with the following notation:

y_k : output of kth neuron of hidden layer

z_j : output of jth neuron of hidden layer

x_i : output of ith input to the neural network.

w_{kj} : weight between kth neuron and jth hidden neuron.

w_{ji} : weight between jth neuron and ith input.

net_k : output of linear combiner in kth neuron.

f_k , f'_k : activation function and its derivative of kth neuron

net_j : output of linear combiner in jth hidden neuron..

f_j , f'_j : activation function and its derivative of jth hidden neuron

d_k : desired output of kth neuron.

Each layer has units with sigmoid activation function that compute its output according to the following formula:

$$f(net) = \frac{1}{1+e^{-net}} \quad \text{..... (1)}$$

where, net is the weighted sum of the unit inputs plus a bias or offset term q .

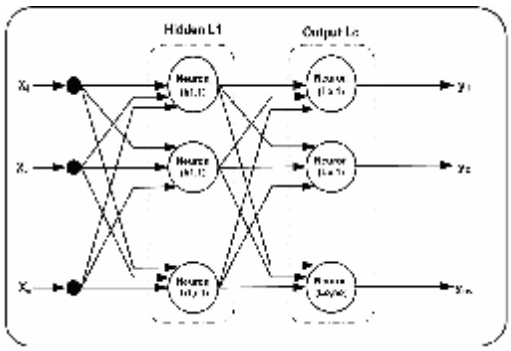


Figure 1:Multi-Layer feed forward Neural Network

The output of each neuron in hidden and output layer can be calculated as follow:

$$net_j = \sum w_{ji} x_i + q_j \quad \text{..... (2)}$$

$$z_j = f_j(net_j) \quad \text{..... (3)}$$

$$net_k = \sum w_{kj} x_j + q_k \quad \text{..... (4)}$$

$$y_k = f_k(net_k) \quad \text{..... (5)}$$

Once the network has been structured for a specific application, it is ready to be trained. Training a network means adapting its connection weights so that the network exhibits the desired computational behavior for all input patterns. Back-propagation known also as Error Back-Propagation or Generalized Delta Rule is the most widely used supervised training algorithm for neural networks. Connection weights of the neural network can be adapted by BP algorithm as follow:

Let the energy function chosen to be minimized is

$$E^p = \frac{1}{2}\sum_{k=1}^{m}(d_k^p - y_k^p)^2 \quad \text{..... (6)}$$

Update of output-layer weights

$$s_k^p = (d_k^p - y_k^p) f_k'(net_k^p) \quad \text{..... (8)}$$

$$\Delta w_{kj}^p(t+1) = h s_k^p z_j^p + a\Delta w_{kj}(t) \quad \text{..... (7)}$$

Updates of Hidden-Layer Weights

$$s_j^p = f_k'(net_k^p) \cdot \sum_{k=0}^{m} s_k w_{kj} \quad \text{..... (8)}$$

$$w_{ji}^p(t+1) = h s_j^p x_i^p + a\Delta w_{ji}(t) \quad \text{..... (7)}$$

where, g is the learning rate, which accelerates the learning procedure. Large values of g can cause oscillation, to avoid oscillation at large g, momentum term a is added to make the change in weights dependent on the past weight change.

The problem with ANN trained by back-propagation algorithm is that a number of parameters have to be set before any training can begin.

3. Genetic Algorithms

A GA is a mathematical search technique based on the principles of natural selection and genetic recombination [1]. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function). Some of the basic terminologies used in the field of genetic algorithms are:

- **Genotype** represents a potential solution to a problem, and is basically the string of values chosen by the user, also called chromosome.
- **Phenotype** is the meaning of a particular chromosome, defined externally by the user.
- **Chromosome** is a data structure that holds a "string" of task parameters, or genes. This string may be encoded as a binary bit-string or as an array of integers (floating point or real-coded representation).
- **Gene** is a subsection of a chromosome that usually encodes the value of a single parameter.
- The **fitness** of an individual is a value that reflects its performance (i.e. how well solves a certain task)
- **Recombination** or crossover produces new individuals in combining the information contained in the parents.
- **Mutation** occasionally injects a random alteration for one of the genes.

The process involved in GA optimization problems can be summarized as follows see figure 2:

1. Randomly generate an initial population of potential solutions.
2. Evaluate the suitability or 'fitness' of each solution.
3. Select two solutions biased in favor of fitness.
4. Crossover the solutions at a random point on the string to produce two new solutions.
5. Mutate the new solutions based on a mutation probability.
6. Goto 2.

Figure 2: Flowchart for Genetic Algorithm

4. The Hybrid Intelligent System

In this work a standard genetic algorithm is used [9, 15, 16, and 17]. The (learning rate and momentum for hidden and output layers) in addition to connection weights are the variable parameters of GA chromosome. The six steps followed by the genetic algorithm are as follows:

1- The initial population: This population includes 10 chromosomes that are obtained by randomly assigning parameters to each member.

2- Encoding: The real chromosomes of genomes representing the parameters

3- Training and testing. Each network of the population is trained by back-propagation for 500 iterations with training patterns.

4- The fitness. The fitness value for a given network is the Mean Square Error as in equation (6). The fitness value of the population is the average fitness value over all the members of the population.

5- The evolution stage. Three operators are applied over the population: selection, crossover and mutation operators. The operators are applied to the chromosomes of the input population, to produce the evolved new population.

6- The stopping criterion is satisfied when either the maximum number of generations is achieved

5. Training and Simulation

5.1. Data Description

Training and testing was carried out using the "Optical Recognition of Handwriting Digits" database, which is made available by E. Alpaydin, and C. Kaynak. The database can be downloaded at [18]. It consists of 3823 training pattern of Arabic digits. Patterns are 32x32 bitmap images see figure 3 for the digit 2, which are converted to a 4x4 block size to reduce the dimensionality of the inputs to the neural network to 8x8 bitmap images.

Figure (2) shows an example of converting 32x32 bitmap to 8x8 block bitmap. The conversion is achieved by taking the 32x32 binary images as inputs and groups neighboring pixels in blocks.

The sum of "ON" pixels in each block is used to create a pixel in resulting image. The resulting image is then converted to a vector of input values and fed to the neural network as a single input pattern.

Output patterns are divided to ten classes from 0 to 9 as shown in Figure (3). The testing sets, different from all training sets, are composed of 1797 patterns.

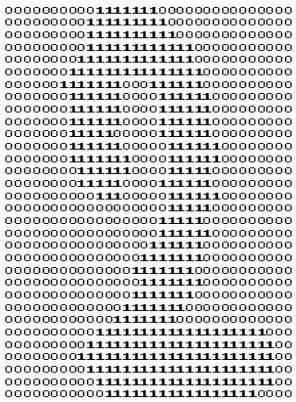


Figure 3: 32x32 bitmap of digit 2

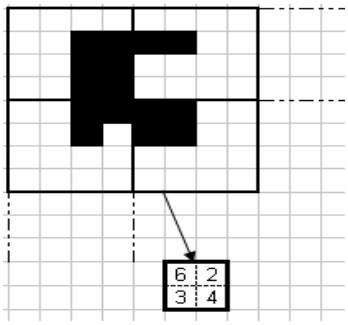


Figure 4: 32x32 to 8x8 image conversion using 4x4 block size.

5.2. The Model

Neural network used has 64 inputs, one hidden layer containing 15 neuron and output layer of 10 neurons. In addition to, two pairs of learning rate and momentum for each layer. All parameters of the network including weights were randomly initialized.

6. Experimental Results

In order to quantify the recognition capability of neural network configured with genetic algorithm, the analysis was performed in two stages as follow :

6.1. **Trail-and-Error Training** In training each problem using BP, learning rates and momentums of each layer could have been manipulated to find the best network configuration. The different combinations of the learning rates and momentums are used to try to find the right combination that will allow the solution to escape local minima but not skip over the global solution. The epoch is defined as one complete pass through the data set. For the handwritten number recognition system, the learning rate was set at 1.0 and the momentum factor set to 1.0. for each layer Both of these parameters were reduced with a reduction factor of 0.1 for every 500 epochs. This procedure was repeated twice with different initial weights. The total number of obtained network was 2x10x10=200 networks. This was done in order to keep the solution from oscillating and therefore helping to converge upon a solution. The best network of all 200 networks was then chosen and tested.

6.2. **Training with Genetic Algorithm** The model that described in section 5 was fed into the GA process. The basic parameters of GA that must be explored are the population size, the probability of crossover p_c , the probability of mutation

p_m and the number of generations. To determine the best possible values of these parameters, a number of experiments were carried out. The type of crossover used was the most common one, the one-point crossover [21]. The best values of p_c and p_m obtained are 0.9 and 0.01. the runs of the GA process were made with a population size of 10 networks trained with BP for 500 epoch for 20 generation (i.e. 200 network). The best network was then tested.

6.3. **Results** The performances of the two approaches discussed later were compared using two measures the MSE and the percentage of correction.
For the Trail-and-Error approach, figure 5 illustrates that the final MSE has a large variation over all networks. This means that the optimal is not guaranteed.

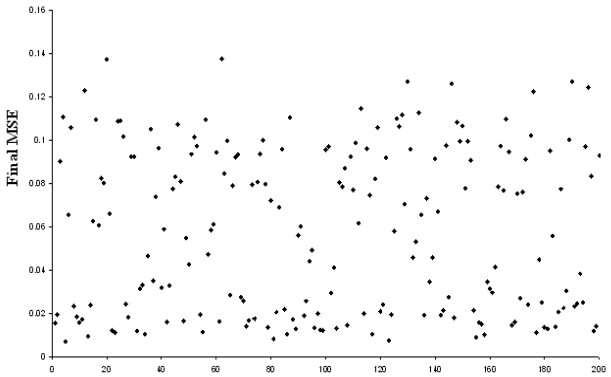


Figure 5: Best MSE over 200 run using Trail-and-Error

By selecting the best network obtained from this approach, and applying the training data as a test set, the achieved MSE is varying from 0.0013 to 0.0076 with average MSE 0.0043 as shown in figure 6. And the percentage of correction is varying from 96.3% to 100% with average value of 98.69%. the confusion matrix of the desired output versus the actual output is shown in figure 7

Perf.	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
MSE	0.00303	0.00683	0.0013	0.0041	0.0043	0.00398	0.0043	0.0016	0.0076	0.0066
% Correct	98.6702	97.9434	99.474	98.9717	99.742	99.2021	98.939	100	96.316	97.644

Figure 6: MSE and Percentage correction of training data trained with BP

Output / Desired	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
O(0)	371	3	1	0	0	0	1	0	0	1
O(1)	0	381	0	0	0	0	1	0	5	1
O(2)	0	0	378	0	0	1	1	0	0	0
O(3)	0	0	0	385	0	1	0	0	0	1
O(4)	3	0	0	0	386	0	0	0	3	1
O(5)	0	0	0	2	0	373	0	0	2	1
O(6)	2	1	1	0	1	0	373	0	1	0
O(7)	0	1	0	0	0	0	0	387	0	0
O(8)	0	1	0	0	0	0	1	0	366	4
O(9)	0	2	0	2	0	1	0	0	3	373

Figure 7: Confusion Matrix of training data trained with BP

For the test set, the achieved MSE is varying from 0. 0.0040 to 0.0344 with average MSE 0.016 as shown in figure 8. And the percentage of correction is varying from 77% to 98.3% with average value of 91.78%. the confusion matrix of the desired output versus the actual output is shown in figure 9

Perf.	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
MSE	0.0069	0.0225	0.0040	0.0167	0.0151	0.0175	0.0074	0.0128	0.0344	0.0228
% Correct	95.5056	90.1099	98.3051	90.7104	95.0276	95.0549	96.1326	89.944	77.011	90.000

Figure 8: MSE and Percentage correction of testing data trained with BP

Output / Desired	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
O(0)	170	2	0	0	3	2	1	0	0	0
O(1)	0	164	0	0	2	1	1	0	15	2
O(2)	0	1	174	1	0	2	0	0	0	0
O(3)	0	0	1	166	0	0	0	0	5	7
O(4)	8	1	0	0	172	0	4	1	1	2
O5)	0	0	0	1	0	173	0	13	10	4
O(6)	0	0	0	0	0	1	174	0	0	0
O(7)	0	0	1	0	1	0	0	161	0	0
O(8)	0	8	1	9	3	0	1	2	134	3
O(9)	0	6	0	6	0	3	0	2	9	162

Figure 9: Confusion Matrix of testing data trained with BP

For the GAs approach, as can be seen, figure 10shows the best MSEs found after each generation. The best solution was found after 15 generation. In figure 11, the corresponding average MSEs of the entire population after each generation are shown. One can see that the average performance of the population generally improves with the comparison of figure 5.

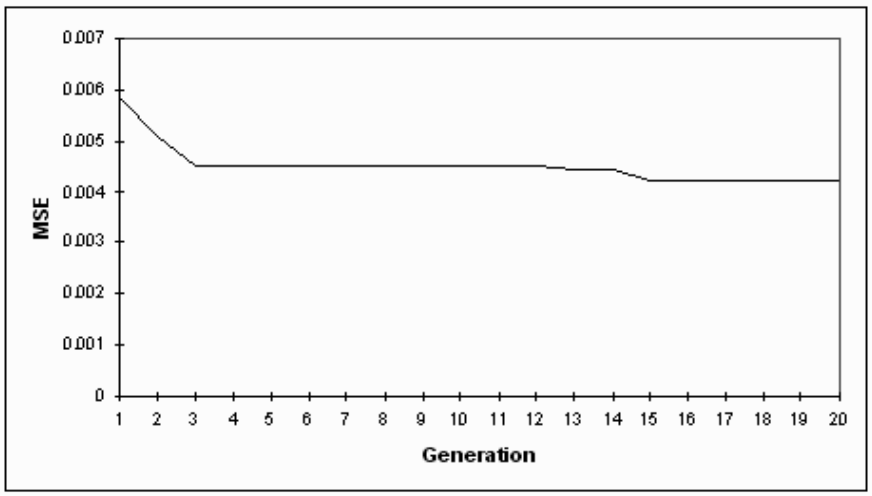


Figure 10:Best Fitness (MSE) versus Generation

The best network obtained from this approach, For the test set, the achieved MSE is varying from 0.0012 to 0.0055 with average MSE 0.0035 as shown in figure 12. And the percentage of correction is varying from 98.42% to 100% with average value of 99.5%. the confusion matrix of the desired output versus the actual output is shown in figure 13

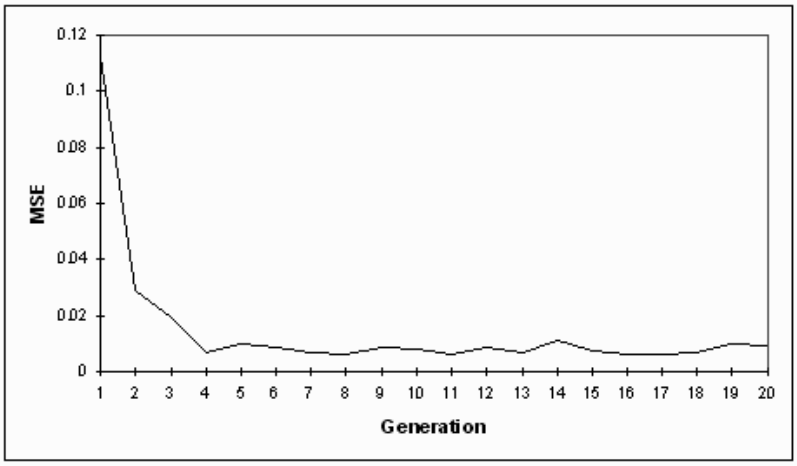


Figure 11:Average Fitness (MSE) versus Generation

For the test set, the achieved MSE is varying from 0.0031 to 0.0207 with average MSE 0.0104 as shown in figure 14. And the percentage of correction is varying from 89.01% to 98.87% with average value of 95.91%. The confusion matrix of the desired output versus the actual output is shown in figure 15

Perf.	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
MSE	0.00123	0.00465	0.003	0.00441	0.0035	0.00257	0.00299	0.002	0.0055	0.0051
Percent Correct	99.734	99.4859	99.737	98.9717	100	99.734	99.7347	100	98.421	99.215

Figure 12: MSE and Percentage correction of training data trained with GBP

Output / Desired	*O(0)*	*O(1)*	*O(2)*	*O(3)*	*O(4)*	*O(5)*	*O(6)*	*O(7)*	*O(8)*	*O(9)*
O(0)	375	0	0	0	0	0	0	0	2	0
O(1)	0	387	0	0	0	0	1	0	2	1
O(2)	0	0	379	0	0	0	0	0	0	0
O(3)	0	0	0	385	0	1	0	0	0	0
O(4)	0	0	0	0	387	0	0	0	0	1
O(5)	0	0	0	3	0	375	0	0	0	0
O(6)	1	0	1	0	0	0	376	0	2	0
O(7)	0	1	0	0	0	0	0	387	0	0
O(8)	0	0	0	0	0	0	0	0	374	1
O(9)	0	1	0	1	0	0	0	0	0	379

Figure 13: Confusion Matrix of training data trained with GBP

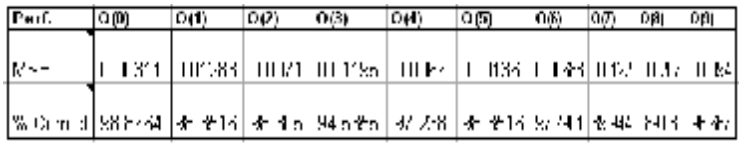


Figure 14: MSE and Percentage correction of testing data trained with GBP

Output / Desired	O(0)	O(1)	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)	O(8)	O(9)
O(0)	176	0	0	0	0	0	0	0	0	0
O(1)	0	179	2	0	2	0	2	0	8	2
O(2)	0	0	174	4	0	0	0	0	0	0
O(3)	0	0	0	173	0	1	0	0	5	1
O(4)	0	1	0	0	176	0	1	1	0	0
O(5)	2	0	0	2	0	179	0	13	2	1
O(6)	0	1	0	0	0	0	177	0	0	0
O(7)	0	0	0	0	1	0	0	161	0	1
O(8)	0	1	1	2	1	0	1	1	155	1
O(9)	0	0	0	2	1	2	0	3	4	174

Figure 15: Confusion Matrix of testing data trained with GBP

7. Conclusions

The experimental results show that genetic algorithms have a strong potential to find good solutions for the neural network configuration problem, and therefore is a good alternative to select the most appropriate network for a given task. For the handwritten digit recognition problem, the average recognition accuracy of 95.2 % was found by the Trail-and-Error approach. The neural network was improved by selecting configuration parameters of the neural network using GA to 97.7%.

8. References:

[1] Manojit Dam, Deoki N. Saraf, Design of neural networks using genetic algorithm for on-line property estimation of crude fractionator products, Computers and Chemical Engineering vol. 30 pp. 722–729, 2006

[2] Konstantinos P. Ferentinos,Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms,Neural Networks vol. 18 pp. 934–950, 2005 Elsevier Ltd.

[3] S.H. Ling, H.K. Lam, and F.H.F. Leung, A variable-parameter neural network trained by improved genetic algorithm and its application,¨ in Proc. Int. Joint Conf. Neural Networks 2005 (IJCNN2005), Montreal, Canada, 31 Jul.-4 Aug. 2005, pp. 1343-1348.

[4] EN-HUI Zheng', Min Yang, Tuning of Neural Networks based on Genetic Algorithm and Statistical Learning Theory, Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004

[5] Montana DJ, "Neural Network Weight Selection Using Genetic Algorithms, " in Intelligent Hybrid Systems, Edited by Goonatilake S & Khebbal S, John Wiley & Sons, pp.85-104, 1995.

[6] Odri, S.V., Petrovacki, D.P., Krstonosic, G.A., "Evolutional Development of a Multilevel Neural Network, " Neural Networks, Vo1.6, pp.583-595, 1993.

[7] Braun, H., Weisbrod, J., 1993, Evolving Neural Networks for Application Oriented Problems, Proceedings of the Second Annual Conference on Evolutionary Programming, eds. D.B. Fogel and W. Atmar, La Jolla, CA: Evolutionary Programming Society, pp.62-71.

[8] Haykin S, "Multilayer Perceptrons, " in Neural . Networks: A Comprehensive Foundation, John Griffin (ed), IEEE Press, Macmilan College Pu CO., pp. 138-229, 1994.

[9] Goldberg, D., 1989, Genetic Algorithms in Search, Optimization and Machine Learning, Reading MA: Addison-Wesley.

[10] Koza, J., 1992, Genetic Programming, Cambridge, MA, MIT Press.

[11] kbeiro, J.L. and Treleaven P.C., 1994, Genetic-Algorithm Programming Environments, Computer, pp.28-43.

[12] Fogel, D.B., "Theoretical and Empirical Properties of Evolutionary Computation," in Evolutionary Computation: Toward a New Philosophy of Machine Iritelligence, IEEE Press, pp.121-186, 1995.

[13] Porto, V.W., Fogel D.B., Fogel L.J., "Alternative Neural Network Training Methods, " IEEE Expert, Vol 10, No.3, pp.16- 22, 1995.

[14] Kosko B. Neural Networks and Fuzzy Systems – A Dynamical Systems Approach to Machine Intelligence. Prentice Hall, 1992

[15] Davis, L. (ed), 1987, Genetic Algorithms and Simulated Annealing, Pitman, London.

[16] Fogel, D.B., "Theoretical and Empirical Properties of Evolutionary Computation", in Evolutionary Computation:

Toward a New Philosophy of Machine Iritelligence, IEEE Press, pp.121-186, 1995.
[17] Davis, L. (ed), 1991, Handbook of Genetic Algorithms, New York: Van Nostrand Reinhold.
[18] "Optical Recognition of Handwriting Digits" database ftp://ftp.ics.uci.edu/pub/machine-learning- tabases/optdigits/
[19] Mandana Ebadian Dehkordi, Nasser Sherkat, Tony Allen, "Handwriting style classification" Volume: 6, Issue: 1, August, 2003. pp. 55 - 74.
[20] Seok; Suen, Ching Y.," A class-modular feedforward neural network for handwriting recognition" Volume: 35, Issue: 1, January, 2002. pp. 229-244
[21] Goldberg, D. E. " Genetic Al Algorithms in search, optimization and machine learning." Reading, MA: Addison , 1989.