Modelling Simple Network Graphs Using the Matrix Vector Transition Net

A. Spiteri Staines

***Abstract*—** Petri nets and graph modelling are popular topics in computing and other fields. Different structures and complexities can be represented using graphs. In information graphics graph topologies find suggested use for representing formal and informal natural occurring structures and relationships. Basically graph types can be divided into two simple types i) directed and ii) non-directed. Other characteristics and properties can be included. Certain types of simple graphs can easily be represented using matrix structures.

The incidence matrix, and the adjacency matrix for a graph can be used as input or outputs in the MVTN (matrix vector transition net). The MVTN was presented in previous work. The reasons for combining graph structures with the MVTN are discussed in section II) and III). The MVTN is based on Petri net principles. Some advanced uses of the MVTN and using graph structures as inputs are given in sections IV) and V). Some simple examples are also given and discussed in section VI). These models can be executed and preserve Petri net like properties. They can be further enhanced. Some results and findings are presented in section VII).

***Keywords*—** Graphs, Matrices, Matrix Vector Transition Net, Modelling, Networks, Petri Nets.

I. Introduction

MODERN systems have many different types of relationships and dependencies which can be internal or external. Some types of relationships are easily representable and describable using graphs and similar structures [1]-[7]. This is visible in software modelling methods like the UML. In the simplest form a network graph is a set of nodes joined by a set of lines or arrowed directed lines known as edges.

Some typical examples of network graphs are: collaboration networks, friendship graphs, association graphs, relationship graphs, transportation networks, computer communication systems, internet structures and intranetworking. Some network graphs have a short life and whilst others are more long lasting. For transport systems [1], logistics, process workflows, computer and network architectures, etc., entity relationships can be explained using graphs. Even certain types of communication problems and network structures used in computing can be represented using graphs [2]-[3]. These can provide alternative viewpoints.

Graph representation can simplify the problem of system representation by breaking down the structure into smaller counterparts that are easier to represent and understand.

Natural Graphs can range from simple ones that occur naturally in diverse structures to more complex ones. One basic example of natural graph structures are those that occur in social networking. Some of these structures range from simple to complex linking and can have a high density. Usually there is some form of natural or unnatural correspondence between graph structures and real world entities [14]-[17].

Different forms of representational matrices can be used for representing graphs. Typically these can be classified into two main types which are i) incidence matrices and ii) adjacency matrices. These two can be further classified into other forms like weighted, non-weighted, etc.

Modern systems are composed of different elements and are time variant or dependent configurations. One simple way used for describing the relationships between these systems is to use graphs. Graphs can visually depict the system structure or the existing interrelationships in these system configurations.

Graph structures in computing and other similar fields can be considered to be abstract data types based on sound mathematical principles useful for representing a variety of structures and events. Sometimes there exists a quasi-intuitive representation of certain structures in a graph format. These structures and the MVTN (matrix vector transition net) [2]-[4] are visual modelling notations that can be seen and understood. In the fields of computing, graph like structures are widely used across different notations and methods. E.g. UML activity diagrams are a form of digraphs. Class diagrams are graphs that have special node and edge types. Sequence diagrams can be reduced to digraphs. Petri nets are bipartite digraphs.

II. Motivation

This paper is restricted to basic graph structures mainly using directed or undirected graphs. This can be considered to be an oversimplification of what happens in real life. However many complex structures in the real world can be reduced to some simpler form making them more understandable and readable thus omitting parts that might not be interesting [8]. In simple form graphs consist of an ordered set of nodes and edges also known as arcs and vertices. Certain graphs may assign values to these or use some form of symbolic labelling. Three basic structures can be used to represent the graphs [9]. These are i) adjacency lists, ii) adjacency matrices and iii)

Anthony (Tony) Spiteri Staines, is with the Department of Information Systems, Faculty of ICT, University of Malta, (corresponding phone: 00356-21373402,e-mail: toni_staines@yahoo.com)

incidence matrices. In the i) adjacency list, vertices are stored as records or objects. Every vertex stores a list of adjacent values. This type of structure can be enhanced to store more detail. The ii) adjacency matrix is a simple 2-d structure where the rows represent vertices and the columns represent destination vertices. The iii) incidence matrix is a 2-d matrix where rows represent vertices and columns represent edges. The entries indicate how the vertices relate to the edges and vice-versa [9].

III. Related Work and Background

This work shows how the adjacency and incidence matrices for simple graphs can be used as inputs and outputs for the matrix vector transition net. This MVTN has been defined and elaborated in previous works [2]-[4]. In essence this modelling structure is a modified Petri net where instead of ordinary places, matrices or vectors are used. More modelling power and greater detail is obtained [3].

One key feature of simple graph structures is that these can be represented using matrices. Hence this work explores the possibility of using actual graph representation matrices as inputs or outputs from the MVTN for modelling purposes [3]. The graph representational matrices can be combined as inputs and outputs from the MVTN [4]. This signifies that the MVTN can be used to test the graph structures via their respective matrices. This means that the MVTN can be used to detect if a particular structure is present.

There is the possibility of using the MVTN for modifying the graph structures. Edges and nodes can be added or removed as required. This would be relevant for modern systems where their network elements can be reconfigured dynamically in real time.

Graphs have been used to model computer networks and various types of configurations. Simple graphs can be represented using adjacency and incidence matrices. These are visual representations of structures and complexity [9]-[12], [18].

This approach shows how it is possible to model complex communication between different processing elements using Petri net like behavior. Different cases like those of an abstract switch, producer – consumer problem, network message routing etc. have been shown in [2]-[4]. The MVTN structures are fully executable and the operations can be represented using simple matrix algebra operations [4].

Place transition nets have been used to model different systems and scenarios [5]-[8], [13]. In [4] the workings of the abstract switch are clearly explained. This is a fully executable example. A reachability graph can be generated for each successive transition firing. The key difference is that the MVTN uses matrices or vectors for inputs and outputs instead of normal Petri net places [2]-[4].

IV. Problem Statement

Dynamic systems exhibit changes during their useful life. Changes are brought about via transitions. Modern system configurations are dynamic in nature. Normally a system also has static and dynamic parts. The system structure or configuration can be effected. At any given moment in time a system has a perceived global state composed of sub states [12]. Once an event takes place the global state of the system and also its structure might change. Changes in a system take place at certain time intervals until the system ceases to exist. Changes can be brought about by internal or external activities. Complex dynamic systems are decomposable into discrete components. Any transition taking place in a system can be 'memoryless' or have 'memory'. Memory transition are more complex [12]. The MVTN and graph representational matrices can be used for creating some form of 'collective memory' effect [3].

The graph representation where changes are time variant can be used to keep a trace of a system. Any 'event' that takes place can change the configuration in real-time. This type of behavior is typical in self adapting networks and the idea of self-healing systems. Ordinary and more advanced classes of Petri nets can be used to model the memoryless systems [12],[13]. I.e. the transitions can be considered to have a memoryless effect. This is not entirely true for complex systems.

In simple terms the problem definition is to represent a system structure using a simple graph and use this graphs as an input or output for the MVTN and add details. A system graph structure can be used to start or stop a transition or it can be the output or input of a transition for special types of processing and experimentation. The idea of this work is to find new realistic uses for the MVTN. These should be practical uses and relate to real world issues and problems. It should be possible to compact a system and find simpler ways of representing behavioral structures. These structures should represent and identify transitions that change the system configuration. Different components of the system should be clearly identifiable. These components are considered to be integral parts of the system composition.

Normally in Petri net theory the inputs to the system can also be called actors [12]. Here the actors are actual system structures or configurations. A syntactical definition of a system from the viewpoint of the MVTN can be given as a composite system of the form composite system::=graph structure* <controlled events>.

Model based verification and analysis are useful prior to actual design so that the functionality of the models are tested and checked before actually going ahead. This type of analysis can be applied even to existing systems. This can help to obtain a good comprehension of what is actually occurring.

V. Proposed Solution

The actual problem solution is very simple. Simple graph structures can be represented using either an incidence matrix or an adjacency matrix. These matrices can be combined with the MVTN. The proposed solution to combine the MVTN with other constructs is represented in fig. 1. The general idea

is that an input graph is transformed into a matrix representational structure which is processed or checked using the MVTN. An output matrix or some other output is obtained from the system. If an output matrix is produced this can be converted or transformed into a graph. The graph representational matrices can be used as inputs/outputs from the MVTN. The MVTN can also be combined or linked to other Petri nets or Petri net constructs as shown in previous

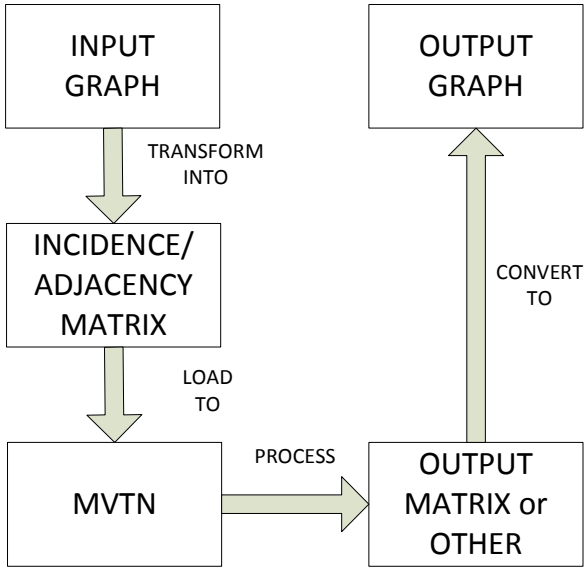


Fig. 1 using graph representation matrices with the MVTN

work [3]. So, these structures can be used for complex workings. E.g. it is possible to check for a particular graph structure and if the graph structure is correct then fire a particular transition or enable a certain execution pathway. Fig. 2 shows the general idea of the proposed solution

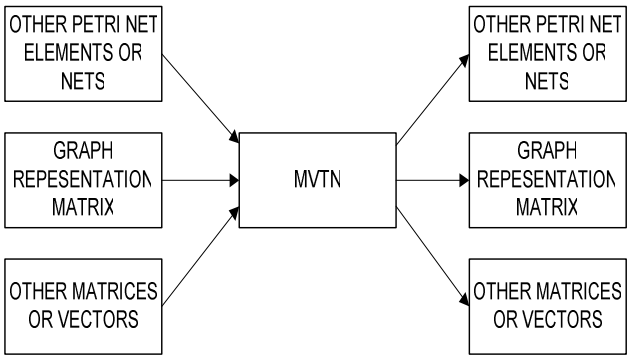


Fig. 2 graph representation matrices and other elements

indicating how different notations and elements can be combined with the matrices for the graphs being used.

There are some constraints and restrictions that must be explained for this solution: i) graphs used as input must have a limited or finite structure. ii) The MVTN input matrix for the graph has to be allowed to have negative values. iii) The graphs must have proper labelling for the nodes and edges. iv) matrices require proper labelling.

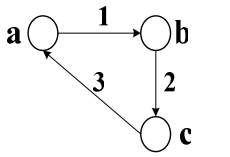
Obviously if there is no ending or incidence matrix for the graph structures this cannot be done. One observation and modification to the MVTN is that the matrix can contain negative values. This was not allowed in previous work. Thus the matrices need proper labelling and have to be given an indication of what they can contain.

VI. Case Study: Simplified Examples

The elementary and basic examples presented in this section are used for indicating practical use of this approach and testing its validity.

A. Basic Graph Example

Fig. 3 shows a basic digraph with its i) incidence and ii) adjacency matrices. These matrices are used as inputs for the MVTN. The graph can be transformed into the matrices and



$$\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

INCIDENCE MATRIX

$$\begin{array}{c} a \\ b \\ c \end{array}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

ADJACENCY MATRIX

Fig. 3 simple digraph with its i) incidence and ii) adjacency matrix

vice versa. It is important to label the graph edges and nodes for the process of transformation. In the graph incidence matrix the rows represent the nodes and the columns represent the edges. The negative values show an input edge i.e. the direction of the edge and a positive value shows it is an output edge.

B. Adding an Extra Edge, Adjacency matrix and MVTN

This example in fig.4 shows how an extra edge is added to a simplified non directed graph using the MVTN and the output matrix. Here a simple edge is added between D2 and D3. This is simply done by incrementing the corresponding entry

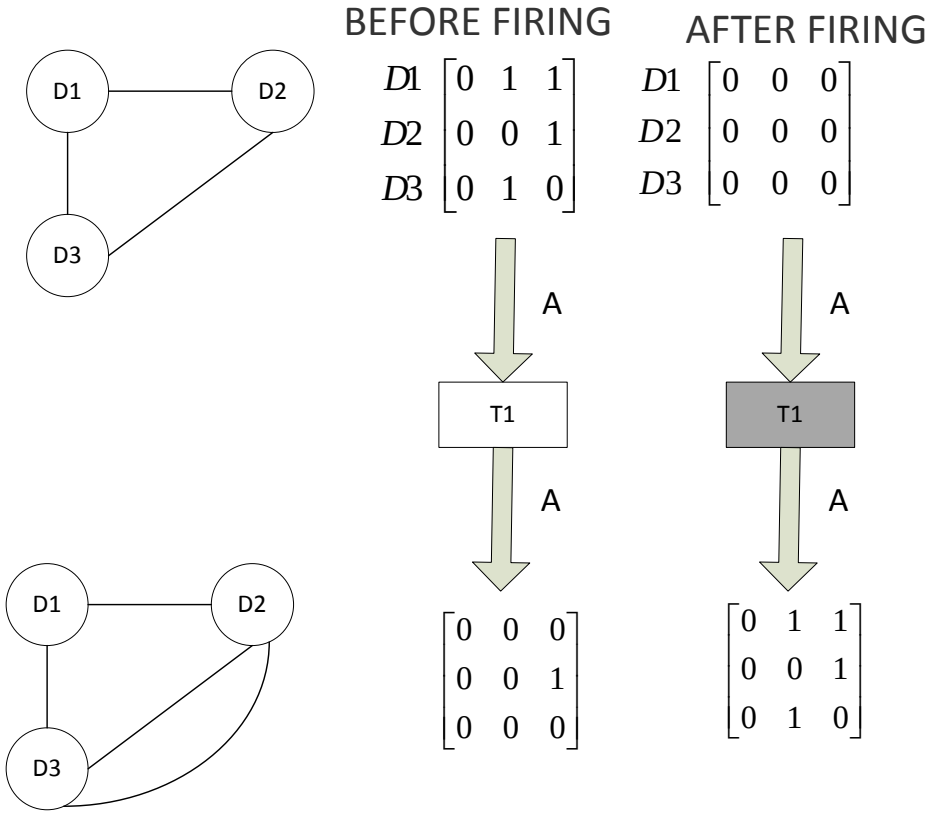


Fig. 4 adding an extra edge via adjacency matrix

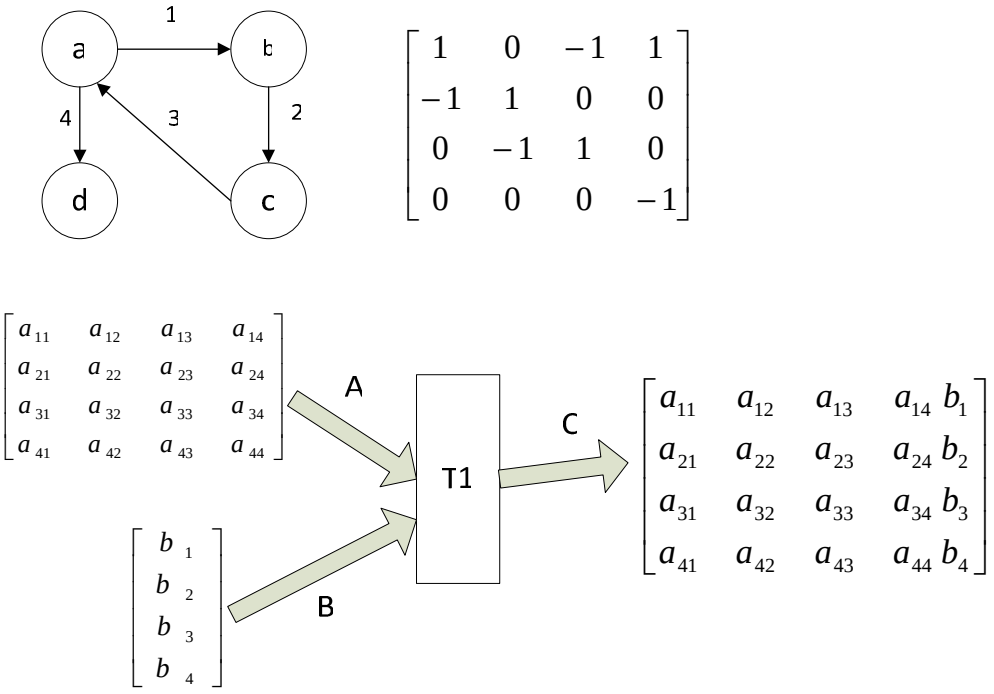


Fig. 5 adding an extra edge to a digraph via incidence matrix

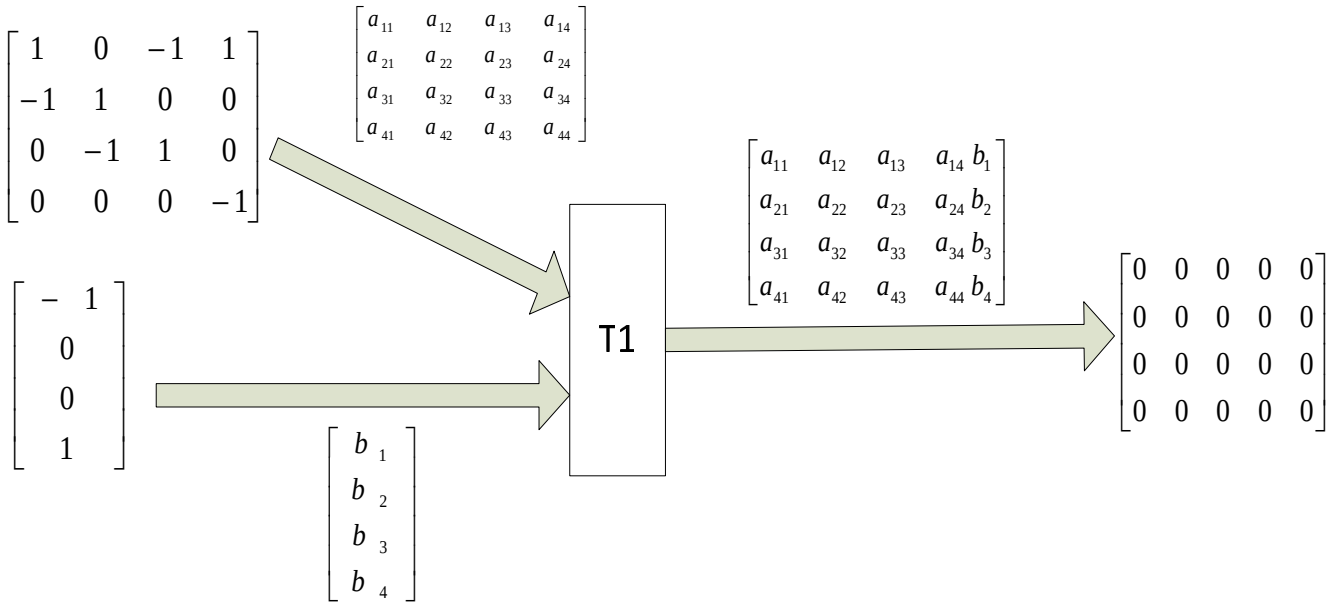
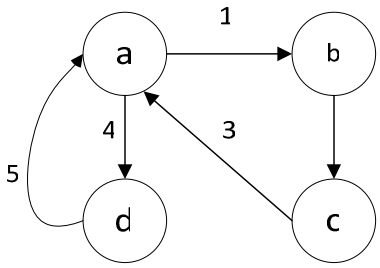


Fig. 6 MVTN with proper inputs and outputs to add extra edge

in the input matrix by 1. The contents of the output matrix on the output of the MVTN are added to the input matrix. This action can even be completed or carried out in other ways. The adjacency matrix is more suited for undirected graphs.

C. Adding an Extra Edge, Incidence Matrix and MVTN

Another approach to adding an extra edge is presented using a digraph. For this purpose, the incidence matrix of the graph is used. The incidence matrix seems to be better suited to model digraphs in this work. The graph is shown on the top l.h.s. of fig. 5. The corresponding incidence matrix is shown on the top r.h.s. of fig. 5. Fig. 6 shows the MVTN with more added detail and Fig. 7 shows the final result with the final output matrix and the corresponding modified graph. The example presented in fig. 5,6 is more detailed for the sake of clarity.



$$\begin{bmatrix} 1 & 0 & -1 & 1 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Fig. 7 final result after firing T1

D. Combining two Simple Graphs and MVTN

This example in fig. 8 briefly outlines how two separate graphs can be combined into a single graph. In fig. 8 two distinctly labelled graphs are shown. The output of the example is just a simple combination of two graphs. The graphs are not linked to each other and remain disjoint even in the combined matrix. The graphs can be joined with some modifications.

E. Checking for a Particular Graph

This example shows how a graph representation matrix can be used for checking and firing transitions in the MVTN. In Fig. 9 the matrix A is checked to see if it contains the correct values for the incidence matrix for a particular graph. If this is the case then A-B yields a null value and transition T1 can fire. Two tokens are removed from P1 and two tokens are placed in P2 and one token in P1 respectively. More comprehensive examples of how two separate graphs can be checked can be created. These can be used to trigger associated transitions in the MVTN.

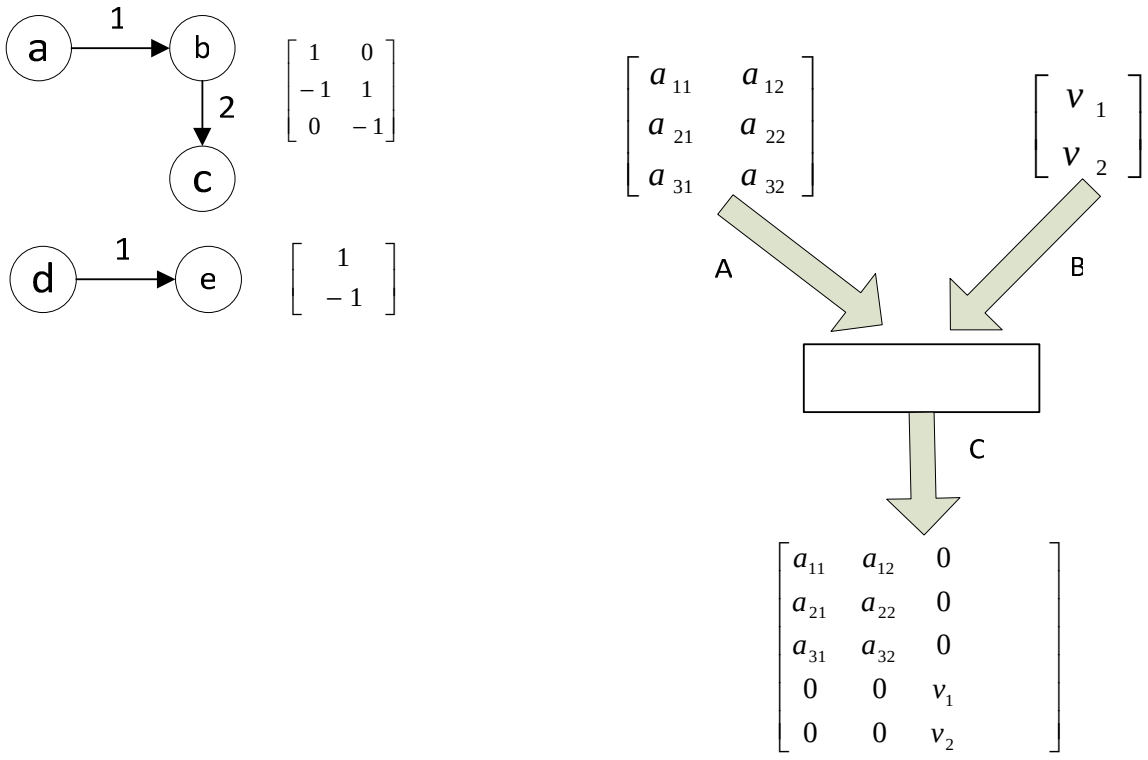


Fig. 8 combining two graphs via the MVTN

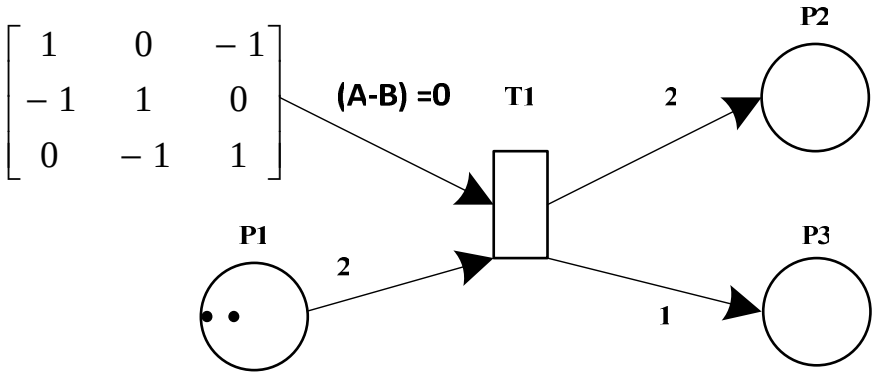


Fig. 9 checking a graph representation matrix

VII. Results and Findings

The simplified case study examples clearly show that it is possible to use the MVTN to model and check very simple directed and undirected graphs. For particular graph types the adjacency matrix is more suitable than the incidence matrix. However the incidence matrix can provide detail about the edge direction for digraphs. Similar results could be obtained from different configurations or setups of the MVTN. It is up to the user to select what is most appropriate for a particular scenario. The MVTN can be used to remove and add edges to a graph via its representational matrix. It can be used to combine two separate graph matrices into a singular matrix. One of the most tedious parts is the conversion of a graph into its representation matrix and the reconversion back into a graph.

It is possible to test the graph representation matrix for the presence of a particular graph. The examples given are executable models that are based on Petri net like theory. Even though just a handful of examples have been given in this paper, these can serve as the building blocks to create even more complex structures and models. More advanced

modelling approaches can be defined from the ideas presented.

Dynamic network structures can be created using this approach. The structures can be used for system diagnosis and decision purposes. The following structures can be represented using these models: i) Dynamic system structures, ii) representational structures, iii) validation, verification and test structures and iv) structures for performance criteria.

More practical uses of this work would be to apply these models for the following areas: i) route testing, ii) route reconfiguration, iii) optimization based on structural topology modification and iv) testing.

The models presented in the previous section show mainly the following processes: i) edge adding, testing, and ii) combining two structures. The opposite processes could be defined e.g. edge removal and splitting a graph into separate entities etc.

VIII. Conclusion

Various new ideas can be explored in future studies. However there are always some limitations. The focus here was more on a theoretical approach to modelling graphs via their matrices. The ideas presented can serve to study particular network problems. These are useful for visual representation and modelling. They can be used in the area of information graphics. Matrices are particularly interesting because they can contain useful information in a condensed and concise format.

The limitations of this work are related mainly to complexity issues and the amount of processing involved. Graph structures can become overly complex if the number of edges and nodes are increased substantially [18]. Complex graphs have not been considered.

Other problems exist when dealing with graphs of a more complex nature i.e. possibly having a mixture of node types and interrelationships. This is obvious if the massive classification of graph types available in literature is examined. In this case reduction and simplification methods could be used.

Sometimes for simple problems the graph representational matrices can be used on their own hence the MVTN would not really be necessary.

References

[1] A. Spiteri Staines, A Colored Petri Net for the France-Paris Metro, NAUN, International Journal of Computers, Issue 2,Vol. 6., 2012,pp. 111-118.

[2] T. Spiteri Staines, F. Neri, A Matrix Transition Oriented Net for Modelling Distributed Complex Computer and Communication Systems, WSEAS Transactions on Systems, Vol 13, 2014, pp. 12-22.

[3] A. Spiteri Staines, Extending the Matrix Vector Transition Net Approach for Modelling Interaction, New Developments in Circuits, Systems, Signal Processing, Communications and Computers (CSSCC), INASE, WEIN, Austria, 2015, pp. 126-132.

[4] T. Spiteri Staines, Implementing a Matrix Vector Transition Net, British Journal of Mathematics & Computer Science, ISSN: 2231-0851,Vol.: 4, Issue.: 14, 2014, pp. 1921-1940.

[5] T. Spiteri Staines, Representing Petri Nets as Directed Graphs, Proceedings of the 10th WSEAS international conference on Software engineering, parallel and distributed systems SEPADS'11, WSEAS, Cambridge UK, 2011, pp. 30-35.

[6] A. Spiteri Staines, From Task Graphs to Petri Nets, International Journal of Emerging Technology and Advanced Engineering, Vol 3, Issue 5, May 2013,pp. 36-42.

[7] A. Spiteri Staines, Some Fundamental Properties of Petri Nets, International Journal of Electronics Communication and Computer Engineering, IJECCE, vol.4, Issue 3,pp. 1103-1109.

[8] M.B. Dwyer, L.A. Clarke, A Compact Petri Net Representation and its Implications for Analysis, IEEE Transactions on Software Engineering, vol. 22, issue 11, 1996,pp. 794 – 811.

[9] K.H. Rossen, Handbook of Discrete and Combinatorial Mathematics (Discrete Mathematics and Its Applications), CRC PRESS, 1999.

[10] K.M. Abadir, J.R. Magnus, Matrix Algebra, Cambridge University Press, 2005.

[11] F. Ayres (jr), Theory and Problems of Matrices, Schaum's Outline Series, Schaum, 1974.

[12] K. van Hee, Information Systems: A Formal Approach, Cambridge Univ. Press, 2009.

[13] CPNTools, CPN Group, Department of Computer Science, University of Aarhus, Denmark http://cs.au.dk/CPnets/

[14] B. Scholz-Reiter, C. Zabel, Integration of Load Carriers in a Decentralized Routing Concept for Transport Logistics Networks, WSEAS Proceedings of the 2nd International Conference on Theoretical and Applied Mechanics (TAM '11) Corfu, Greece, 2011,pp. 259-264.

[15] C. Knieke, B. Schindler, U. Goltz, and A. Rausch, Defining Domain Specific Operational Semantics for Activity Diagrams. Technical Report IfI-12-04, TU Clausthal, Clausthal, Germany, 2012.

[16] J. Osis, and E. Asnina, Topological Modelling for Model-Driven Domain Analysis and Software Development: Functions and Architectures, Model-Driven Domain Analysis and Software Development: Architectures and Functions, 2010, pp. 15-39.

[17] Tracking Design Changes with Formal Machine-Checked Proof, Higher Order Logic Theorem Proving and Its Applications, Vol: 859, LNCS, Springer-Verlag,1994, pp. 177-192

[18] J.L. Gross, J. Yellen, Handbook of Graph Theory: Discrete Mathematics and its Applications, CRC PRESS, 2003.