Graph Drawing Approaches for Petri Net Visualisation and Representation

ANTHONY SPITERI STAINES
Department of Computer Information Systems
University of Malta
MSIDA 2080
MALTA

Abstract: - Petri net structures can benefit from being modelled using graph drawing approaches. This work presents some advanced graph drawing approaches that can be used for visualising Petri net models. These are i) topological shape metrics, ii) visibility drawing, iii) orthogonalization, iv) hierarchical and v) bi-graph partitioning. The models show that this can be successfully done and the Petri nets that are generated in this work are suitable for visualisation. Graph visualisation is an important topic and the results show that there is a large potential to apply this approach to drawing Petri Nets in novel ways.

Key-Words: - Representation, Matrices, Ordinary Petri nets, System Modelling

Received: November 3, 2020. Revised: May 30, 2021. Accepted: June 11, 2021. Published: July 6, 2021.

1 Introduction

Graphs in all their forms and their related theory are an integral part of computer system modeling. They are used extensively for representation and design purposes [1]-[11], [18]-[20].

Graphs help to transform observational patterns that just exist at the level of mind into visible notations. These are useful for investigating computer systems and their behavioural characteristics. Models in the modern world have multiple use. This is not only limited to the computer science domain but can be practically used to observe and understand complex systems in engineering, planning, manufacturing, telecommunications etc. Models allow users to make planning decisions based on experience and good planning characteristics [1].

Design of computer systems and programming is not the most complex type of behavior that can be identified. E.g. the equilibrium model of gas in space used to predict behavior and an economic model that represents a system's behavior are not so simple to represent and prediction is a problem.

The tool to deal with complexity is abstraction. Abstraction involves paradoxically ignoring some or many of the complex details in a system. Models that reflect the macroscopic behaviour provide for abstraction. These models distinguish the simplification of enormous complexities. The OCCAM razor principle of simplicity is imperative for working with models. Ideally an abstract model should be related to contemporary programming languages, system design concepts and computer architectures. But ideally the modelling language or notation should be written using a style or presentation that can withstand the test of time!

This presents the idea that abstract concepts are reasoned about formally or mathematically exhibiting sound structural principles. Still this is being lost in the ability to abstract and then to implement it. This sequence can be changed.

Abstraction implies decomposing the problem into smaller manageable/ understandable tasks. These can be defined as sub-problems that are easier to comprehend or work on.

A patterned approach can be used to solve the problems. Two main structures are identifiable in computing i) control structures and ii) data structures. As an analogy these are similar to sequential logic and combinatorial logic used in circuit design and analysis. Petri net formalisms use graph representation along with mathematical notations. Parallelisms can be modeled in asynchronous, distributed, parallel, deterministic type of systems [1]-[9]. The drawing and representation of Petri nets will benefit greatly from the application of graph drawing techniques found in literature. There are some forms of equivalence between Petri nets and other graphical notations.

Many of the mainstream approaches used in graph drawing for visualization and presentation of graphs e.g. planarization and orthogonalization in conjunction with other approaches can be applied to Petri net drawing [1]-[11]. This work can be extended to other domains of modeling e.g. the UML graphical models.

Graph drawing approaches have a long term of coverage and are useful in modern development [18]-[20]. Graph drawing approaches for Petri Net modeling are important because they could be used to create more uniform and readable Petri Net model drawings. This would save a reader time to interpret the model. The models drawn could be aesthetically more pleasing to look at. The main contribution of this work is the application of graph drawing approaches to Petri net models. Whilst graph drawing approaches have been extensively used in software engineering, they have not often been combined with Petri nets. This work deals with this problem and idea.

2 Background and Motivation

Graph drawing approaches are not a new topic and they have extensive use in the field of software modelling. They also have useful applications for visualisation.

Petri nets are useful formalisms presented graphically. They share a dual identity having mathematical representation. Petri nets [14]-[15] share several common properties with digraphs [1],[2]. They are useful for representing static and dynamic structures and have found extensive use in literature and in various fields. Graph drawings are like syntactic glue that hold the system properties together [1]-[11].

Different strategies and algorithms exist for graph drawing and visualization. Several papers and books exist that promote different types of algorithms that claim to be improvements of others. In reality it is very difficult to select the best strategies and algorithms because there is so much to choose from. In essence, in the simplest form a graph is just a set of edges and vertices where there are certain dependencies. More details and dependencies can be added to the graph complicating things e.g. directed edges, weights, etc. The techniques used for graph drawings are applicable to drawing Petri nets and other modelling notations.

It is impossible to cover all the material and ideas related to graph drawing [1]-[11] but some ideas can be given. i) Graph drawings are used for visualization so there are many different styles to represent graphs and improve their layouts. ii) There are several ways to represent the same layout. This point is derived from i). Drawing and layout of graphs implies that the same structure could be drawn or improved using several techniques [10]. This creates the fundamental problem of selecting the best technique. iii) Graph drawing is related to computational geometry. iv) Planar and orthogonal graph drawings seem to offer the best choice for Petri net representation and visualisation for the fundamental reasons that when Petri nets are drawn planarization of the graph can offer aesthetically nice features that create a good drawing for the user. Planarization and orthogonalization are useful for circuit schematic drawings used in engineering which closely represent Petri net drawings. Even if planarization is considered there are several features that can be included in planarization e.g. the rotation of the edges, planar polyline and strictly upward planar polylines etc. v) Aesthetics is a complex topic that cannot really be solved. Certain representations are better than others but this depends on user perception. I.e. what determines the best layout of a graph? vi) The issue of scaling and size. This refers to minimizing the area used for drawing. Several approaches used in conjunction with orthogonal and planar drawing can be used for this. However how much should scaling be used? What is the cut-off point for scaling? vii) Partitioning the graph. Partitioning a graph implies that the graph is divided into sections that improve its readability and structure. Partitioning seems to be related to top-down strategy where the graph is partitioned using a bi-graph approach. By using bi-graphs the graph is restructured and the overall result is a well presented structure consisting of several sub-graphs that are usefully and resourcefully linked together. viii) Hierarchical drawing implies a top-down strategy for drawing graphs. This simple approach can be used for Petri net drawing. The concept of hierarchy can be combined with other graph drawing approaches [1]-[11].

These constraints and other concepts like symmetry, minimization, precedence, efficiency, etc. can add more weight to the graph drawing problems.

The main motivation for this work is to show the importance of the above mentioned techniques for representing and visualizing Petri net related structures [12]-[16] and the extensive amount of work that can be done in this area. A balance has to be found between aesthetics and over engineering of Petri net diagrams.

3 Problem Definition

Graph representations and structures are very important tools for visualisation and modelling. Graphs are widely used in the field of requirements engineering and software engineering. Petri nets can be considered to be graphical and mathematical types of formalisms that have widespread use. Drawing and representing Petri nets as graphs is not a trivial task. The attractiveness of the drawing, based on principles of layout, simplification, abstraction and aesthetics will have a great impact on the usability. I.e. users prefer to use a model that looks attractive. When edges and nodes are properly harmonised, the model looks good and feels right.

In this regards graph drawing techniques can be used to draw Petri nets which are normally drawn as planar bipartite digraphs. There are no standards or established procedures for Petri net drawing. It is obvious that several mixed and non-uniform methods are used. Several Petri net drawings are not 'optimized' from the graph drawing point of view. E.g. the edges can be part curves or straight lines, etc. Sometimes even the nodes are of different sizes.

There are so many different graph drawing methods [1]-[7]. Which is the best for drawing Petri nets? Even for a simple question like: Should the drawing of the Petri nets use a i) planar approach or ii) orthogonal approach? There is no easy straightforward answer. It is not possible to find a single approach that is the best so several approaches have to be considered and presented [8].

There is also the problem of how to represent the Petri net to transform it into a graph [12]. The information that is contained in the normal incidence matrix is not enough. It does not show the actual inputs and outputs. To see these, the separate input and output flow matrices have to be examined. Edge representation can get cancelled out in the incidence matrix. E.g. a self-loop would not show up. More structures or matrices have to be used. The incidence matrix would give an ambivalent view from the input and output matrices.

After representing the Petri net in a condensed or compacted form [12], then it could be possible to apply various graph drawing techniques to represent the model graphically. The main stream techniques and other problems that have to be considered are listed in the background and motivation section in part 2. It would be useful if some form of structured or semi-structured approach is given to do this. There can be some order to be followed.

The actual implementation of the algorithms for graph drawing and their real effectiveness is assumed. This work does not focus on their implementation.

The problems defined in this part can open up many more high level problems and sub-level problems that require further investigation and might not get a clear cut answer from the viewpoint of the user. This is because of user perception and individual likes.

Some of the graph representation approaches can conflict with one another [1]-[11]. One of the main issues is that of representation. In this context representation implies finding other forms of depicting or showing the same structure. Which is the best? If too many details are shown the diagram or notation is uncompacted. The compacted form is smaller or condensed. A reduced form has a more elegant representation [12]-[13].

The problem is to find different representation of the same structures. Why? Relational structures like Petri nets, P-nets, bipartite graphs are useful to the degree that they can effectively communicate with different groups of people and stakeholders. A good diagram is properly laid out and easy to understand at a glance.

Petri net drawings [12]-[15] or representations are graphical structures. In essence they are bi-partite digraphs composed of vertices and edges. They are drawn using geometrical shapes like circles for places and boxes (square or rectangular) that represent transitions. Material for graph drawing is widely available in literature and the internet. Certain representations might be more difficult than others to follow.

This work is a theoretical experiment that tries to identify and classify several graph drawing and representation methods that can be used in conjunction with drawing / redrawing Petri nets.

Various approaches are used for drawing Petri nets. E.g. sometimes i) straight lines are used, ii) curved edges and iii) orthogonal drawings, iv) Petri nets that are compartamentalized and v) a mixture of several approaches. These approaches are found in [1]-[11].

4 Experimental Solutions

This work is a theoretical experiment that tries to identify and classify several graph drawing and representation methods that can be used in conjunction with drawing / redrawing Petri nets.

The proposed solutions are based on robust approaches used in the field of graph drawing and representation. The solutions in this paper will basically focus on the transformation of the Petri net data to a graph structure and its optimization.

Examples will be given along with solutions. The solutions or proposed methods are as follows:

i) Compact Petri net matrix form for graph drawing
ii) Topological shape metrics drawing approach
iii) Visibility shape metrics drawing approach
iv) Hierarchical drawing approach
v) Bi-graph drawing approach

The techniques mentioned above are given in general form. Some steps have also been changed. I.e. representation of the graph is done using the matrix given in 4.1.

4.1 Compact Petri Net Matrix Representation

The form of the matrix R depicted below is used to represent a basic Petri net. The rows denote the places in the Petri net and the columns denote the transitions [17]. Each entry in the matrix is a two dimensional vector of form *v= (a,b)* where *a* represents the input flow (input edge) from a place Pn to a transition Tn and *if exists* this value is denoted negatively *else* a zero value is placed. Whilst *b* is the output from a transition Tn to a place Pn and *if exists* this value is denoted positively *else* a zero value is placed.

$$R=\begin{bmatrix}(a11,b11) & (a12,b12) & (a13,b13) & \dots\dots & (a1n,b1n)\\ (a21,b21) & (a22,b22) & (a23,b23) & \dots\dots & (a2n,b2n)\\ (a31,b31) & (a32,b32) & (a33,b33) & \dots\dots & (a3n,b3n)\\ \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots\\ (an1,bn1) & (an2,bn2) & (an3,bn3) & \dots\dots & (ann,bn\_)\end{bmatrix}$$

As an example the matrix below can be used to create or draw the Petri net in fig. 1

$$\begin{bmatrix}(-1,0) & (0,1) & (0,0)\\ (0,0) & (-1,0) & (0,1)\\ (0,1) & (0,0) & (-1,0)\end{bmatrix}$$

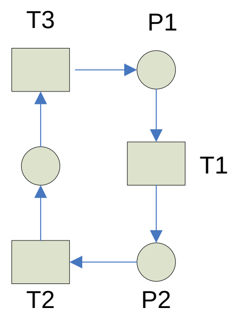

Fig. 1. Petri Net derived from matrix

4.2 Topological Shape Metrics Approach

In essence the topological shape metrics graph drawing technique presented in [1] can be given in four main steps which are: i) graph matrix representation, ii) planarization, iii) orthogonalization and iv) compaction. The examples presented below show this process as applied to Petri net graph drawing.

i) The following matrix represents a Petri net having three transitions and two places.

$$\begin{bmatrix}(-1,0) & (0,1) & (0,1)\\ (0,1) & (-1,0) & (-1,0)\end{bmatrix}$$

ii) The next step is planarization and a Petri net can be drawn. At this stage there are several possible drawing strategies that can be used. E.g. for just representing the edges (e.g. polyline or straight-line, polyline grid, arc diagrams etc.)

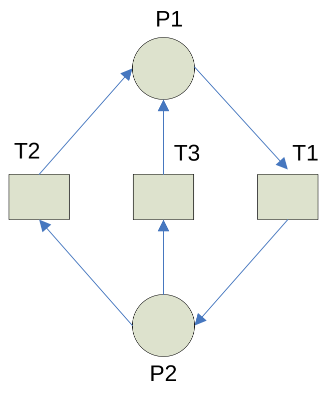

Fig. 2. Petri Net drawing after planarization

iii) The next step is orthogonalization and the Petri net in fig. 2 is redrawn using principles or orthogonality in graph drawings. Again there are many algorithms, techniques and principles that are involved in orthogonality. This is depicted in fig. 3.

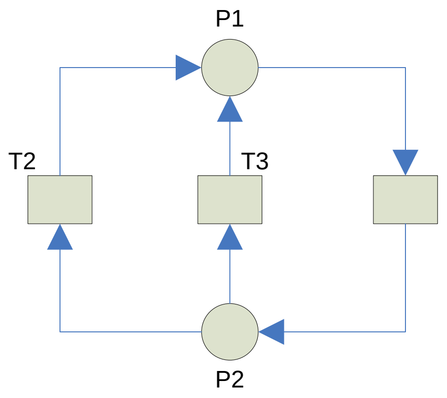

Fig. 3. Petri Net drawing after orthogonalization

iv) Compaction or reduction is a straightforward reduction of the model size [1]. Compaction is useful for creating small meaningful models. The idea of compaction or reduction is given in fig. 4 which follows from fig. 3.

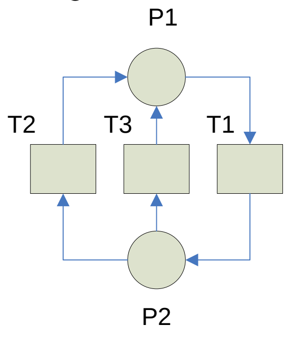

Fig. 4. Petri Net drawing after orthogonalization and compaction

4.3 Other Possible Forms of Orthogonalization

Several other possibilities are available for orthogonalization [1],[6]. Fig. 5 shows an alternative orthogonalization representation for fig. 3 and fig. 4 models.

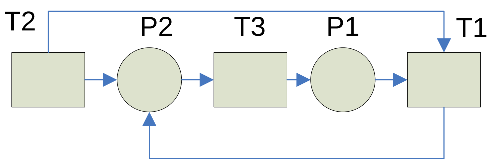

Fig. 5. Petri Net drawing after flattening out

The diagram in fig. 6 shows various possible forms of the orthogonalization flattening out process. Kindly note that this model is not the same Petri Net as in fig. 5.

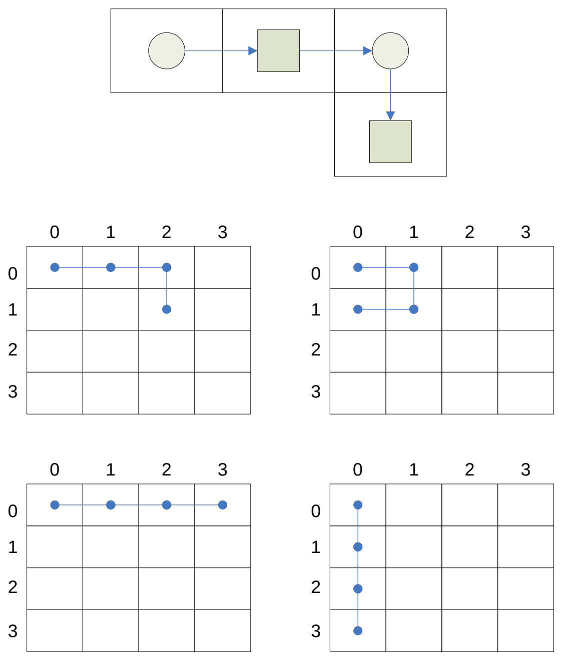

Fig. 6. Other possible forms or orthogonalization

4.4 Visibility Shape Metrics Approach

This name implies that the approach is a general purpose technique for producing graphs using the polyline drawing method. The name of this approach might sound confusing, it is basically a simple approach. For this process some steps are similar to those of the topological shape metrics approach [1]. The first step would be the i) graph matrix representation, ii) planarization, iii) visibility step that constructs a visibility representation of the graph. Basically every edge can be mapped to a vertical segment whilst every vertex or node is mapped to a horizontal segment. iv) replacement step constructs a final polyline drawing b replacing the horizontal and vertical segments. Several approaches can be used [1]-[5].

Here a partial example is given. Just the visibility steps and the replacement steps are presented for simplicity's sake [1]. Strategies are possible and the reader is reminded that the examples given in fig. 7 and 8 are just tentative artistic experimental illustrations or drawings of what can be done and how the algorithms could work and that this has not been tried out in practice.

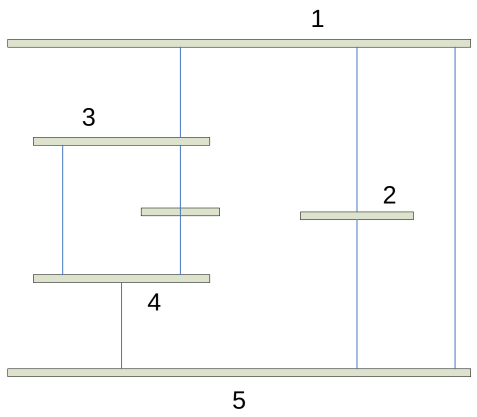

Fig. 7. Experimental visibility representation of a Petri Net

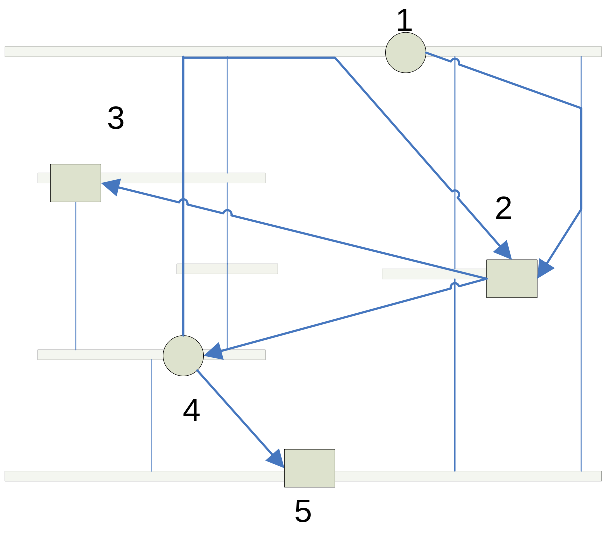

Fig. 8. Possible tentative replacement step creating an alternative visibility approach Petri Net model.

4.5 Hierarchical Drawing Approach

In the actual hierarchical drawing approach steps like i) layer assignment, ii) crossing reduction and iii) x-coordinate assignment etc [1],[4],[9] are used. Dummy nodes are used in various steps and removed later. However here the hierarchical approach given does not follow the actual algorithm and is introduced as an intuitive idea instead. The basic process for the hierarchical drawing approach [4] is to create a top-down or bottom up graph and add the nodes gradually. *This idea or pattern follows the creation of a downward creation of a digraph through the addition of nodes and edges as required*. This type of modeling should be intuitive and follows analogies and patterns used in certain abstract data types, trees, B-trees, decision trees, Bayesian networks, etc.

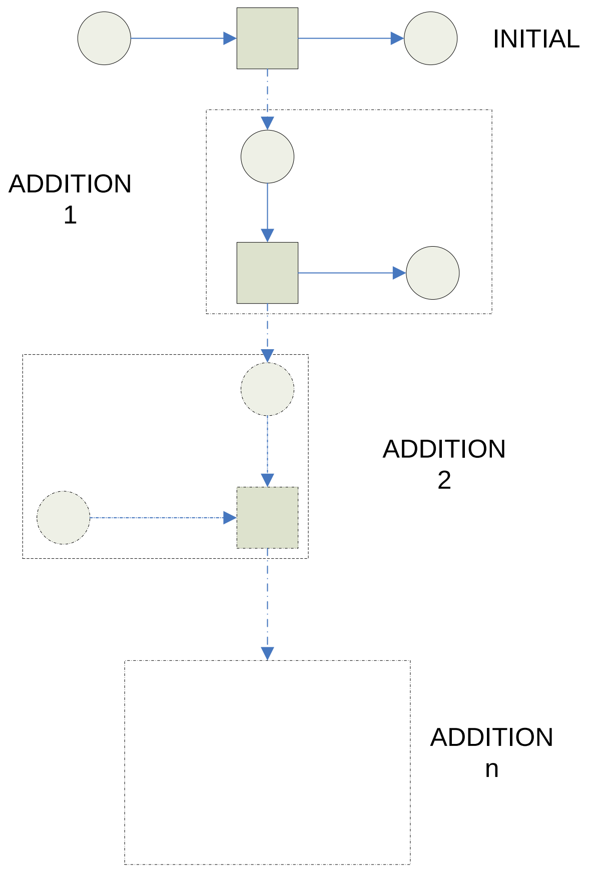

Fig. 9. Tentative hierarchical graph drawing approach applied to Petri net drawing

4.6 Bi-Graph Partitioning Approach

The bi-graph drawing approach [11] is intuitive and the proposed idea is that of i) creating sub-graphs of the main graph and ii) placing the sub-graphs into regions and compartments. This is an idea of general decomposition and divide and conquer process. There are several ways and strategies how this can be done and this can be based on the graph's spatial or mathematical properties. In this work it is shown how this can be done via the compact Petri Net matrix representation.

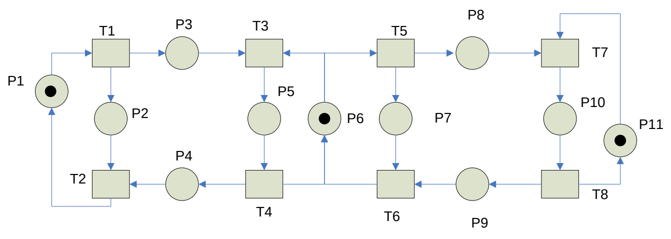

Fig. 10. Petri Net model suitable for partitioning

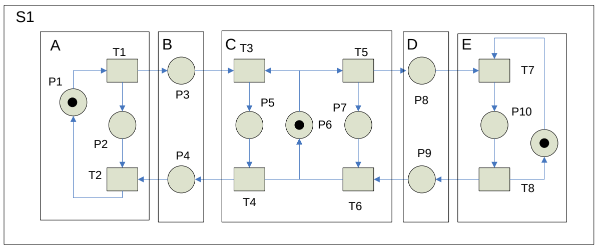

Fig. 11. Bi-graph representation via partitioning of model in fig. 9

The model in fig. 10 is a normal Petri Net suitable for the bi-graph partitioning drawing approach. Fig. 11 represents the partitioning of this model. Fig. 12 shows the compact Petri Net representation matrix. The same partitioning of fig. 11 can be done using this compact Petri Net matrix representation. The partitioning is fig. 11 this is mainly done using the places p1 – p11 of the model drawn in fig. 9.

	T1	T2	T3	T4	T5	T6	T7	T8
P1	(-1,0)	(0,1)						
P2	(0,1)	(-1,0)						
P3	(0,1)		(-1,0)					
P4		(-1,0)		(0,1)				
P5			(0,1)	(-1,0)				
P6			(-1,0)	(0,1)	(-1,0)	(0,1)		
P7					(0,1)	(1,0)		
P8					(-1,0)		(0,1)	
P9						(0,1)		(-1,0)
P10							(0,1)	(-1,0)
P11							(-1,0)	(0,1)

Fig. 12. Compact Petri Net matrix representation for fig.9

In fig. 12 the compact Petri Net matrix is given and the grouping in the matrix can be intuitively done using just the places. E.g. P1 and P2, P3 and P,…, etc. From fig. 12 the separation and groupings for the places is self-evident.

	T1	T2	T3	T4	T5	T6	T7	T8
P1	(-1,0)	(0,1)						
P2	(0,1)	(-1,0)						
P3	(0,1)		(-1,0)					
P4		(-1,0)		(0,1)				
P5			(0,1)	(-1,0)				
P6			(-1,0)	(0,1)	(-1,0)	(0,1)		
P7					(0,1)	(1,0)		
P8					(-1,0)		(0,1)	
P9						(0,1)		(-1,0)
P10							(0,1)	(-1,0)
P11							(-1,0)	(0,1)

Fig. 13. Compact Petri Net matrix representation for fig.10 showing partitioning as for fig. 11

Fig. 13 shows possible partitioning into sub-graphs as can be used for the bi-graph approach.

5 Assumptions and Limitations

Some assumptions have been made for this study. i) It is assumed that algorithms exist or algorithms can be created, altered and modified accordingly in order to support the transformation processes described here. ii) another important part is that the Petri Nets to be used are ordinary, iii) the Petri Nets are properly labeled using correct sequences, iv) the Petri Nets are of a restricted size, v) the Petri Nets can be simplified or compacted as required. vi) It is important to note that even though the approaches look similar to those used in graph drawing certain rules could be changed or manipulated as required. Even though the title could look similar the approach might be different.

Additionally there are some limitations to this work. It has been assumed that the graph drawing techniques can be modified as required to support the transformation of the Petri net models. Arrows, line drawings, node layout etc. all require other work. The use of dummy vertices for planarization and orthogonalization have not been considered.

Some of the transformations are rather intuitive whilst other transformations are rather complex and can involve a lot of steps. The examples used for this work are very simply toy models. This work can also be used for modeling other classes of Petri nets. This might require some changes and additions to the approach. It requires further investigation.

6 Some Findings and Discussion

Some of the interesting findings are as follows:

i) Petri nets graphical representation need to be properly represented and laid out for visualisation.

ii) Graph drawing techniques and algorithms used for drawing and visualization of graphs can be applied to draw Petri nets which are in essence normal bi-partite graphs with some other properties!

iii) There are several graph drawing techniques and algorithms that can be used with possible modification added to them as required.

iv) If the Petri net has more nodes and edges than the more complex and difficult it becomes to represent.

v) The size of the Petri net diagram affects its readability.
vi) Aesthethical graph drawing principles can also be applied to Petri nets.
vii) Graph drawing visualization techniques can be taught as skills to software engineers so that they properly draw comprehensible models. (etc. could be included in model drawing skills).
viii) This topic is important for model driven engineering and graph-to-graph model transformations.
ix) Combination of several graph drawing techniques is possible.

This theoretical experiment shows the importance of graph structures. Drawing graph structures using automated tools is a complex task and problem. It would involve several equations, methods and several computer algorithms complicating things. The drawing process is complex and that several computations would be involved in this task. This work can be compared with [18] and [20], which share similar principles.

Even if any one of the approaches mentioned in the paper e.g. orthogonalization as shown in 4.2, is considered, there are so many different algorithms and techniques involved that several works could be expounded just on this small part.

The idea of the work presented in this paper can be transferred to other areas in software engineering such as requirements engineering or software modelling where graph type objects are used for representing different structures like architectural representation. Other uses of these graphs can be used to model things like the IoT, cloud computing, etc.

7 Concluding Comments

The area of graph drawing is a very interesting area that derives its importance from the field of mathematics and graph theory. Many essential computer related structures used in hardware and software can be represented graphically. Petri nets can be represented in different graphical formats. The drawing and layout or Petri Nets is important for readability, attractiveness and aesthetical reasons.

Pictures are the traditional means for communicating and simply depicting system representation to persons. Normally much attention is paid to the graphical or pictorial part. Petri nets can be used to serve both ends.

Some possible limitations of the methods described in this paper are that they are computationally complex and expensive to use.

This paper can be the starting point for educating software engineers in graph representation. This is a very important area.

References:

[1] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, *Graph Drawing Algorithms for the Visualization of Graphs*, Pretence Hall, 1998.

[2] G. Di Battista, A. Garg, G. Liotta, R. Tamassia , E. Tassinari , F. Vargiu, An Experimental Comparison of Four Graph Drawing Algorithms, *Computational Geometry Theory and Applications,* Vol: 7, Elsevier, 1997, pp. 303-325.

[3] N. Chiba, K. Onoguchi, T. Nishizeki, Drawing Plane Graphs Nicely, T. *Acta Informatica*, Vol :22, 1985, pp. 187-201.

[4] K. Sugiyama, S. Tagawa, M. Toda, Methods for Visual Understanding of Hierarchical System Structures, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol: 11 , No: 2 , 1981, pp. 109-125.

[5] M. Eiglsperger, M. Kaufmann, M. Siebenhaller, A Topology-Shape-Metrics Approach for the Automatic Layout of UML Class Diagrams, · *Proceeding of SoftVis '03,* ACM, 2003, pp. 189-198.

[6] I. Córdoba, G. Varando, C. Bielza, P. Larranaga, A partial orthogonalization method for simulating covariance and concentration graph matrices, 2018.

[7] N. Gelfand, R. Tamassia, Algorithmic Patterns for Orthogonal Graph Drawing, *Graph Drawings*, 1998.

[8] R. Tamassia, G. Di Battista, C. Batini, Automatic Graph Drawing and Readability of Diagrams, *IEEE Transactions on systems, Man and cybernetics*, Vol: 18, no: 1, 1988, pp. 61-79.

[9] S. C. North, G. Woodhull, On-line Hierarchical Graph Drawing, *Proc. Of the 9th Int. Symposium on Graph Drawing*, 2001, pp. 232-246.

[10] M. Alqadah, G. Stapleton, J. Howse, P. Chapman, Evaluating the Impact of Clutter in Euler Diagrams, Proc. of 8th International Conference, Diagrams 2014 pp. 108-122.

[11] R. Milner, Pure bigraphs: Structure and dynamics, *Information and Computing*, Vol. 204, Issue 1, Elsevier, 2006, pp. 60-122.

[12] A. Spiteri Staines, Alternative Matrix Representation of Ordinary Petri Nets, Transactions on Computers, WSEAS, Vol 18, 2019, pp. 11-18.

[13] A. Spiteri Staines, Implementing a Matrix Vector Transition Net, BJMCS, Vol. 4, Science Domain, 2014, pp. 1921-1940.

[14] T. Murata, Petri nets: Properties, Analysis and Applications, *Proc. of IEEE*, vol. 77, issue 4, 1989, pp. 541-580.

[15] M. Zhou, K. Venkatesh, Modeling, *Simulation, And Control Of Flexible Manufacturing Systems: A Petri Net Approach* (Series in Intelligent Control and Intelligent Automation), World Scientific, 1999.

[16] A. Spiteri Staines, Matrix Representations for Ordinary Restricted Place Transition Nets, *Transactions on Computers*, WSEAS, Vol 16, 2017,pp. 23-29.

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

[18] V. Kasayanov, E. Kasayanova, T. Zolotuhin, Visualization of Graph Representations of Data-Flow Programs, *Transactions on Information Science and Applications*, WSEAS, Vol 15, 2018, pp. 140-146.

[19] C. Easttom, M. Adda, An Enhanced View of Incidence Functions for Applying Graph Theory to Modeling Network Intrusions, ***WSEAS** Transactions on Information Science and Applications*, ISSN / E-ISSN: 1790-0832 / 2224-3402, Volume 17, 2020, Art. #12, pp. 102-109.

[20] V. Kasyanov, T. Zolotuhin, A System for Big Attributed Hierarchical Graph Visualization, ,WSEAS Transactions on Computers, ISSN / E-ISSN: 1109-2750 / 2224-2872, Volume 17, 2018, Art. #18, pp. 151-155.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

Author Contributions:
Anthony (Tony) Spiteri Staines was responsible for writing all sections and all the paper. (i.e. sections 1-9) including the abstract material. Also the experiments of section 4 and the results and findings are the work of this sole author.

This paper and its research has been funded by a research grant given to the author by the University of Malta, Malta.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)