Improving the Modelling Capability of an Integrated Fault Detection and Diagnostic Petri Net Methodology for Dynamic Systems

Taofeeq Alabi Badmus*, Darren Prescott** and Rasa Remenyte-Prescott***

**Resilience Engineering Research Group, University of Nottingham, Nottingham, UK & Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria; email address: taofeeq.badmus@nottingham.ac.uk & taofeeq.badmus@fuoye.edu.ng*

***Resilience Engineering Research Group, University of Nottingham, Nottingham, UK; email address: darren.prescott@nottingham.ac.uk*

****Resilience Engineering Research Group, University of Nottingham, Nottingham, UK; email address: r.remenyte-prescott@nottingham.ac.uk*

Abstract.
The existing Generalised Stochastic Petri Net and modified Bayesian Stochastic Petri Net (*GSPN-mBSPN*) methodology has demonstrated improved modelling capabilities for fault diagnosis in dynamic systems with feedback control loops. However, the *GSPN-mBSPN* approach uses predefined input conditional probability tables (*iCPTs*) for fault diagnosis, limiting its usage in dynamic fault diagnosis due to the time required to define and populate the *iCPT* entries accurately. This paper presents an algorithm to automatically generate *iCPT* tables, enhancing the modelling capability of the *GSPN-mBSPN* approach for fault diagnosis of dynamic systems under time-varying conditions. The *GSPN* module in a *GSPN-mBSPN* model of a dynamic system is analysed and structured into sub-net modules, representing system components, monitoring parameters, and interactions. These sub-net modules provide data structures for the *iCPT* tables, describing the working and failure states/modes of system components, states of the monitoring parameter, and causal relationships between system component states and observable process parameters. The algorithm populates the entries of the *iCPT* tables based on the analysis of the sub-net modules. Application of the algorithm to a water tank level control system demonstrates improved speed and accuracy in generating *iCPTs* for dynamic fault detection and diagnosis applications using *GSPN-mBSPN* approach.

1. Introduction

Numerous approaches have been documented in the literature for assessing the reliability of complex dynamic systems and diagnosing faults (Liu and Zio, 2017; Shukla and Arul, 2020; Vasilyev *et al.*, 2021; Xu *et al.*, 2023). One promising research direction involves integrating Bayesian Network features into a Petri Net model. These approaches offer advantages such as low parameter settings, well-structured interdependencies between system components, and the ability to handle time-varying data (Vagnoli and Remenyte-Prescott, 2022), crucial for developing reliable and effective fault diagnostic methods. However, Bayesian-supported Petri Net methods have limitations. For instance, as the number of parent nodes and dependency levels increases, the computational complexity of the equations describing the interdependencies in a Bayesian Stochastic Petri Net model also grows (Taleb-Berrouane, Khan and Amyotte, 2020).

Furthermore, the existing Generalised Stochastic Petri Net and modified Bayesian Stochastic Petri Net (*GSPN-mBSPN*) methodology for fault detection and diagnosis relies on predefined input conditional probability tables (*iCPTs*) for its fault diagnostic process. This reliance hinders its effectiveness in diagnosing faults in dynamic operating systems under time-varying conditions. Moreover, populating the entries of the *iCPT* tables can be time-consuming and error-prone (Forner, Kumar and Kinshuk, 2013), especially with a vast amount of data (Zhou *et al.*, 2018; Kabir and Papadopoulos, 2019). To overcome these challenges, this paper proposes a method that automatically generates the structure of the *iCPT* tables used in the fault diagnostic module of a *GSPN-mBSPN* model. The proposed method dynamically updates the entries of the tables through the analysis of the *GSPN* module. By doing so, it aims to enhance the overall modeling capabilities of the *GSPN-mBSPN* methodology, particularly for fast, accurate, and time-varying fault detection and diagnostic applications.

2. Overview of the Existing GSPN-mBSPN Approach

A formal definition of the existing *GSPN-mBSPN* methodology is first presented here to provide some necessary background knowledge.

Definition 1. A *GSPN-mBSPN* method is defined as $GSPN - mBSPN = \{GSPN, P_{TP}, A_{TA}, P_{CORP}, P_{COP}, T_{CRT}, InferenceCode, A_{RA}, mBSPN, A_{COA}, A_{CORA}\}$ such that:

1. $GSPN$ is the conventional Generalised Stochastic Petri Net formalism as defined in (Nourredine *et al.*, 2023).
2. P_{TP} is a set of test places (Chen *et al.*, 2010) that enhance the modeling efficiency of Petri net models, particularly in complex systems.
3. A_{TA} is a set of test arcs.
4. P_{CORP} denotes a set of conditional output reset places, which are a specialised type of reset places (Andrews, 2013). These places serve as interfaces for transmitting observed evidence from the system's behavioral GSPN module to its fault diagnostic *mBSPN* module.
5. P_{COP} signifies a set of *conditional output/trigger places*. These places trigger the fault diagnostic module when an abnormality is detected during system operation.
6. T_{CRT} represents a set of conditional reset transitions, which are a special type of reset transitions. T_{CRT} is utilised to model fault detection when the system operating state deviates from its expected normal conditions.
7. $InferenceCode \rightarrow \{0, 1, 2\}$, denotes the codes for Bayesian inference sampling algorithms: 0 for forward sampling, 1 for rejection sampling, and 2 for likelihood weighting sampling (Russell and Norvig, 2010).
8. A_{RA} is a set of reset arcs, where $a_{RA}(p_{CORP}, t_{CRT})$ represents the weight of the reset arc from the conditional output reset place p_{CORP} to the conditional reset transition t_{CRT}.
9. $mBSPN$ is a fault diagnostic Petri net module of the integrated *GSPN-mBSPN* method.
10. A_{COA}: a set of conditional output arcs between T_{CRT} and P_{CORP} or between T_{CRT} and P_{COP} such that
    a. $$a_{COA}(t_{CRT}, p_{CORP}) = \begin{cases} 0 & if\ Inference\ algorithm = forward\ sampling \\ \{i+1 |\ i < |P^d_{CPP}(t^d_{CPT})|\} & Otherwise \end{cases},$$
    where i is the index of the observed conditional probabilistic place $p_{CPP} \in P^d_{CPP}$ in the vector of the conditional probabilistic places of transition t^d_{CPT} (i.e., $P^d_{CPP}(t^d_{CPT})$) such that $p_{CORP} \rightarrow e_{EP}(t^d_{CPT})$ where $e_{EP}(t^d_{CPT})$ is the evidence place of the conditional probabilistic transition t^d_{CPT}.
    b. $$a_{COA}(t_{CRT}, p_{COP}) = \begin{cases} 1 & if\ m(p_{COP}) = 0 \\ 0 & Otherwise \end{cases}$$
    where $m(p_{COP})$ is the current marking of the conditional output/trigger place p_{COP}.

3. The Proposed Improvement on the *GSPN-mBSPN* Approach

The existing *GSPN-mBSPN* methodology requires a manual definition of input conditional probability tables (*iCPT*) for *CPT* transitions in text files. The *GSPN-mBSPN* model of a dynamic system is also described in text files as a set of modelling elements. These files are then input into a custom program to represent and simulate the model. However, the manual population of *iCPT* entries is time-consuming and error-prone. To address this, an algorithm is developed to automatically generate *iCPTs* for *CPT* transitions using the *GSPN* module of the *GSPN-mBSPN* model. The algorithm, as described by the flowchart and pseudocode in Figure 1 and Algorithm 1, respectively, involves analysing and structuring the places and transitions of the *GSPN* module into sub-net modules representing system components and interconnections. Sub-net modules are created based on the type of places, transitions, and the arcs connecting them. Timed transitions with input and output places of type "*component*" form sub-net modules for system components. Immediate transitions with common input and output places of type "*component*" and test/inhibitor places of type "*normal*" form sub-net modules for monitoring parameters. Interconnection sub-net modules are formed by immediate transitions with common input and output places of type "component" and test/inhibitor places equivalent to lower-level sub-net modules' input and output places. A sub-net module's input and output places represent the states of a component, monitoring parameter, or propagated variable in the *GSPN* module. The created sub-net modules are stored in a vector and analysed by the *Create_iCPT()* function in Algorithm 1 to generate the *iCPTs* automatically. These sub-net modules and their corresponding *iCPTs* can be used to develop an *mBSPN* module in the *GSPN-mBSPN* model. Each sub-net module in the *GSPN* module corresponds to a conditional

probabilistic transition t_{CPT}^{d} in the *mBSPN* module. The input and evidence places of a t_{CPT}^{d} of a monitoring parameter are port places linked to a conditional output place and a conditional output reset place in the fault detection module of the *GSPN-mBSPN* model. The output places of the transition are conditional probabilistic places created based on the number of input and output places in the corresponding sub-net module.

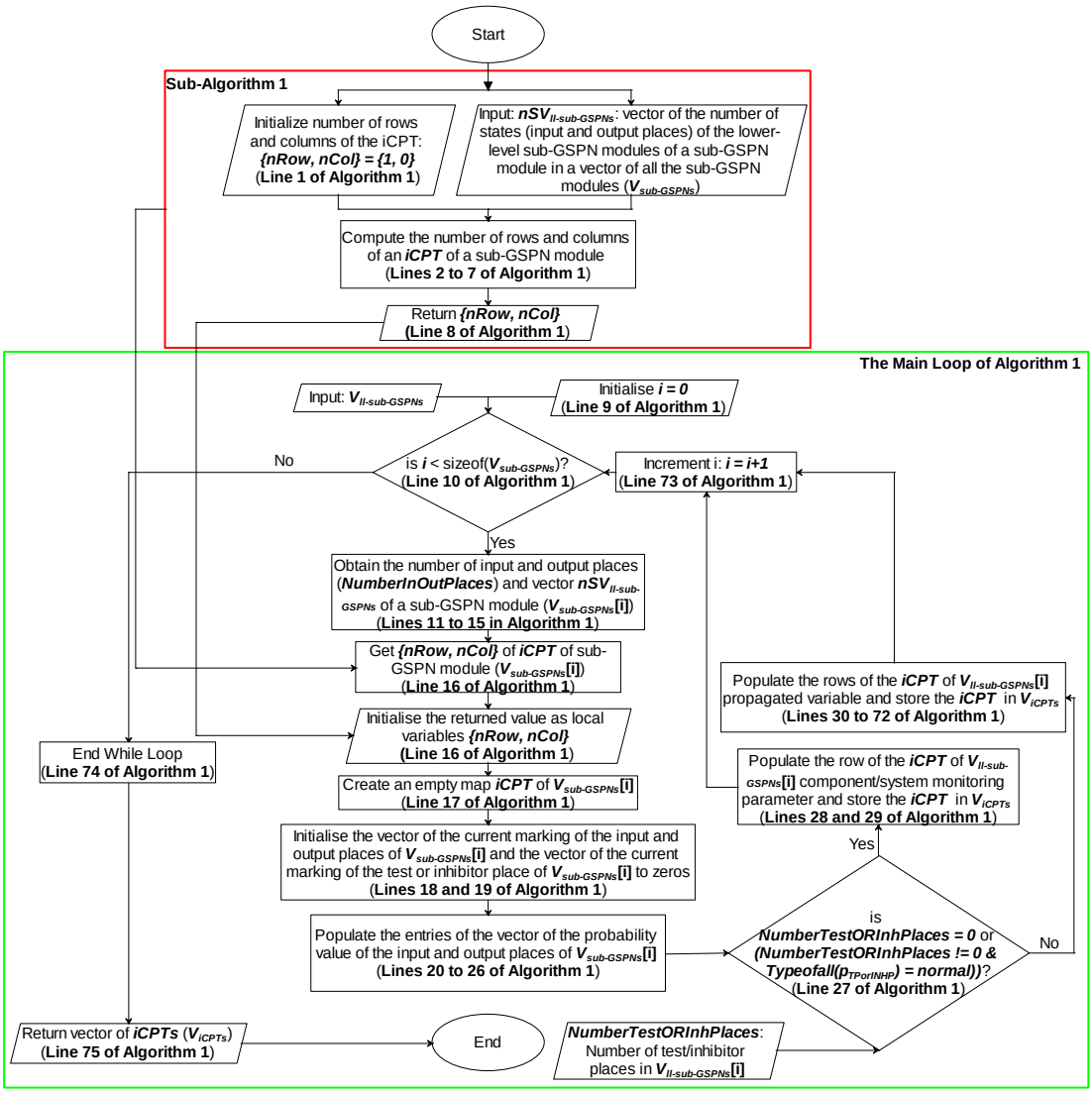


Figure 1. Flowchart for Automatic Generation of iCPT Tables of CPT Transitions

Algorithm 1: Pseudo-code for Automatic Generation of *iCPTs* for *CPT* Transitions in *GSPN-mBSPN*

Input: $V_{sub-GSPNs}$ is a vector of the sub-GSPN modules in a *GSPN-mBSPN* model, $nSV_{ll-sub-GSPNs}$ is a vector storing the number of lower-level sub-GSPN ($ll-sub-GSPN$) module(s) states (input and output places) interconnected to a higher-level sub-GSPN module via test/inhibitor places.
Output: Generated vector of *iCPTs* for *CPT* transitions (that is V_{iCPTs})

Sub-Algorithm1: Function *Compute_Number_iCPT_Rows_Columns($nSV_{ll-sub-GSPNs}$) Pseudo-code*

1. $\{nRow, nCol\} \leftarrow \{1,0\}$
2. *Initialise* loop counter variable $i \leftarrow 0$
3. **while** $i < sizeof(nSV_{ll-sub-GSPNs})$ **do**:
4. $nRow \leftarrow nRow * nSV_{ll-sub-GSPNs}[i]$
5. $nCol \leftarrow nCol + nSV_{ll-sub-GSPNs}[i]$
6. $i \leftarrow i + 1$
7. **endwhile**
8. **Return** $\{nRow, nCol\}$

The Main Algorithm: Function *Create_iCPT($V_{sub-GSPNs}$)*

9. *Initialise* vector index variable $i \leftarrow 0$

10. **while** $i < sizeof(V_{sub-GSPNs})$ **do**
11. $NumberInOutPlaces =$ *number of input and output places* **in** $V_{sub-GSPNs}[i]$
12. **if** $(NumberInOutPlaces < 2)$ **then**
13. $NumberInOutPlaces \leftarrow 2$
14. **endif**
15. $nSV_{ll-sub-GSPNs} \leftarrow V_{sub-GSPNs}[i]$ *// Get the vector* $nSV_{ll-sub-GSPNs}$ *of* $V_{sub-GSPNs}[i]$
16. *initialise* $\{nRow, nCol\} =$ **call** *Compute_Number_iCPT_Rows_Columns(*$nSV_{ll-sub-GSPNs}$*)* //**Sub-**//**Algorithm1**
17. **create** $iCPT$ *//create an empty map for storing the entries of the iCPT of* $V_{sub-GSPNs}[i]$
18. **initialise** $CurrentTestORInhPlacesMarking[nCol]$ **with** zeros
19. **initialise** $ProbVal[NumberInOutPlaces]$ **with** zeros
20. **get** $CurrentInOutPlacesMarking$ **from** $V_{sub-GSPNs}[i]$
21. **if** $(CurrentInOutPlacesMarking = ProbVal[NumberInOutPlaces])$ **then**
22. **clear** $ProbVal[NumberInOutPlaces]$
23. **populate** $ProbVal[NumberInOutPlaces]$ **with** $\frac{1}{NumberInOutPlaces}$
24. **else**
25. $ProbVal[NumberInOutPlaces] \leftarrow CurrentInOutPlacesMarking$
26. **endif**
27. **if** $\left(\begin{array}{c} NumberTestORInhPlaces \text{ in } V_{sub-GSPNs}[i] = 0 \text{ or} \\ \left(NumberTestORInhPlaces \neq 0 \text{ and } Typeofall(p_{TPorINHP}) \text{ in } V_{sub-GSPNs}[i] = \text{normal}\right) \end{array}\right)$ **then**
28. **insert** $\{CurrentTestORInhPlacesMarking, ProbVal\}$ **in** $iCPT$
29. **store** $iCPT$ **in** V_{iCPTs} ***// Storing iCPT of a system monitoring parameter or component***
30. **else** ***// This is an interconnection sub-GSPN module***
31. **create** $SubModulesMarkingsCombination$ *//Creation of a 3D vector to store a 2D vector //holding all the possible combinations of markings of test/inhibitor places in* $V_{sub-GSPNs}[i]$
32. **initialise** $nCol \leftarrow 0$ // initialisation of $nCol$ of $V_{sub-GSPNs}[i]$ to zero
33. **for each** $ll - sub - GSPN$ **of** $V_{sub-GSPNs}[i]$ **do**
34. **create** $TestORInhPlacesMarkingsCombination$ *//Creation of a 2D vector to store the //possible combination of the markings of the test/inhibitor places in* $V_{sub-GSPNs}[i]$
// Get the total number of states of the current $ll - sub - GSPN$
35. **get** $nSofllsubGSPN =$ *number of states of* $ll - sub - GSPN$ **in** vector $nSV_{ll-sub-GSPNs}$
36. $nCol \leftarrow nCol + nSofllsubGSPN$ *// increment* $nCol$ *by* $nSofllsubGSPN$
// Generate possible marking permutation of the input and output places of $ll - sub - GSPN$
37. **Initialise** loop counter variable $j \leftarrow 0$
38. **while** $j < nSofllsubGSPN$ **do**
39. **initialise** $tempVec[nSofllsubGSPN]$ **with** zeros *//Entries of a temporary vector for storing //the marking combinations of the input and output places of* $ll - sub - GSPN$ *are set to 0s*
40. **update** $tempVec[j] \leftarrow tempVec[j] + 1$
41. **store** $tempVec[j]$ **in** $TestORInhPlacesMarkingsCombination$
42. **get** $M'(p)$ **in** $inoutpV_{ll-sub-GSPN}[j]$ //$M'(p)$ is the current marking of a place p at the //location j in the vector of the input and output places of $ll - sub - GSPN$ ($V_{ll-sub-GSPN}^{INPOUTP}$)
43. **compute** $index \leftarrow j + nCol - nSofllsubGSPN$ *//Compute index of p to be updated*
//Update $CurrentTestORInhPlacesMarking$ *of* $V_{sub-GSPNs}[i]$ *value at index with* $M'(p)$
44. **update** $CurrentTestORInhPlacesMarking[index] \leftarrow M'(p)$ **in** $V_{ll-sub-GSPN}^{INPOUTP}[j]$
45. $j \leftarrow j + 1$
46. **endwhile**
47. **store** $TestORInhPlacesMarkingsCombination$ **in** $SubModulesMarkingsCombination$
48. **endforeach**
49. **create** vector **iterator** $SubModulesMarkingsCombinationIter$ *//The creation of a 3D vector //iterator to store the 2D vector iterators of the entries in* $SubModulesMarkingsCombination$
50. **for each 2D vector in** $SubModulesMarkingsCombination$ **do**
51. **create** $TestORInhPlacesMarkingsiter$ *//Creation of a 2D vector iterator to the beginning //of each 2D vector in* $SubModulesMarkingsCombination$
52. **store** $TestORInhPlacesMarkingsiter$ **in** $MarkingsCombinationIter$
53. **endforeach**

// Cascading content of the $MarkingsCombinationIter$ vector to obtain all the possible
// marking combinations. First, initialise size of the $MarkingsCombinationIter$ vector to size
initialise $size \leftarrow sizeof(MarkingsCombinationIter)$
While $MarkingsCombinationIter[0] \neq endof(SubModulesMarkingsCombination[0])$ **do**
// Get the current marking combination data vector from $MarkingsCombinationIter$
$M =$ **get** *row vector* **from** $MarkingsCombinationIter$
// Insert a new row into the iCPT of $V_{sub-GSPNs}[i]$
if $(CurrentTestORInhPlacesMarking = M)$ **then**
insert $\{CurrentTestORInhPlacesMarking, ProbVal\}$ **in** $iCPT$
else
insert $\{M, ProbVal\}$ **in** $iCPT$
endif
//increment the iterator to determine the next possible marking combination
$++MarkingsCombinationIter[size-1]$
//Determine when the new cascade of marking combinations begins
initialise $k \leftarrow size - 1$
while $(k > 0$ **and** $MarkingsCombinationIter[k] ==$
$endof(SubModulesMarkingsCombination[k]$ **do**
initialise $MarkingsCombinationIter[k] \leftarrow$
$startof(SubModulesMarkingsCombination[k]$
$++MarkingsCombinationIter[k]$
$k \leftarrow k - 1$
endwhile
endwhile
endif
increment $i \leftarrow i + 1$
endwhile
Return V_{iCPTs}

4. Application of the Improved GSPN-mBSPN Methodology

In this section, we consider a water tank level control system depicted in Figure 2 to demonstrate the effectiveness of the proposed algorithm for generating conditional probability tables for a GSPN-mBSN model of a dynamic system. The system includes three valves (V1, V2, and V3) with three states (working, failed open, failed close), two level sensors (S1 and S2) with three states (working, failed high, failed low), two controllers (C1 and C2) with three states (working, failed high, failed low), six pipelines (P1 to P6) with two states (not blocked, blocked), one overspill tray with a monitoring sensor SP1 (water present, no water present), and three flow sensors (VF1, VF2, and VF3) for monitoring flow in and out of the system. Further details about the water tank level control system can be found in the published article by (Hurdle, Bartlett and Andrews, 2009).

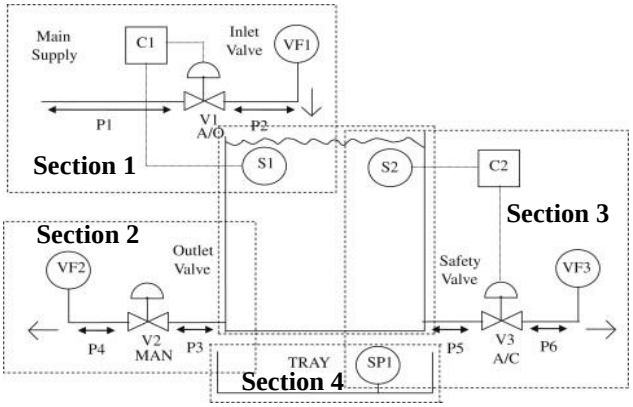
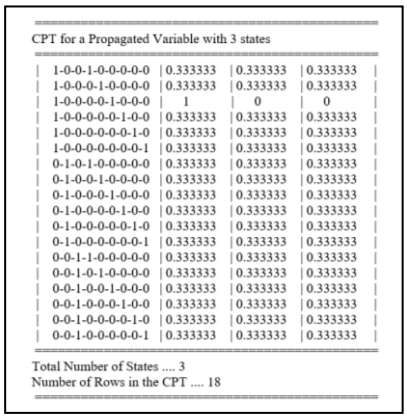


Figure 2. Schematic Diagram of the Water Tank Level Control System

The *GSPN-mBSPN* model of the water tank system consists of 254 places, 161 transitions, and 962 arcs. Analysis of the *GSPN* module of the *GSPN-mBSPN* model by the developed bespoke C++ program produces twenty interconnections and sixteen component sub-net modules, including the system monitoring parameter sub-net module. The program analyses these modules using the algorithm presented in Section 3 to

automatically generate thirty-six input conditional probability tables (*iCPTs*) corresponding to these modules, with a total of 1780 *iCPT* entries. The *iCPTs* are populated dynamically based on the current state of the components and interconnected variables in the system. Examples of automatically generated input conditional probability tables for sample components, propagated variables, and monitoring variables are shown in Figure 3. Mean creation times, confidence intervals, and margin of errors (half-width) for different categories of *iCPTs* in the *GSPN-mBSPN* model are presented in Table 1. These values were obtained after 50 simulations using the *iCPT* auto-generation algorithm implemented in the bespoke C++ program. Simulations were performed on a Windows 10 64-bit Intel(R) Core (TM) i3 system with a 3.60 GHz processor and 8.00 GB of RAM. The mean creation time of the *iCPT* for pipe P1 state initially fluctuates but gradually stabilizes after 25 simulations, as shown in Figure 4. Similar trends were observed for the other generated *iCPTs*. Running 50 simulations ensures the accuracy of the results presented in Table 1. The mean creation time of an *iCPT* for a propagated sub-net module in the water tank system increases as the number of interconnection sub-net modules increases. Specifically, in Section 4, the mean creation time of the *iCPT* for sensor SP1 is higher. This is because the presence of water in the tray can be caused by overflow if water flows into the tank via Section 1 without flowing out through the outlets in Sections 2 and 3 of the system.



CPT for a Propagated Variable with 3 states

1-0-0-1-0-0-0-0-0	0.333333	0.333333	0.333333
1-0-0-0-1-0-0-0-0	0.333333	0.333333	0.333333
1-0-0-0-0-1-0-0-0	1	0	0
1-0-0-0-0-0-1-0-0	0.333333	0.333333	0.333333
1-0-0-0-0-0-0-1-0	0.333333	0.333333	0.333333
1-0-0-0-0-0-0-0-1	0.333333	0.333333	0.333333
0-1-0-1-0-0-0-0-0	0.333333	0.333333	0.333333
0-1-0-0-1-0-0-0-0	0.333333	0.333333	0.333333
0-1-0-0-0-1-0-0-0	0.333333	0.333333	0.333333
0-1-0-0-0-0-1-0-0	0.333333	0.333333	0.333333
0-1-0-0-0-0-0-1-0	0.333333	0.333333	0.333333
0-1-0-0-0-0-0-0-1	0.333333	0.333333	0.333333
0-0-1-1-0-0-0-0-0	0.333333	0.333333	0.333333
0-0-1-0-1-0-0-0-0	0.333333	0.333333	0.333333
0-0-1-0-0-1-0-0-0	0.333333	0.333333	0.333333
0-0-1-0-0-0-1-0-0	0.333333	0.333333	0.333333
0-0-1-0-0-0-0-1-0	0.333333	0.333333	0.333333
0-0-1-0-0-0-0-0-1	0.333333	0.333333	0.333333

Total Number of States 3
Number of Rows in the CPT 18

Figure 3: Samples of the Automatically Generated Conditional Probability Tables

Table 1: Creation Time of Some Selected Input Conditional Probability Tables

S/No	Conditional Probability Table with:	Mean Creation Time (ms)	Half width	Confidence Interval ($CI_{95\%}$)	% Error at $CI_{95\%}$
1	2 states, 1 row (e.g., Pipe P1 states)	28.84	1.51	[27.33, 30.35]	5.24 %
2	6 states, 1 row (Tank level discretisation states)	29.34	1.52	[27.82, 30.86]	5.18 %
3	3 states, 18 rows (e.g., Sensor S1 readings)	29.64	1.53	[28.11, 31.17]	5.16 %
4	2 states, 2 rows (e.g., Flow sensor VF1 states)	30.54	1.55	[28.99, 32.09]	5.08 %
5	2 states, 48 rows (e.g., States of sensor SP1 states in the tray)	30.78	1.56	[29.22, 32.34]	5.07 %

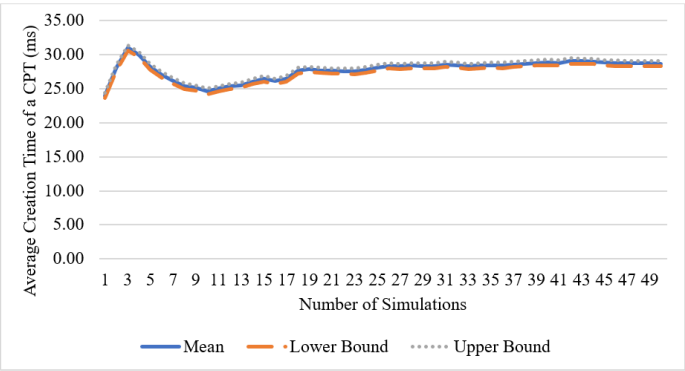


Figure 4: The Mean Creation Time of iCPT for Pipe P1 State against the Number of Simulations

5. Conclusion

This paper proposes an algorithm for automatically generating conditional probability tables (CPTs) based on the GSPN module in a GSPN-mBSPN model of a dynamic system. The algorithm was implemented in a C++ bespoke code developed for the modelling and analysis of a GSPN-mBSPN model. A GSPN-mBSPN model of a water tank level control system was used as a case study to check the efficiency of the implemented algorithm in terms of speed and accuracy. The CPTs from the analysis of the GSPN-mBSPN model of the water tank level control system demonstrated improved accuracy and efficiency in defining CPTs for a GSPN-mBSPN model of a dynamic system. Thus, this proved the effectiveness of the implemented algorithm. Future research aims to show how the improved GSPN-mBSPN approach could automatically generate the fault diagnostic module (mBSPN) of a GSPN-mBSPN through further analysis of the enhanced GSPN-mBSPN approach. Besides, the application of the method for fault detection and diagnosis of a dynamic system under time-varying dynamic conditions will be explored in future work.

Acknowledgement

The authors would like to thank Petroleum Technology Development Fund (PTDF), Abuja, Nigeria, for funding and supporting Taofeeq Alabi Badmus PhD research through the fund Overseas Scholarship Scheme (OSS) programme [award number P6797054741222445].

References

Andrews, J. (2013) 'A modelling approach to railway track asset management', *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 227(1), pp. 56–73. Available at: https://doi.org/10.1177/0954409712452235.

Chen, Y.C. *et al.* (2010) 'Petri-net based approach to configure online fault diagnosis systems for batch processes', *Industrial and Engineering Chemistry Research*, 49(9), pp. 4249–4268. Available at: https://doi.org/10.1021/ie901410p.

Forner, L., Kumar, V.S. and Kinshuk (2013) 'Assessing design of online courses using bayesian belief networks', in *Proceedings - 2013 IEEE 5th International Conference on Technology for Education, T4E 2013*. IEEE Computer Society, pp. 36–42. Available at: https://doi.org/10.1109/T4E.2013.17.

Hurdle, E.E., Bartlett, L.M. and Andrews, J.D. (2009) 'Fault diagnostics of dynamic system operation using a fault tree based method', *Reliability Engineering and System Safety*, 94(9), pp. 1371–1380. Available at: https://doi.org/10.1016/j.ress.2009.02.013.

Kabir, S. and Papadopoulos, Y. (2019) 'Applications of Bayesian networks and Petri nets in safety, reliability, and risk assessments: A review', *Safety Science*. Elsevier B.V., pp. 154–175. Available at: https://doi.org/10.1016/j.ssci.2019.02.009.

Liu, J. and Zio, E. (2017) 'System dynamic reliability assessment and failure prognostics', *Reliability Engineering and System Safety*, 160, pp. 21–36. Available at: https://doi.org/10.1016/j.ress.2016.12.003.

Nourredine, O. *et al.* (2023) 'A new generalized stochastic Petri net modeling for energy-harvesting-wireless sensor network assessment', *International Journal of Communication Systems* [Preprint]. Available at: https://doi.org/10.1002/dac.5505.

Russell, S.J. and Norvig, P. (2010) *Artificial Intelligence A Modern Approach*. Third Edition. Edited by M.J. Horton et al. Upper Saddle River, New Jersey: Pearson Education, Inc.

Shukla, D.K. and Arul, A.J. (2020) 'Static and dynamic reliability studies of a fast reactor shutdown system using smart component method', *Annals of Nuclear Energy*, 136. Available at: https://doi.org/10.1016/j.anucene.2019.107011.

Taleb-Berrouane, M., Khan, F. and Amyotte, P. (2020) 'Bayesian Stochastic Petri Nets (BSPN) - A new modelling tool for dynamic safety and reliability analysis', *Reliability Engineering and System Safety*, 193. Available at: https://doi.org/10.1016/j.ress.2019.106587.

Vagnoli, M. and Remenyte-Prescott, R. (2022) 'Updating conditional probabilities of Bayesian belief networks by merging expert knowledge and system monitoring data', *Automation in Construction*, 140. Available at: https://doi.org/10.1016/j.autcon.2022.104366.

Vasilyev, A. *et al.* (2021) 'Dynamic Reliability Assessment of PEM Fuel Cell Systems', *Reliability Engineering and System Safety*, 210. Available at: https://doi.org/10.1016/j.ress.2021.107539.

Xu, J. *et al.* (2023) 'A new approach for dynamic reliability analysis of reactor protection system for HPR1000', *Reliability Engineering and System Safety*, 234. Available at: https://doi.org/10.1016/j.ress.2023.109147.

Zhou, Y. *et al.* (2018) 'Using Bayesian network for safety risk analysis of diaphragm wall deflection based on field data', *Reliability Engineering and System Safety*, 180, pp. 152–167. Available at: https://doi.org/10.1016/j.ress.2018.07.014.