# Fault Diagnosis of Discrete Event Systems Using Hybrid Petri Nets

R. Rangarangi Hokmabad, M. A. Badamchizadeh, and S. Khanmohammadi

*Abstract*—**A new method for fault diagnosis of discrete event systems modeled by Neural Petri Nets (NPNs) is presented in this paper. Assuming that the PN structure and initial marking are known, faults are modeled by unobservable transitions. Neural networks (NNs) have important role to improve the method. The outputs of them are connected to unobservable transitions and yield the percentage of faults that may happen for prioritize the faults by online computation of the set of possible fault events. In this method the operator checks the fault that has more value at first. So we reduce the time that spends for repairing the system. Moreover, the graphical representation of the nets allows the diagnoser agent to compute off-line reduced portions of the net in order to improve the efficiency of the online computation, without a big increase in terms of memory requirement.**

*Index Terms*—**Discrete event systems (DES), fault diagnosis, Neural networks (NNs), Petri nets (PNs)**

## I. Introduction

The diagnostics of industrial processes is a scientific discipline aimed at the detection of faults in industrial plants, their isolation, and finally their identification. Its main task is the diagnosis of process anomalies and faults in process components, sensors and actuators. Early diagnosis of faults that might occur in the supervised process renders it possible to perform important preventing actions. Moreover, it allows one to avoid heavy economic losses involved in stopped production, the replacement of elements and parts [1].

The operation of large and complex systems requires that the coordination systems possess fault recovery capabilities. In automated manufacturing systems, this ability is included mainly to eliminate unnecessary risks to humans or hazardous situations into the system as well as to maintain the production rate. However, introducing this capability to the coordination system possesses challenging problems that have been addressed through several approaches and methods. Most of these approaches include stages, such as detection, isolation, and confinement of the fault [2].

Discrete event systems (DES) based methodologies for fault diagnosis are applicable not only to systems normally modeled as DES, but also to systems that traditionally are treated as continuous-time dynamic systems. In general, DES approaches to fault diagnosis are suitable for failures that cause a distinct change in the state of system components but do not bring the system to a halt: examples are equipment failures (stuck failure of valves, stalling of actuators, controller failures, etc.) usual in flight control systems or heating and air conditioning systems, and process failures (buffer overflow) usual in manufacturing systems [3].

Recently, neural networks (NNs) have been applied to the fault diagnosis problem because of their good capabilities in function approximation. Specifically, online approximation method using NNs has been presented for identifying the fault functions [4], [5].

In [6] a new architecture for a fault diagnosis competitive neural network is introduced. In this system, the test matrix and the probability vector of faults are not known a prior. The neural system starts from a completely vague state and the weights of connections, which affect the possibility of detecting the fault in each unit, are modified during the learning procedure on the base of different tests.

In [7] faults are not explicitly taken into account in the model, and two types of faults have been defined: a place fault that corrupts the net marking, and a transition fault that causes an incorrect update of the marking after events occurrences and In [3] just the maximum and minimum of faults are detected and they are not sure if the specific fault would occurs or not.

In this paper a new technique for the fault diagnosis is used. Faults are modeled by unobservable transitions. Moreover, we assume that there may be additional unobservable transitions associated with the system legal behaviour and that the marking reached after the firing of any transition is unknown. The Petri nets (PNs) are connected to NNs and the weights of NNs are training then the outputs of NNs specify the percentage of each fault that may happen. The proposed diagnoser works on-line: it waits for the firing of an observable transition and employs an algorithm based on the definition and solution of some integer linear programming problems to decide whether the system behaviour is normal or exhibits some possible faults. The results characterize the properties that the PN modeling the system fault behaviour has to fulfill in order to reduce the on-line computational effort.

A good compromise to speed up the online diagnosis is to precompute something offline. This is particularly efficient when PN are used to model the plant. In fact, working on the net structure some useful information can be computed offline improving the efficiency of the online diagnosis, without a big increase in terms of memory request. In this paper it is shown that the programming problems to be solved by the diagnoser can be formulated on reduced portions of the net properly computed offline.

Manuscript received February 20, 2010; revised March 25, 2012.

Roya Rangharanghi Hokmabad has been graduated from University of Tabriz (e-mail: r.rangarangi87@ms.tabrizu.ac.ir).

Mohammad Ali Badamchizadeh is with the Department of Control, Faculty of Electrical and Computer Engineering, University of Tabriz (e-mail: mbadamchi@tabrizu.ac.ir).

Sohrab Khanmohammadi is with the Control Engineering Department, Faculty of Electrical and Computer Engineering, University of Tabriz (e-mail:khan@tabrizu.ac.ir).

## II. Backgrounds and Basic Assumptions

### A. Basic Petri Nets Notation

Petri nets (PNs) are a graphical and mathematical modeling tool applicable to many systems. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. As a graphical tool, PNs can be used as a visual communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems. A Petri net (PN) is a particular kind of directed graph, together with an initial state called the initial marking, $M_0$. The underlying graph N of a PN is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. 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 of k parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns to each place a non negative integer. If a marking assigns to place p a nonnegative integer k, we say that p is marked with k tokens. Pictorially, we place k black dots (tokens) in place p. A marking is denoted by M, an m-vector, where m is the total number of places. The $p^{th}$ component of M, denoted by M(p), is the number of tokens in place p [8].For a complete review on PNs refer to [8]. A formal definition of a PN is given in Table I.

### B. Marking Projections

*Definition :* A PN [8] is a bipartite graph described by *PN=(P, T,* ***Pre****,* ***Post*****)**, where *P* is a set of places with cardinality *m*, *T* is a set of transitions with cardinality *n*, ***Pre***: $P\times T\rightarrow$N and ***Post***: $P\times T\rightarrow$N are the *pre-* and *post-incidence matrices*, respectively, which specify the arcs connecting places and transitions. Matrix ***C=Post-Pre*** is the $m\times n$ *incidence matrix* of the net *PN*. Table I shows a formal definition of a PN.

For the pre- and post-sets we use the dot notation, e.g., $\cdot t=\{p\in P: Pre(p,t)>0\}$. The state of a PN is given by its current marking, which is a mapping ***M***: $P\rightarrow$N, assigning to each place of the net a nonnegative number of tokens. A PN system $\langle <PN,M_0>$ is a net *PN* with an initial marking $M_0$. A transition $t_f\in T$ is enabled at a marking ***M*** if and only if (iff) for each $p\in .t_j$, it holds $\boldsymbol{M(p)\geq Pre(p,t_j)}$ and we write $M[t_j>$ to denote that $t_j\in T$ is enabled at marking ***M***. Let $\sigma=t_{b_1}t_{b_2}...t_{b_k}$ be a sequence of transitions and let $k=|\sigma|$ be its length, given by the number of transitions that $\sigma$ contains. If a transition $t\in T$ appears in the sequence *σ*, we write $t\in\sigma$. Moreover, the notation $M[\sigma>\succ$ indicates that the sequence σ is enabled at ***M*** and $M[\sigma>M'$ indicates that the enabled sequence *σ* may fire at ***M*** yielding ***M'***. We also denote $\vec{\sigma}(t)=q$ the firing vector associated with a sequence σ, i.e., $\vec{\sigma}(t)=q$ if transition *t* is contained *q* times in σ. A marking ***M*** is said reachable from $<PN,M_0>$ iff there exists a firing sequence *σ* such that $M_0[\sigma>M$. The set of all markings reachable from $M_0$ defines the reachability set of $<PN,M_0>$ and is denoted by $R(PN,M_0)=\{M\,|\,\exists\sigma: M_0[\sigma>M\}$.

TABLE I: Formal Definition of a Petri Net

| |
|---|
| A Petri net is a 5-tuple, PN = (P, T, F, W, $M_0$) where: |
| $P=\{p_1, p_2,..., p_m\}$ is a finite set of places, |
| $T=\{t_1, t_2,..., t_n\}$ is a finite set of transitions, |
| **Pre** : P × T → N (**Post** : P × T → N) |
| **C=Post-Pre** |
| F $\subseteq$ (P x T) U (T x P) is a set of arcs (flow relation), |
| W: F {1, 2, 3, ... } is a weight function, |
| Mo: p → {0, 1, 2, 3, ... } is the initial marking, |
| P ∩ T = $\phi$ and P U T ≠ $\phi$ . |
| A Petri net structure N = (P, T, F, W) without any specific initial marking is denoted by N. |
| A Petri net with the given initial marking is denoted by (N, $M_0$). |

### C. Neural Petri Net

Artificial neural networks (ANN) are highly parallel and distributed computation structures that can learn from experience and perform inferences. PNs, on the other hand, provide an effective modeling framework of distributed systems. The basic concepts of PNs are utilized to develop ANN-like multilayered PNs architectures of distributed systems [9].

The NPN is formally defined as a 6 tuple: NPN = *(P, T, Z, A,C,* $M_0$*)* where:

*P*: is a set of places;
*T*: is a set of transitions;
*Z*: is set of arcs, Z $\subseteq$ (P×T) ∪ (T×P);
*A*: is a pattern of connectivity among places and transitions;
*C*: is a set of states of outputs of NNs.

The resulting NPN is a feed forward network with alternating columns of transitions.

In [10], they built ANN-like architectures of distributed intelligence that can learn from experience. The resulting Neural Petri Net (NPN) is feedforward network with alternating columns of places ad transition.

The NPN is a pure PN (self –loops are not allowed). This leads to a feedforward architecture. The interaction of ANN with the environment is through the unobservable transitions and places. Fig.1 shows a simple NPN. $t_6$ is an unobservable transition and models a faulty behavior of a system. The inputs of ANN $\{X_1...X_5\}$ are connected to places (a, b…e) and the output of ANN yields the percentage of fault $t_6$ that may happen.

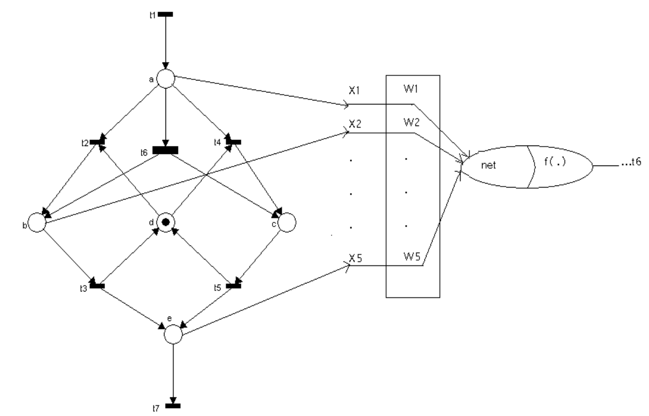



Fig. 1. A neural Petri net

## III. Generalized Marking

Let ***m*** be the current net marking, *C*=***Post-Pre*** be the incidence matrix and σ be a firing sequence of transitions $\sigma = t_1 ... t_k$ such that $m[t_1 > m_1 [t_2 > m_2 ... [t_k > m_k$, and this is denoted as $m[\sigma > m_k$. If a sequence σ fires, a new marking ***ḿ*** is reached. From the state equation it follows that for the firing count vector σ it is possible to write
$m + C\sigma = m' \geq 0$.

Suppose that where ε is a sequence of unobservable transitions and $t \in T_0$. Let $\mu = m + Ce_t = m + C(.,t)$. It may happen that ***μ*** has negative components, since t may not be enabled under the marking ***m***. The negative components in ***μ*** mean that the unobservable sequence *ε* must have fired in order to explain the firing of *t*, which is the unique observed event. A marking that may have negative components is called *g-markin g*[2].

Suppose that a fault event is associated to the unobservable transition $t_f$, and that one wants to know if $t_f$ has occurred prior to the observation of *t.* Note that it is not necessary to compute explicitly *ε* or its firing count vector ***ε***, but simply to check if $\varepsilon(t_f)$ is greater than zero.

The following example shows how the evolution rules for the g-marking given above can lead to negative marking components. For a complete review on g-marking refer to [3].

*Example :* Consider the net *N* in Fig. 2 and let $\mu_0 = [0\ 0\ 0\ 1\ 0\ 0]^T$ be the initial g-marking of the net and $\{t_6 ... t_8\}$ be unobservable transitions. If the firing of $t_3 \in T_0$ is observed, then $t_3$ may fire, since an observable transition is always enabled under any g-marking. The firing of $t_3$ yields the g-marking $\mu_1 = mu_0 + C(.,t_3) = [0\ \text{-}1\ 0\ 2\ 1\ 0]^T$. The negative marking $\mu_{1|p_2} = -1$ means that an unobservable sequence must have fired to explain the firing of $t_3$.

Throughout this paper, the negative components of a g-marking represent the tokens that are needed to explain either the firing of an observed transition, or the firing of an unobservable transition that must have fired.

As far as the fault diagnosis is concerned, the g-markings allow the fault diagnosis agent to store in a compact way all the needed information about the state space estimation.

## IV. Fault Detection

The approach used in this paper is based on the fact that the firing of an observable transition requires a proper marking of its input places. If the same event is associated to more than one transition, the observation of an event could not necessarily correspond to the firing of a single transition.

Taking into account this problem complicates too much the approach. A similar assumption is not required for unobservable transitions, which can be assumed to be associated to an arbitrary event, since such event is unobservable.

If an observable transition fires then if the new marking has negative components, the fault diagnoser will start working. The inputs of ANN are connected to the places. If the number of tokens in a place was positive, the related input of ANN would be 0 and if the number of tokens in another place was negative the related input of ANN would be 1. The outputs of ANN yield the percentage of faults that may happen. In this way we prioritize the faults and at first the operator checks the fault that has more value. So in this method we reduce the time that spends for repairing the system. The difference between outputs of ANN and the value that operator had have after checking the system be the error and the weights of ANN is training then these stages repeat again.

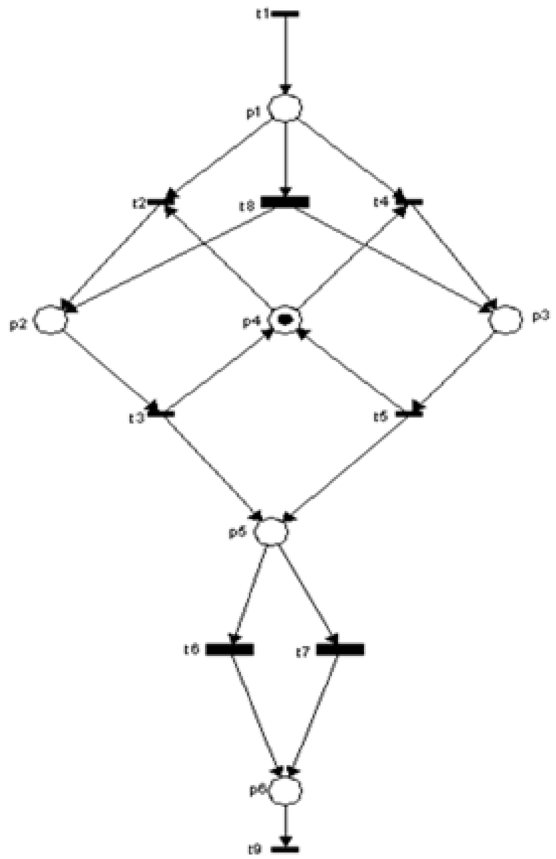



Fig. 2. Example of Petri net model

*Example: A Manufacturing System*

Let us consider the manufacturing system whose PN model is shown in Fig. 3, with $T_0 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$ and $T_{u_0} = \{t_{11}, t_{12}, t_{13}\}$.

This manufacturing system consists of three machines, $M_1$, $M_2$ and $M_3$, which process parts conveyed on equipments. The system has two operators, $F_1$ and $F_2$. $F_1$ could manage $M_1$ and $M_2$. $F_2$ could manage $M_1$ and $M_3$. In each equipment the operation is performed in two stages.

Stage1: works with $M_1$.

Stage2: works with $M_2$ or $M_3$

- $t_1$ fires when a new equipment is received.
- $t_2$ fires when $F_1$ starts working on equipment with $M_1$.
- $t_3$ fires when $F_1$ finishes working on equipment with $M_1$.
- $t_4$ fires when $F_2$ starts working on equipment with $M_1$.
- $t_5$ fires when $F_2$ finishes working on equipment with $M_1$.
- $t_6$ fires when $F_1$ starts working on equipment with $M_2$.
- $t_7$ fires when $F_1$ finishes working on equipment with $M_2$.
- $t_8$ fires when $F_2$ starts working on equipment with $M_3$.
- $t_9$ fires when $F_2$ finishes working on equipment with $M_3$.
- $t_{10}$ fires when the operation on equipment is finished.
- $t_{11}, t_{12}$ and $t_{13}$ model the faulty behaviors of $M_1, M_2$ and $M_3$.
- The initial number of tokens in $p_4, p_5, p_6, p_7$ and $P_8$ model the fact that the machines and operators are waiting.

Some execution steps of the proposed method are reported in Tab. II. In this case, if the estimated g-marking has negative components, then the firing of transition that are modeled as faults would test and NNs that are connected to these transitions would work and compute the percentage of faults may happen to perform fault detection and identification.

## V. Conclusion

The paper addresses the fault detection problem of Discrete Event Systems (DES) and proposes an on-line diagnoser in a Neural Petri Net (NPN) framework. A procedure observes and stores the sequence of system events and decides on-line whether the system behavior is normal or some faults may have occurred. To this aim, at each observed event it is provide the possible occurred faults or certifies the system normal behaviour. In order to achieve this result, *g-markings* are introduced in this paper. G-markings are net markings that may have negative components and whose estimation is always unique. The online computation consists of solving programming problems formulated on net structure and based on g-markings.

TABLE II: Execution of the Fault Detection Algorithm on The Net of Fig. 3.

| ***Action*** | ***μ*** | ***The percentage of fault $t_{11}$ may happen*** | ***The percentage of fault $t_{12}$ may happen*** | ***The percentage of fault $t_{13}$ may happen*** |
|---|---|---|---|---|
| Initialization | [0 0 0 1 1 1 1 1 1 0 0 0 0] | 33% | 33% | 33% |
| $t_1$ fires | [1 0 0 1 1 1 1 1 1 0 0 0 0] | --- | --- | --- |
| $t_4$ fires | [0 0 0 0 1 1 1 0 0 1 0 0] | --- | --- | --- |
| $t_8$ fires | [0 -1 0 0 1 0 1 -1 0 1 0 1] | 10.35% | 15.63 % | 80.2% |
| $t_9$ fires | [0 -1 1 0 1 1 1 0 0 1 0 0] | 12.32% | 40.33 % | 39.92 % |
| $t_5$ fires | [0 0 1 1 1 1 1 1 1 0 0 0 0] | --- | --- | --- |
| $t_2$ fires | [-1 0 1 0 1 1 0 1 1 0 0 0] | 75.32% | 14.26 % | 10.23 % |
| $t_6$ fires | [-1 -1 1 0 0 1 -1 1 1 0 1 0] | 80.64% | 65.45 % | 12.78 % |

With respect to the approaches proposed in the related literatures, the proposed method specify the percentage of faults in order to provide a reasonably efficient method suitable use with large systems. However, the algorithm use some off-line calculations based on the structure of the considered Petri net (PN) system in order to decrease the memory capacity. In this way, the proposed fault detection technique can be more easily applicable.

Further improvements in the efficiency of the proposed method could be obtained if we assume that after an event sequence occurrence the reached marking is known or univocally determined. In this situation, an incremental solution approach could be devised. However, identifying the conditions necessary to univocally determine the reached marking is expected to require a significant amount of effort to be specified and developed. Hence, this issue will be tackled in a successive work.

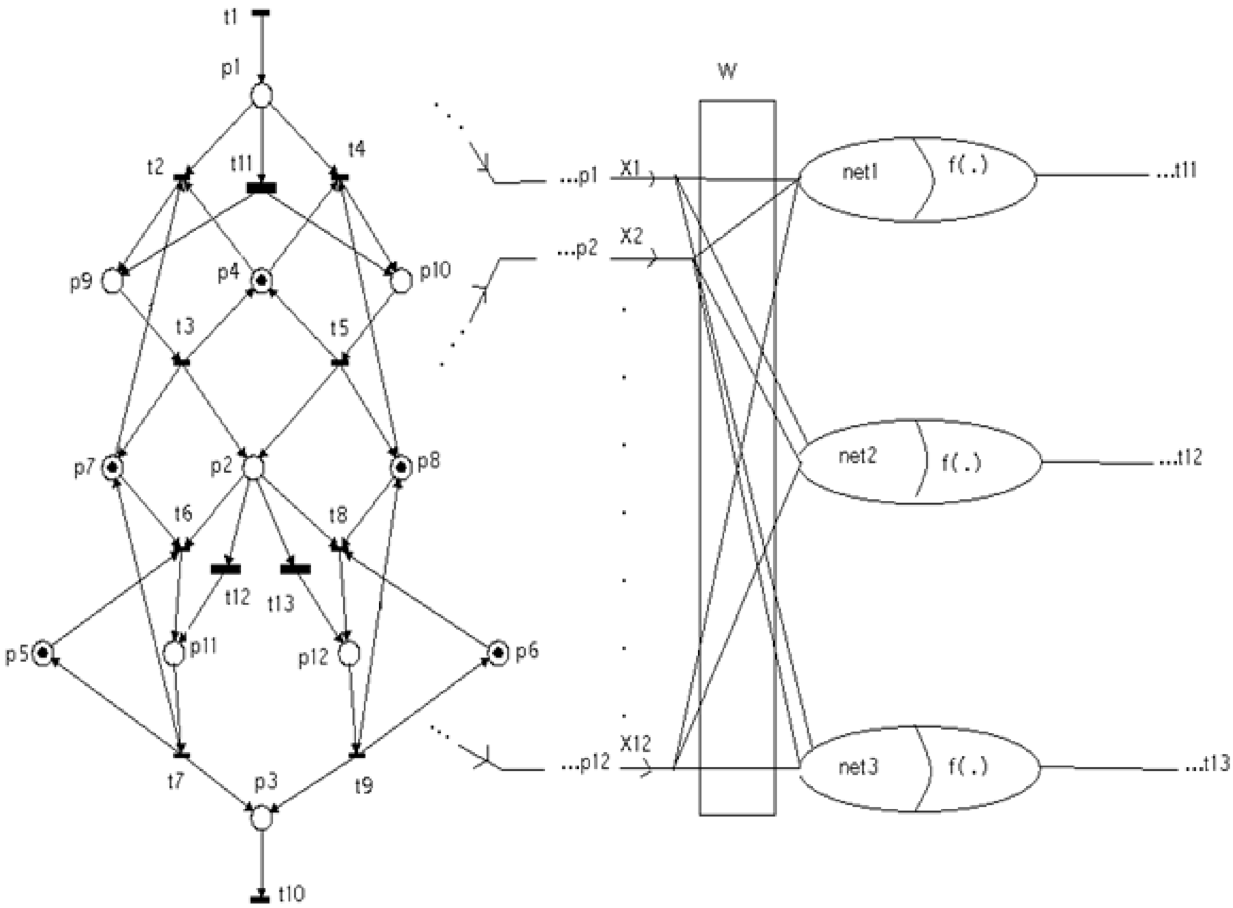



Fig. 3. NPN model of the manufacturing system.


## REFERENCES

[1] K. Patan, Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes, Springer Pub, 2008.

[2] R. Isermann, "Process fault detection based on modeling and estimation methods. A Survey," Automatica, vol. 20, pp. 387–404, 1984.

[3] F. Basile, P. Chiacchio, and G. Tommasi, "An efficient approach for online diagnosis of discrete event systems," *IEEE Trans. Automatic control*, vol. 54, no. 4, pp. 748-759, Apr. 2009.

[4] S. Huang and K. K. Tan, "Fault Detection and Diagnosis Based on Modeling and Estimation Methods," *IEEE Trans*. On neural networks, vol. 20, no. 5, pp. 872-881, May 2009.

[5] A. T. Vemuri and M. M. Polycarpou, "Neural-network-based robust fault diagnosis in robotic systems," *IEEE Trans*. Neural Netw, vol. 8, no. 6, pp. 1410–1420, Nov. 1997.

[6] S. Khanmohammadi, I. Hasanzadeh, and H. R. Zarei Poor, "Fault diagnosis competitive neural network with prioritized modification rule of connection weights," *ELS. Artificial Intelligence in Engineering*, vol. 14, pp. 127-132, 2000.

[7] Y. Wu and C. N. Hadjicostis, "Algebraic approaches for fault identification in discrete-event systems," *IEEE Trans. Automat. Control*, vol. 50, no. 12, pp. 2048–2053, Dec. 2005.

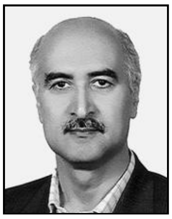
[8] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[9] P. T. T. Binh and N. D Tuyen, "Fault Diagnosis of Power System Using Neural Petri Net and Fuzzy Neural Petri Net," *IEEE*, 2006.

[10] Richard Zurawski and Mengchi Zhou "Petri Nets and industrial application: A tutorial," *IEEE Trans. on industrial electronic*, vol. 41, no. 6, Dec, 1994.


**Roya Rangharanghi Hokmabad** was born in Tabriz, Iran in 1982. She received her B.S. degree in Electrical Engineering from Azad University of Tabriz, Iran in 2005 and her Ms. C. degree in Control engineering form University of Tabriz in 2011. Her research interests are fault tolerant, Neural networks, Intelligent control.

**Mohammad Ali Badamchizadeh** was born in Tabriz, Iran, in December 1975. He received the B.S. degree in Electrical Engineering from University of Tabriz in 1998. He received the M.Sc. and Ph.D. degree in Control Engineering from University of Tabriz in 2001 and 2007, respectively. He is now assistant professor in the Faculty of Electrical and Computer engineering at University of Tabriz. His research interests include Hybrid dynamical systems, Stability of systems, Adaptive Control and intelligent systems.



**Sohrab Khanmohammadi** received his B.S. degree in Industrial Engineering from Sharif University, Iran in 1977 and M.Sc. degree in Automatic from University Paul Sabatie, France in 1980 and his Ph.D. degree in Automatic from National University, ENSAE, France in 1983. He is now a professor of Electrical Engineering at University of Tabriz. His research interests are Fuzzy control, Artificial Intelligence applications in Control and Simulation on industrial systems and human behavior. Email: khan@tabrziu.ac.ir