Petri Network Simulator

This was realized first as a project for Petri Network Course. This course was taught by Professor Marin Popa at students from second year of University Spiru-Haret, Faculty of Mathematics Computer-Science. After that I developed further more the project creating the actual project who is deposed at http://sourceforge.net/projects/petrinetwork/.

This simulator simulate two of most common Petri Network: mPN (Mark Petri Network) and mJPN (Mark Jump Petri Network) and find a different characteristics for graphs and digraphs.

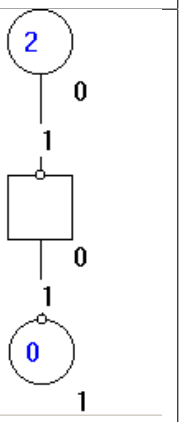
The simulator was made in Visual C++ 6.0 as Multiple Interface Document.

The purpose other purpose of this simulator, except the use in the laboratory, it is a starting platform for a more comprehensive simulator for PN and Graphs. For PN with automate generation of accessibility tree and cover graphs, eventually with transition graphs for timing network.

This simulator has the following capabilities:

- takes a PN from a file and realize the graphical PN who can be modified and simulated
- makes the PN from zero in full graphical capability like commercial VISIO
- creates and finds: connexity, cyclicity, euler cycle and minimum cost tree for graphs an digraphs.

Example 1 *A simple mJPN and the corresponding file*

Meaning of fields	file	result
Pre Post Mark Jump	1 0 0 1 2 0 In 0,2 Out 2,0 In 0,2 Out 1,1	

The program has the following user functions to edit the network. These functions are put at disposition to the user by a popup menu.

- **Select view** - it is used to assign this view for teaching.

- **Redone the graph** - redone the graph from the matrix after diferent algorithms.
- **Transition** - insert a new transition in the network. **Remark:** the first transition in the network must has the name 0 and the others in order. You must not have an empty name of transition because this name is use to generate the matrix **Pre,Post** and **Mark.**
- **Location** - insert a new location in the network. **Remark:** the first location in the network must has the name 0 and the others in order. You must not have an empty name of location because this name is use to generate the matrix **Pre,Post** and **Mark.**
- **Arc** - insert an arc with an arbitrary number of points in the network. The first and the last point must be in one location or transition. It is used like option free hand from Corel Draw with button left of the mouse pressed. After you make the arc you could enter the value of the arc or press CANCEL when the default value is 1.
- **Arc from 2 points** - insert an arc with 2 points. The resulting arc id drown by minimal distance between the two points. The first and the second point is made with a click of the left button.
- **Arc from 4 points** - it is similar with the **Arc from 2 points** but it has 4 points.
- **Mark Point** - it increments with one unity the mark from the location where you make a click with the left button of the mouse.
- **Remove Mark** - it decrements with one unity the mark from the location where you make a click with the left button of the mouse.
- **Move Transition** - it moves a transition with the inputs and outputs arcs if there are made from 2 points or only the terminations for the rest of the arcs.
- **Move Location** - it moves a location with the inputs and outputs arcs if there are made from 2 points or only the terminations for the rest of the arcs.
- **Delete Transition** - it removes a transition with the input and output arcs.
- **Delete Location** - it removes a location with the input and output arcs.
- **Delete Arc** - it removes an arc.
- **Auto** - it makes the conversion between matrix format (load from a file with the function from the file menu) to the screen format of the Petri Network.
- **Jump** - if you selected the mJPN from Network Menu with this function you could enter the jump table.

After you input the Petri Network, you can simulate it. For that it is put a Simulation Menu where you have the following functions:

- **Generate** - it generates the matrix Pre,Post, Mark and for JPN also the jump matrix. These matrix are stored in an external file who later you could load with the Load function. The default name for the file is "generate.dat".

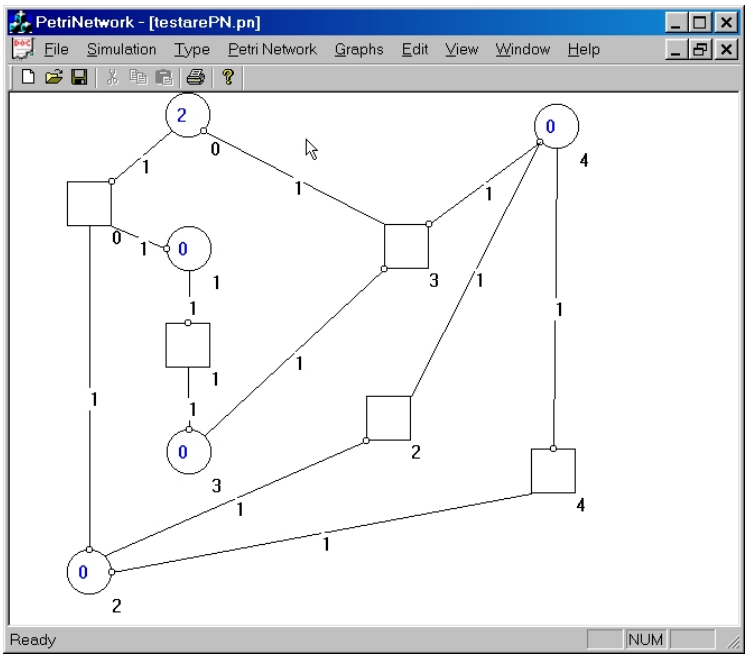
- **Simulate** - it simulates the Petri Network, in this moment only full concurrent mJPN and mPN are supported. The default step of simulation is 1000ms. The simulation is made in a separate WihThread and communicate with the main program with CriticalSection and WinEvent. **Remark:** for a mJPN you could simulate only the equivalent mPN; this is done by selecting from Petri Network Menu the mPN not mJPN.
- **Stop** - it stops the simulation in that point. When you stop the simulation you can operate over the marks and then push **Continue**.
- **Continue** - it restart the simulation who was stopped. Before **Continue** you can modify the marks but you can't modify the structure of the PN.
- **Exit** - exit the simulation.

After you create the graph or digraph like you create the Petri Network, you can use the following functions who select the type of the simulation do you want to simulate:

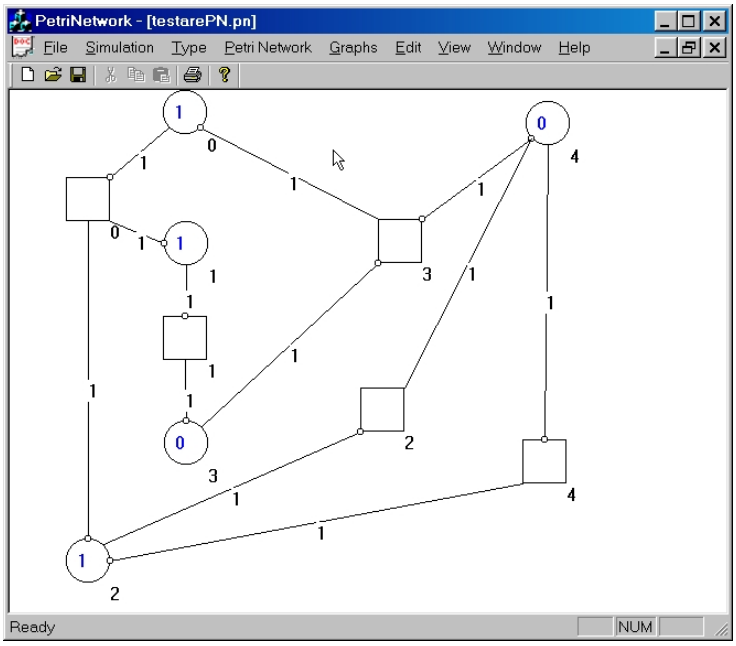
- **Connexity** - it finds if the graph or the digraph is or is not connex.
- **Cyclicity** - it finds if the graph or the digraph is cycle.
- **Eulerian cycle** - it finds an eulerian cycle from the graph or digraph.
- **Tree min cost** - it finds and creates a minimum cost tree from the graph or digraph.
- **Kruskal** it finds and create a minimum cost tree using Kruskal algorithm
- **Prim** it finds and create a minimum cost tree using Prim algorithm
- **Bellman-Kalaba** it finds and create a minimum or maximum path between two nodes.

1 Example for Petri Network

The following petri network is loaded from the following file. After loading I moved the location and transition

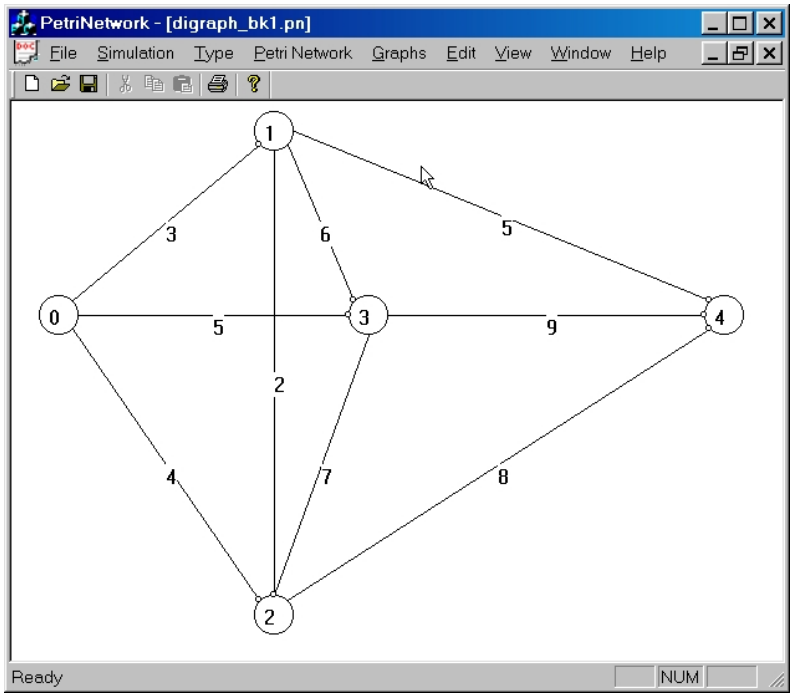


After running serveral steps we have the following configuration

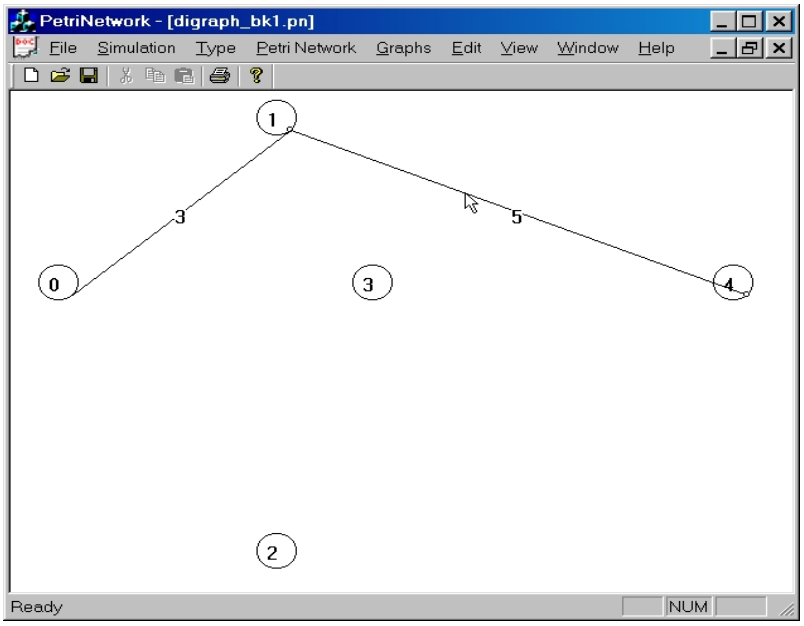


2 Example for digraphs

The problem is to find the minimum path between node 0 and 4 for the following digraph using Bellman Kalaba algorithm.

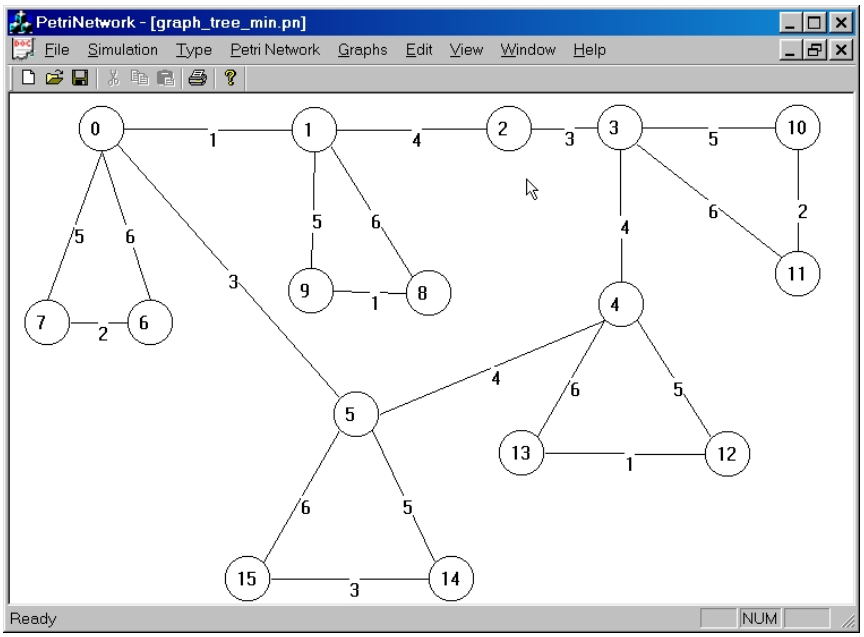


The result is the following graph.

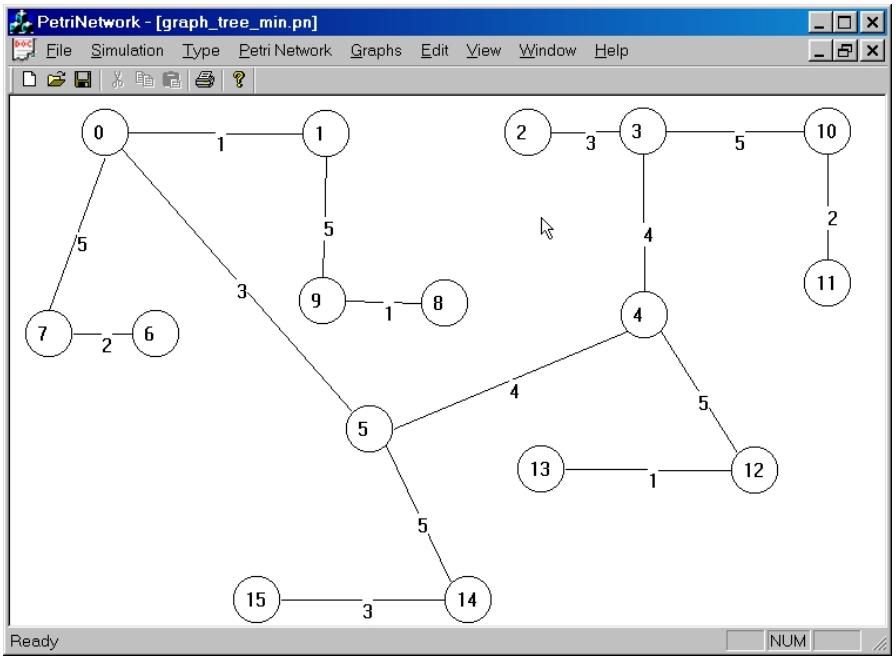


3 Example of finding the minimum cost tree

For the following graph find the minimum cost tree using Kruskal or Prim algorithms.

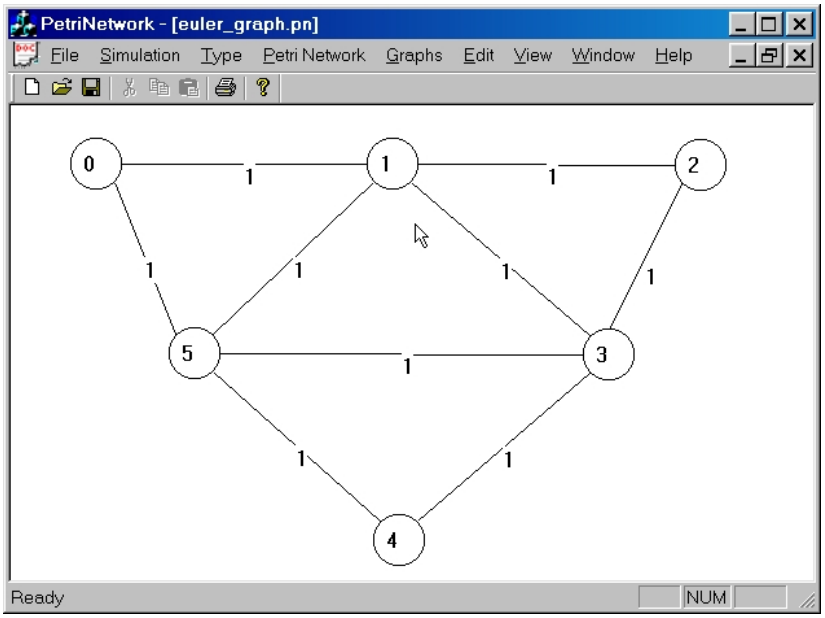


The result is the following graph:



4 Example for Eulerian cycle

Find the Euler cyle for the following graph



The result is the following graph