Hindawi Publishing Corporation
Journal of Industrial Engineering
Volume 2013, Article ID 767914, 10 pages
http://dx.doi.org/10.1155/2013/767914

Research Article

Sequential Failure Analysis Using Novel Algorithms in Sequence Determination of Petri Nets Firing

Abolfazl Doostparast Torshizi, Jamshid Parvizian, and Farshad Tooyserkani

Department of Industrial Engineering, Isfahan University of Technology, Isfahan 84156, Iran

Correspondence should be addressed to Abolfazl Doostparast Torshizi; a.doostparast@aut.ac.ir

Received 31 October 2012; Accepted 10 January 2013

Academic Editor: Josefa Mula

Failure occurrence in industrial systems can be a result of a sequence of failures leading to a total system failure. Up to now, several methods to determine failure sequences and to calculate probability of such failures have been proposed. These methods primarily focus on modeling aspects of the problem and do not present a certain framework to determine potential failure sequences. In this paper, a novel approach based on Petri net modeling of the systems is proposed and several heuristic algorithms are developed. Determination of potential failures in sample industrial problems and comparing the results with other existing methods demonstrates that the presented algorithms are much more efficient in dealing with complex Petri net models while existing methods are not capable of handling such complicated models.

1. Introduction

Risk analysis of complicated systems, such as flexible manufacturing cells, is a challenging task. There are diverse approaches aiming in describing different risky behaviors of the systems. One of the most applicable tools in this field is the Fault Tree Analysis (FTA) method. This method, presented in early 1960s, is only a static graphical technique to find correlations among principal reasons of a system failure [1] which makes it difficult in dealing with complicated systems. Other methods, including Failure Mode and Effect Analysis (FMEA), suffer from a similar deficiency [2, 3].

Failures occurring in systems are not confined to failures of each independent sub-system. Sequential failures of subsystems may also lead to the failure of the entire system. Sequential Failure Logic (SFL) was presented by Fussell et al. [4]. In this research, the focus is on analyzing non-repairable electric supply systems with main and standby power units and switch controls. Exact and approximate methods are used to calculate the probability of occurrence of the output event from priority-AND SFL. It is assumed that elementary events are independent and stochastic [4].

The approach proposed in [4] is then adopted by some researchers, for example, in risk analysis of a human-robot system [5], in the field of product liability prevention [6], and quantitative analysis of dynamic systems like space satellites [7].

The concept of sequential failure analysis [1] has been further developed by introducing counters of transitions in stochastic Petri nets (SPNs) located in various network connections [8]. The probabilities of sequential failures are calculated based on the obtained counters of failure transitions in the net.

A fuzzy approach to the problem of sequential failure is presented in [9]. Here, the authors combined adaptability of fuzzy logic with accuracy and modeling power of Petri nets to perform an efficient failure analysis.

Stochastic Petri nets have also been under attention during last years. For example in [10] Wang et al. have used stochastic Petri nets to assess reliability of systems based on non-homogenous Markov isomorphism. Useless service failures are a serious issue in real world problems so Zhao et al. [11] have used stochastic Petri nets models to detect useless service failures.

Uncertainty is an inherent characteristic of industrial systems. Such uncertainties can be handled by stochastic Petri nets. Garg and Sharma [12] have utilized stochastic Petri nets to model the behavior of complex industrial systems and then

on its basis they try to find some of the reliability measure such as mean time between failures (MTBF) using Lambda Tau methodology. Another important property of industrial machines is their availability. Availability is a crucial topic especially in heavy industries since idle times of machines can impose thousands of dollars to the company. According to this, Beirong et al. [13] have used Generalized stochastic Petri nets (GSPN) to model complex industrial systems to maximize the machine availability. In another research, stochastic failure sequence has been investigated by Su and Wang [14] in order to simulate the dynamic reliability of manufacturing systems using stochastic Petri nets.

Although SFL provides an appropriate tool for evaluating systems, it has some drawbacks. For instance, in SFL it is assumed that failure sequences are known. Of course, this cannot be true in real world problems where there may be many unknown sequences of failures. In order to overcome such deficiencies, a novel approach for calculating probabilities of occurrences of sequential failures is presented in [15]. This research adopts the concept of reachability trees in Petri Nets, and then determines different failure sequences by drawing reachability tree of the Petri nets model of the system. Although this approach seems to be suitable for small systems with limited number of states, it is not beneficial for complicated systems with several states since to draw the reachability tree for such systems is nearly impossible. Hence, our goal in this paper is to enhance the method introduced in [15] and develop new algorithms to determine failure sequences of large systems automatically.

The reminder of the paper is as follows. In Section 2, basic concepts of Petri nets and their application in failure analysis are discussed. The framework of sequential failure analysis is presented in Section 3. In Section 4, the developed method is discussed and the performance of the developed algorithms is analyzed. Section 5 is devoted to an illustrative example in order to demonstrate capacities of the developed method. The paper concludes in Section 6.

2. Petri Nets and Their Application in Failure Analysis

Petri Nets are graphical and mathematical modeling tools applicable to many systems. They offer formal graphical description possibilities for modeling systems consisting of concurrent processes. Petri Nets have been used extensively for modeling and analyzing of discrete event systems. As a graphical tool, Petri nets can be used for visual communication aims 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.

For more details about evolution of Petri nets, reader is referred to [16, 17]. A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$, where

$P = \{p_1, p_2, \ldots, p_m\}$,

$T = \{t_1, t_2, \ldots, t_n\}$,

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relations),

$W : F \rightarrow \{1, 2, 3, \ldots\}$ is a weight function,

$M_0 : P \rightarrow \{0, 1, 2, \ldots\}$ is the initial marking,

$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

The dynamic behavior of a system is modeled by changing state or marking in Petri nets according to the following (firing) rules.

(1) A transition t is said to be enabled if each input place p of t is marked with at least $w(p,t)$ tokens, where $w(p,t)$ is the weight of the arc from p to t.

(2) An enabled transition may or may not fire depending on whether or not the event actually takes place (firing conditions are ok).

(3) Firing of an enabled transition t removes $w(p,t)$ tokens from each input place p to t and adds $w(t,p)$ tokens to each output place p of t, where $w(p,t)$ and $w(t,p)$ are the weights of the arcs from p to t or t to p, respectively.

In graphical representation of a Petri net, places are represented by circles and transitions are shown by hollow bars. The relationships between places and transitions are represented by direct arcs. For example, the Petri net of Figure 1 depicts the firing of a transition.

In un-timed Petri net one can prohibit controlled transitions from firing but cannot force the firing of a transition at a particular time. In timed Petri nets controlled transitions are forced to fire by observing the time dependent firing functions. In timed Petri nets, each transition has its specific time which determines the transition's holding time. When a transition is fired during its holding time, markings of networks are not changed. By elapsing holding time, the markings will change according to the firing rules.

Application of Petri nets in failure analysis is an emerging active field of research. The application of PNs is similar to the application of "Fault (Event) Tree Analysis (FTA and ETA)" which are two strong graphical tools for pre (post) event reliability and risk analysis. As this is a rather new field, the literature is not yet rich; however researches on safety analysis and reliability growth [18, 19], reliability evaluation [20–22], and reliability of manufacturing systems [23–25] have already been presented.

Some researchers believe that PNs can be an appropriate alternative for FTA [19, 20], since it not only graphically symbolizes the cause and effect relationships among the events, but also represent dynamic behavior of the system. Fault trees, which are basic graphical risk analysis tools, can be transformed to Petri Nets. For more details, readers are referred to [19].

3. Framework of Sequential Failure Analysis

General framework of Sequential Failure Analysis (SFA) in the literature of reliability and risk analysis is shown in Figure 2 [1].

This methodology utilizes Fault Tree Analysis (FTA) and FMEA and dynamic Petri net modeling for identifying all possible failures and their sequences of occurrence. As shown in Figure 2, the framework of sequential failure analysis

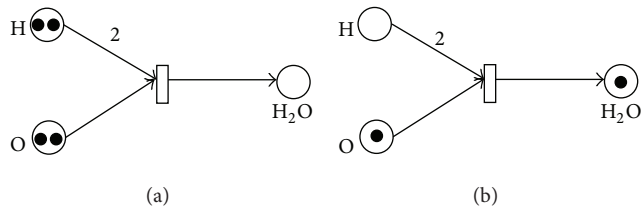

Figure 1: Transition firing: (a) marking before firing, (b) marking after firing.

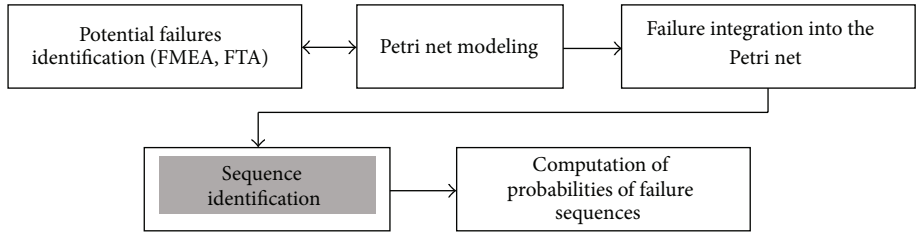

Figure 2: Framework of sequential failure analysis [1].

consists of five steps. Since there is no specific algorithm for sequence identification in the literature, the main goal of this paper is to develop new algorithms in the fourth step, highlighted in Figure 2.

Sequential failure analysis steps start with using FMEA or FTA techniques in order to predict all potential failures. Although FMEA is a general term, it is divided to different branches such as Quality FMEA (QFMEA), Design FMEA (DFMEA), Process FMEA (PFMEA), and so forth. The second step, Petri net modeling, includes modeling the system in a dynamic manner so that all tasks and activities taking place in the system can be seen. The third step is similar to the second one except that it considers system failures and merges such possible failures with the main body of the system Petri model.

Step 4 is our main focus and we will discuss it in next sections. The last step has been considered by many other researchers [1, 15] and we do not discuss it anymore.

4. Methodology

As noted earlier, failures of a system are not limited to failures in sub-systems but they also include a hierarchy of failures in relevant sub-systems. On the other hand, various analyzing methods of Petri nets fail in determining failure sequences leading to total system failure, despite making a schematic view of system behaviors during time. In spite of the capacity of reachability trees in showing sequences of events, they are not efficient in analyzing complicated nets. On the other hand, to the knowledge of authors, no specific algorithm capable of constructing reachability trees, combined with determination of sequences of events, can be found in the literature.

One of the key factors in calculating sequential failures of a system is to determine behavioral sequences of the net leading to the failure. Hence, the proposed algorithm must be able to construct different behavioral states (markings of the net) and the entire sequences of events in a combinatorial manner. In the following, we describe symbols utilized in our methodology and then present our approach.

4.1. Variables and Symbols Definitions

P: Number of all timed and untimed places existing in the Petri net.

T: Number of all timed and untimed transitions existing in the Petri net.

External: An external $T \times P$ matrix. The entries of this matrix are the weights of all arcs connecting each transition (in rows) to each place (in column).

Internal: An internal $P \times T$ matrix. The entries of this matrix are the weights of all arcs connecting each place (in rows) to each transition (in column).

Status: State $T \times P$ matrix. The entries of this matrix are 0 and 1. In fact, this matrix shows how a place (in row) is connected to a transition (in column). If the arc connecting place i to transition j is ordinary, then entry (i, j) of the status matrix is 1; in case of inhibitor arc this component is 0. If there are no arcs between a place and a transition, then the corresponding entry in the status matrix will be again 1.

Info: Evolutionary behavioral matrix of the net. This matrix plays the main role in our heuristic algorithm and it becomes more complete during each step. The entries include markings (behavioral states of the net) and existing firing sequences of the net. We will discuss the structure of this matrix in more details in the following sections.

Level: The last level among different levels of the net being considered.

$M(0)$: Initial marking of the Petri net.

$M(i)$: Marking i of the Petri net.

```
Main Algorithm
Input: Internal = []_{P×T}, External = []_{T×P}, M(0), Status = []_{P×T}
Output: Info matrix                    // Info is defined in Section 4.3.2
External = External*; Info = [0]_{n×n};
Info{1, 1} = M(0); Info{2, 1} = Enabling(M(0), Internal, External);
Stop = 0; level = 1;
while Stop = 0 do
   for g = 2 → 3 → Linefinder(Info)
      Info= Copier(Info, g, level, T);
   end for
   Info = Filler(Info, level, T, Internal, External, Status);
   for s = 1 → Linefinder(Info)
      if Info{s, level} = 0
         Stop = Stop + 1;
      end if
   end for
   if Stop = Linefinder(Info)
      end while
   else Stop = 0; level = level + 1;
   end if
end while
return Info
```

Algorithm 1: Main algorithm.

4.2. Assumptions. Petri nets are:

(i) pure. Purity means that a place cannot be at the same time the input and output of a specific transition,

(ii) live, and

(iii) bounded.

4.3. The Heuristic Algorithm. In this section we present our heuristic algorithm. The following sub-sections describe the *main* algorithm and relevant *functions*.

4.3.1. Main Algorithm. Here we present the main body of the proposed algorithm in Algorithm 1.

4.3.2. The Performance of Algorithm. Algorithm 1 is able to construct reachability tree, entire markings of the systems, and all the firing sequences occurring in the Petri net model, simultaneously. The significance of this algorithm is in analyzing sequential failures where identification of sequential failures is vital in calculating sequences of event leading to total failure of the system.

In this algorithm, firstly a square matrix **Info** with zero elements is constructed. The size of the matrix depends on the size of the Petri net model being considered. In order to avoid confusion when using the matrix **External**, the algorithm uses the transpose of the input matrix **External**. Hence, when speaking about matrix **External**, we mean the transpose of the input matrix **External**. Then, the algorithm substitutes the initial marking $M(0)$ in the cell (1, 1) of **Info** and also substitutes the result of an internal function "Enabling" in cell (2, 1). This internal function gets the marking of a Petri net and gives a row matrix as its output. This row matrix consists of 0 and 1 and demonstrates enabled transitions of the corresponding marking. It is apparent that the number of columns of the output of the function "Enabling" equals T.

As shown in Figure 3, the algorithm is designed to operate level by level. In this paper, levels in reachability tree mean sets of markings in which there are equal numbers of firings from the initial marking to reach such markings.

Then during the next step, the internal algorithm Copier, which will be discussed later, operates on elements of the second row of the matrix **Info** and advances with triple steps. Cells considered by this algorithm are the row matrices demonstrating the enabled transitions of each marking.

Each sequence in matrix **Info** is made of three rows. The algorithm Copier copies three rows of each firing sequence n times. Here n means the number of enabled transitions in each considered sequence.

According to above notations, our proposed algorithm considers three rows for each sequence of firing. The first row shows the markings of each sequence, the second row shows enabled transitions of its corresponding marking, and the third row demonstrates the number of the fired transition in each marking.

Then firing process of the existing transitions in each feasible sequence of the considered level is performed, and new markings, resulted from firing of these transitions, are transported to the next level (column) of the matrix **Info**. This process is performed by internal algorithm Filler which will be explained in the next sections. In order to explain the proposed algorithm, consider the Petri net of Figure 4. Reachability tree and the output matrix **Info** from operation of the algorithm are displayed in Figure 3 and Figure 5, respectively.

In fact, in this algorithm, elements of the matrix **Info** are completed via a wave process (Figure 5). This means

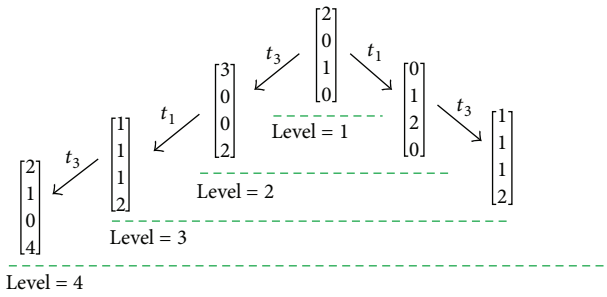

Figure 3: Firing levels form an initial marking in a Petri net.

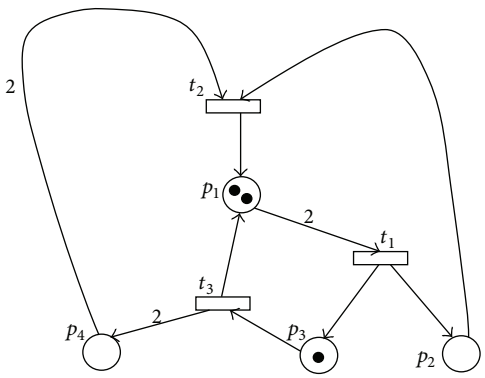

Figure 4: An ordinary Petri net [17].

Level = 1	Level = 2	Level = 3	Level = 4
[2; 0; 1; 0]	[3; 0; 0; 2]	[1; 1; 1; 2]	[2; 1; 0; 4]
[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	
3	1	3	
[2; 0; 1; 0]	[0; 1; 2; 0]		
[0, 0, 0]	[0, 0, 0]		
1	3		

Figure 5: Output of the algorithm with the process of wave operation.

that completion of each level of the reachability tree and evolution of the constructed sequences in the next level are performed simultaneously. It is noteworthy to mention that during evolution of matrix **Info,** elements of rows $2, 5, 8, \ldots$ transform to zero through a multilevel process. The aim is to prevent the algorithm from making repeated markings in the sequence. For example in Figure 5, in the marking of the cell (4, 2) of the matrix **Info**, transition 3 is enabled. If this transition is fired the resulting marking in the fourth level will be equal to the marking of the cell (4, 1) which is iterated. Hence, the sequence terminates at this level.

Evolution of the matrix **Info** terminates only if the new generated marking exists in the existing set of the markings of the sequence. This is to prevent the algorithm from generating repeated sequences. If the new sequence being generated by the algorithm already exists, then the algorithm skips this sequence. This is to prevent the reachability tree of the net from having state space explosion.

We should note that the function Linefinder in Algorithm 1 is an internal function which returns the number of the last row of the matrix **Info** in which there is a positive value. Here we consider the main condition of stop in the main body function and prove its termination.

Termination Condition. According to the assumption of Section 4.2, the proposed algorithm operates only on bounded Petri nets. Internal construction of the function Filler prevents extension of each sequence in case of iteration. Therefore, the possibility of constructing unbounded sequences is zero. On the other hand, since the Petri nets are bounded and algorithm has a performance similar to complete counting process then the entire markings will be definitely counted. This means that according to the principle of the boundedness of the net, there will be one iteration in any of the sequences and all the sequences will be terminated at a level, and the condition for termination condition will be satisfied.

4.3.3. Function Filler. This function is one of the main operators in the main algorithm. This algorithm operates on levels (columns of the matrix **Info**) and fires enabled transitions in such a way that no iteration happens. It also constructs next level in matrix **Info** gradually by a wave shaped motion. Algorithm 2 presents this function.

This function returns a two-dimensional row matrix called "a" and a one-dimensional row matrix, "b". In fact, this function operates on the considered level with triple steps and gives the kinds and numbers of different markings in that level in matrixes "a" and "b", respectively.

Algorithm "Filler" has no termination condition and operates according to the dimension of its input data. This algorithm performs on the first row of each sequence and then analyzes each sequence. In case of existence of enabled transition in the last marking of the sequence, it generates an experimental marking. If this new marking is not iterated then the function adds this new marking to the end of the sequence in the next level and also adds the enabled transitions of the new marking to the next level. In Algorithm 2, we have used a function called "Newmarking" which will be discussed in detail in the coming sections.

4.3.4. Function Copier. This algorithm plays an important role in generating different sequences, by operating on the second row of each existing sequence. This row demonstrates enabled transitions of its corresponding marking. According to the number of enabled transitions in each marking, function "Copier" firstly checks whether firing a transition leads to a new sequence or not. If this is true then the relevant

```
Filler Algorithm
Input: Info, level, T, Internal, External, Status
Ouput: Cons = []_{n×n}
Cons = Info;
[ab] = Finder(Info, level);                    \\ a = []_{1×k}, b = []_{1×k}
for i = 1 → 3 → Linefinder(Info) + 1
   if Cons{i − 2, level} ≠ 0 then
      for m = 1 → k
         if Info{i − 2, level} − a{1, m} = 0 then
            for j = 1 → T
               if Cons{i − 1, level}(j) = 1 then
                  Cons{i − 1, level}(j) = 0; Cons{i, level} = j;
               end if
                  for z = 2 → 3 → Linefinder(Info)
                     Update Info;
                  end for
               for l = 1 →level
                  If Newmarking(Cons{i − 2, level}) is not iterated then
                     Add Newmarking and enabling matrices to the sequence;
                  end if
               end for
            end for
         end if
      end for
   end if
end for
return Cons
```

Algorithm 2: Function Filler of the proposed algorithm.

```
Copier Algorithm
Input: Info, row, level, T
Output: the same Info matrix which is modified
ifInfo{row−1, level} ≠ 0 then
dd = number of ones in Info{row, level};
   if level = 1 then
      for k = 1 → dd − 1
         for l = 1 →level
            Copy the sequence to Info with one row spacing;
         end for
      end for
   else
         for k = 1 → dd − 1
            for l = 1 →level
               Copy the sequence to Info;
            end for
         end for
   end if
end if
return Info
```

Algorithm 3: Function copier.

sequence will be added to the end of the evolutionary matrix "Info". Output of this function is constantly processed by the function "Filler". The corresponding algorithm of this function is presented in Algorithm 3.

4.3.5. Function Newmarking. This algorithm is designed based on dominating concepts in the field of Petri nets. This function constructs and solves linear systems of the Petri net using basic concepts of token transfer and so forth. For more

```
Newmarking Algorithm
Input: Marking =[]_{P×1}, Internal, External, Status, Tranum
Output: Outmarking (a new marking resulting from firing transition Tranum)
Difference = [0]_{P×T};
For k = 1 → T
   for i = 1 → P
      if Status(i, k) = 1 then
         Difference = External − Internal;
      else
         Difference = External;
      end if
   end for
end for
Outmarking = Marking + Difference ×[0]_{T×1} (with element 1 in rownumber Tranum);
return Outmarking
```

Algorithm 4: Function newmarking.

details, the reader is referred to [17]. The algorithm of this function is demonstrated in Algorithm 4.

At the end of this section, the proposed method is entirely represented in Figure 6.

5. An Illustrative Example

In this section we solve a sequential failure problem and demonstrate the capabilities of the proposed method. This example is adopted from [10]. We have coded the proposed algorithms in MATLAB. Consider the Petri net of a machining cell in Figure 7. Input matrices of the proposed heuristic algorithm are

$$\text{initial marking} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$\text{internal matrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\text{external matrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

According to the proposed main algorithm represented in Figure 6, the wavy procedure is applied to the manufacturing cell represented in Figure 7. This manufacturing cell consists of one robotic arm, and a single machine to process the incoming parts. The robotic arm is responsible for loading and unloading the parts to and from the machine. In cases when the robotic arm drops a part then an operator should enter the hazardous zone to solve the problem and load or unload the machine manually. In such situation the operator is in danger of having accident with the robotic arm. This process is totally depicted in Figure 7.

Computational results of executing the heuristic algorithm are presented in Table 1. As it can be seen from Figure 7, the aim is to determine all the firing sequences of transitions leading to firing of transition t_7.

Here we will describe the solution procedure of this problem step by step. According to the flowchart represented in Figure 6 firstly the internal, external, and status matrices should be determined according to Petri net model of the system. Then in the next step the first two elements of the matrix **Info** should be filled using the Enabling function. Then for each firing level, enabled transition are discovered and firing sequences for the entire enabled transitions are performed until reaching a similar firing sequence which has been obtained before.

The functions Linefinder, Filler, and Copier are intermediate algorithms which are responsible for checking uniqueness of a firing sequence, and implementing the wavy procedure to fill the **Info** matrix. As noted before, the wavy procedure tries to find the possible unique firing sequences and find firing levels, simultaneously.

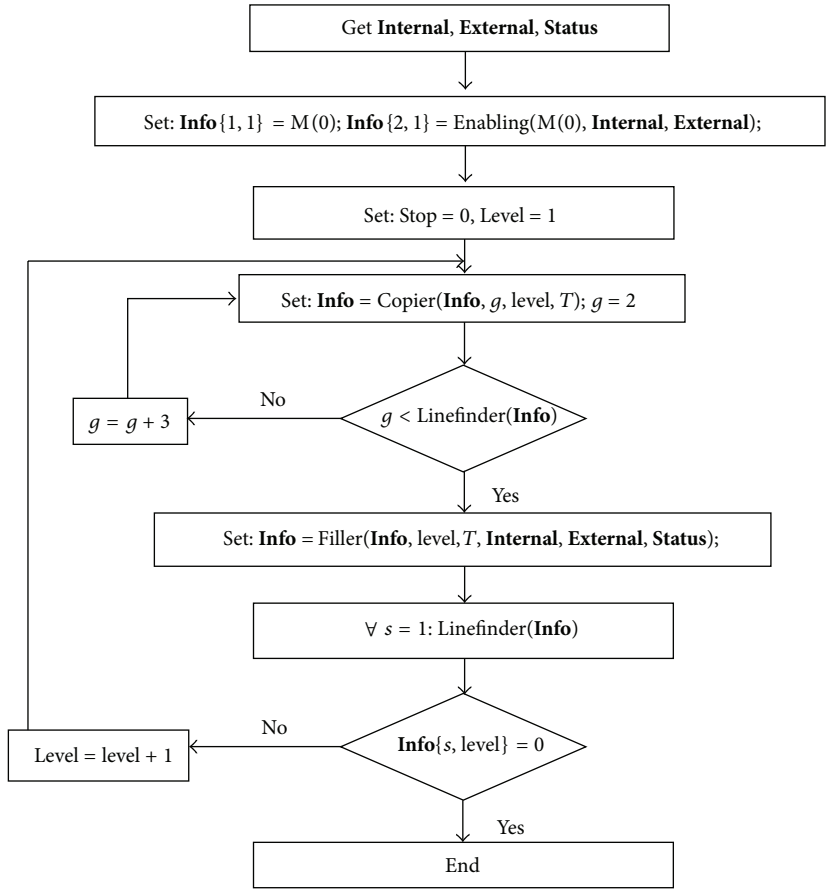

Figure 6: Flowchart of the main algorithm.

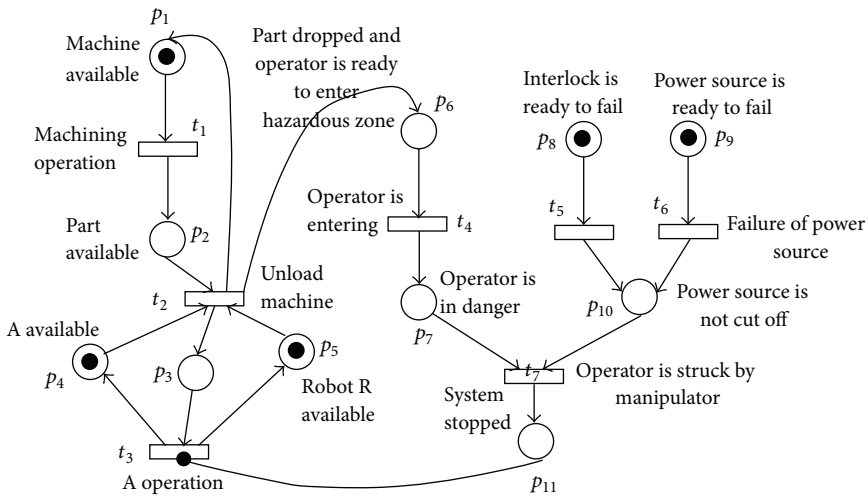

Figure 7: Petri net model of a manufacturing cell.

The main body algorithm in this paper then terminates until the entire possible firing sequences for the entire firing levels have been investigated and then they are reported.

Transition 7 plays the role of hitting an operator by robotic manipulator. Since the number of firing sequences leading to firing t_7 is large we present only some of firing sequences leading to firing failure transition (t_7).

Table 1: Firing sequences leading to firing transition t_7 of Figure 4.

1	$t_5t_1t_2t_1t_3t_2t_4t_7$
2	$t_5t_1t_2t_4t_7$
3	$t_5t_1t_2t_4t_1t_7$
4	$t_1t_2t_5t_4t_7$
5	$t_1t_2t_5t_1t_4t_7$
6	$t_1t_2t_1t_4t_5t_7$
7	$t_1t_2t_4t_5t_7$
8	$t_1t_2t_4t_1t_5t_7$
9	$t_1t_2t_3t_4t_1t_6t_7$
10	$t_1t_2t_1t_4t_5t_3t_7$
11	$t_1t_2t_1t_5t_4t_6t_3t_7$
12	$t_6t_1t_2t_4t_1t_5t_7$
13	$t_1t_2t_3t_1t_4t_2t_1t_5t_6t_7$
14	$t_1t_2t_6t_3t_1t_4t_2t_5t_7$
15	$t_1t_2t_1t_3t_2t_1t_5t_4t_4t_3t_7$
16	$t_5t_1t_2t_1t_3t_2t_4t_3t_1t_7$
17	$t_5t_1t_2t_1t_3t_2t_6t_3t_1t_4t_7$
18	$t_1t_2t_1t_3t_2t_3t_2t_3t_4t_4t_1t_6t_7$

Our proposed algorithm is general and can handle different firing sequences in addition to failure analysis. Table 2 represents computational results of the technique presented in [15]. According to Table 2, this method which is based upon drawing reachability tree of the Petri nets can detect only 8 failure sequences while the proposed method in this paper has detected 18 failure sequences led to firing of t_7. This proves that the older technique can detect only 40% of potential sequential failures but the proposed heuristic algorithms in this paper are capable to approximately detect the whole sequences. Hence, performance of the method [15] cannot be trusted in complex systems.

By analyzing the results of the two tables above, it can be concluded that the maximum number of transition firings detected by the method [15] is 7 firings but according to Table 1 maximum number of firings is 12 which is considerably greater than of [15]. This is because the older method is a graphical-based method and cannot handle complex nets. On the other hand, the technique presented in this paper represents a systematic approach and does not need to draw reachability graphs of the net and has omitted some time consuming parts of the older method.

Computational results of the above example shows that the method used in [15] just considers some of firing sequences leading to failure while method adopted in this paper is much stronger and can determine all the firing sequences.

6. Conclusions

In this paper, some novel algorithms in order to determine firing sequences leading to failures in systems were developed. The proposed method not only can present entire firing sequences in a Petri net but also it can draw reachability tree of that Petri net, simultaneously. We coded these algorithms MATLAB programming language and compared the results with one of the main existing methods in the literature. This comparison demonstrated precision and accuracy of the proposed method.

Table 2: Firing sequences leading to firing transition t_7 of Figure 4 by the method presented in [15].

1	$t_5t_1t_2t_4t_7$
2	$t_5t_1t_2t_4t_1t_7$
3	$t_1t_2t_5t_4t_7$
4	$t_1t_2t_5t_1t_4t_7$
5	$t_5t_1t_2t_1t_3t_2t_4t_7$
6	$t_1t_2t_1t_4t_5t_7$
7	$t_1t_2t_4t_5t_7$
8	$t_1t_2t_4t_1t_5t_7$

References

[1] A. Adamyan and D. He, "Sequential failure analysis using counters of Petri net models," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 33, no. 1, pp. 1–11, 2003.

[2] M. Braglia, M. Frosolini, and R. Montanari, "Fuzzy critically assessment for failure modes and effect analysis," *International Journal of Quality & Reliability Management*, vol. 20, no. 4, pp. 503–524, 2003.

[3] K. Xu, L. C. Tang, M. Xie, and M. L. Zhu, "Fuzzy assessment of FMEA for engine systems," *Reliability Engineering & System Safety*, vol. 75, no. 1, pp. 19–27, 2002.

[4] J. B. Fussell, E. F. Aber, and R. G. Rahl, "On the quantity analysis of priority-AND failure logic," *IEEE Transactions on Reliability*, vol. R-25, no. 5, pp. 324–326, 1976.

[5] Y. Sato, E. J. Henley, and K. Inoue, "Action-chain model for the design of hazard-control systems for robots," *IEEE Transactions on Reliability*, vol. 39, no. 2, pp. 151–157, 1990.

[6] Y. Shibata and Y. Sato, "Case study of risk assessment for product liability prevention," in *Proceedings of the PSAM-4*, vol. 2, pp. 1215–1220, 1998.

[7] L. Ngom, A. Cabarbaye, and C. Barpm, "Taking into account of dependency relations in the Monte Carlo simulation of non-coherent fault trees," in *Proceedings of the PSAM-4*, vol. 3, pp. 2067–2072, 1998.

[8] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat, *Synchronization and Linearity*, John Wiley, New York, NY, USA, 1992.

[9] D. Torshizi A and S. R. Hejazi, "A fuzzy approach to sequential failure analysis using Petri nets," *International Journal of Industrial Engineering and Production Research*, vol. 21, no. 2, pp. 53–60, 2010.

[10] Q. Wang, J. Gao, K. Chen, and P. Yang, "Reliability assessment of Manufacturing system based on HSPN models and non-homogeneous isomorphism Markov," in *Proceedings of the International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, pp. 182–186, 2011.

[11] M. Zhao, Y. Zhou, Y. Yang, W. Song, and Y. Du, "A new method to detect useless service failure model in SPN," *Journal of Convergence Information Technology*, vol. 5, no. 3, pp. 129–134, 2010.

[12] H. Garg and S. P. Sharma, "Stochastic behavior analysis of complex repairable industrial systems utilizing uncertain data," *ISA Transactions*, vol. 51, no. 6, pp. 752–762, 2012.

[13] Z. Beirong, X. Xiaowen, and X. Wei, "Availability modeling and analysis of equipment based on generalized stochastic petri nets,," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, no. 21, pp. 4362–4366, 2012.

[14] C. Su and S. Wang, "Dynamic reliability simulation for manufacturing system based on stochastic failure sequence analysis," *Journal of Mechanical Engineering*, vol. 47, no. 24, pp. 165–170, 2011.

[15] A. Adamyan and D. He, "Analysis of sequential failures for assessment of reliability and safety of manufacturing systems," *Reliability Engineering and System Safety*, vol. 76, no. 3, pp. 227–236, 2002.

[16] C. A. Petri, "Kommunikation mit automaten," Schriften Des IIM no. 3, Institut für Instrumentelle Mathematik, Bonn, Germany, 1962.

[17] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[18] N. G. Leveson and J. L. Stolzy, "Safety analysis using Petri nets," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 3, pp. 386–397, 1987.

[19] S. K. Yang and T. S. Liu, "Failure analysis for an airbag inflator by petri nets," *Quality and Reliability Engineering International*, vol. 13, no. 3, pp. 139–151, 1997.

[20] T. S. Liu and S. B. Chiou, "The application of Petri nets to failure analysis," *Reliability Engineering and System Safety*, vol. 57, no. 2, pp. 129–142, 1997.

[21] G. S. Hura and J. W. Atwood, "Use of Petri nets to analyze coherent fault trees," *IEEE Transactions on Reliability*, vol. 37, no. 5, pp. 469–474, 1988.

[22] V. Kumar and K. K. Aggarwal, "Petri Net modelling and reliability evaluation of distributed processing systems," *Reliability Engineering and System Safety*, vol. 41, no. 2, pp. 167–176, 1993.

[23] J. Changjun, D. Baiqing, and W. Feng, "Study on reliability of manufacturing system based on petri net," *High Technology Letters*, vol. 1, no. 2, pp. 25–30, 1995.

[24] H. Xiong and Y. He, "GSPN based reliability modeling and analysis of CIMS," *Mechanical Science Technology*, vol. 16, pp. 1103–1106, 1997.

[25] C. H. Kuo and H. P. Huang, "Failure modeling and process monitoring for flexible manufacturing systems using colored timed petri nets," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 3, pp. 301–312, 2000.