Activity-Oriented Petri Nets: Aligning Real-world Buffers with Virtual Places

Reggie Davidrajuh
Electrical and Computer Engineering
University of Stavanger
Stavanger, Norway
E-mail: reggie.davidrajuh@uis.no

***Abstract* - Activity-Oriented Petri Nets (AOPN) is a convenient way of modeling and simulation of discrete event systems when a large number of resources are involved. By the AOPN approach, Petri net models are created by focusing only on the activities and by pushing the resources into the background. As a result, the Petri net obtained by this approach works on a virtual platform and the places of this Petri net seemingly has no relevance to the real-world buffers. In case the places are supposed to represent the buffers, this paper explains how the places can be aligned with the buffers. As this paper focuses on a unique problem in AOPN, it is written in a tutorial style, keeping literature study and relevant works to a minimum.**

Keywords - Activity-oriented Petri nets; Petri nets; buffer alignment; buffer dimensioning; GPenSIM; Discrete-event systems

I. INTRODUCTION

Activity-Oriented Petri Nets (AOPN) is a convenient way of modeling and simulation of discrete event systems when a large number of resources are involved [1]. The approach proposed by AOPN creates compact Petri net models of discrete event systems, even if the system possesses a large number of resources. Compact Petri net models are achieved by focusing only on the activities and pushing the resources to the background. AOPN approach has been successfully used to solve problems in academia and in industries [2; 3]. The AOPN approach and the implementation on MATLAB platform (GPenSIM) are developed by the author of this paper.

Focusing only on the activities and neglecting the resources makes the resulting Petri net model operating on a virtual platform; though the places of this Petri net model are supposed to represent the real-world buffers, the places seem to have no relevance to the real-world buffers. Hence, when modeling discrete event systems with the main purpose of studying the number of elements in buffers (e.g. dimensioning buffers in flexible manufacturing system), AOPN approach seems to be ineffective.

The aim and scope of this paper are to show how the places of a Petri net resulting from the use of AOPN approach, can be aligned with the real-world buffers. Since this paper focuses on this unique problem that is applicable only when we use the AOPN approach, this paper is written in a tutorial style presenting materials through an example. In addition, the literature study and the relevant works are kept to a minimum.

In this paper: Section-II introduces the AOPN approach. Section-III presents an application example and the Petri net model is obtained by the AOPN approach in presented in section-IV. Sections V to VII presents the simulation of the application example using GPenSIM.

II. ACTIVITY-ORIENTED PETRI NETS

Activity-oriented Petri Nets (AOPN) is a two-phase modeling approach [1]. In the Phase-I, the static Petri Net model is created. In this phase, mainly the activities are considered and they are represented by transitions embedded with places as buffers. The resources are grouped into two groups such as 'focal' resources and 'utility' resources. Only the focal resources are included in the static Petri Net model; the utility resources will be considered later in the phase-II (the run-time model). Thus, a compact Petri Net model is obtained with only the transitions representing the activities and, if there are any focal resources, they will be represented by places.

In the phase-II, the run-time details that are not considered in the phase-I are added to the Petri Net model; e.g. transitions (activities) requesting, using, and releasing of the utility resources are coded in the run-time model [4].

III. CASE STUDY ON BUFFER ALLOCATION IN FMS

Fig.4 shows a flexible manufacturing system (FMS) with three CNC machines (stations) in a closed environment. Also shown in the figure is the input conveyor belt (ICB) that brings raw materials into the manufacturing environment and an output conveyor belt (OCB) that takes away the manufactured products out of the environment. To move materials among the three stations and between the stations and the conveyor belts, we need utilities like a robot, automatically loading robot, AGV, etc.; these utilities are not mentioned in the figure. However, the figure clearly shows the paths for movement of materials between the stations and the conveyor belts.

A. The Manufacturing Configuration

Let us assume that the FMS is to produce two types of products with the following configuration:

The tasks: There are five tasks to be performed (tasks 'A' to 'E'):

- The Fig.1 shows the dependency graph of tasks, showing the precedence relationship between the tasks.
- Tasks A and B are independent of each other: they can be run in parallel or one before the other. However, these two tasks must be performed on every product.
- Tasks D and E are also independent of each other and can be run in parallel or one before the other. However, these two tasks produce two different classes of products (class-D and class-E).

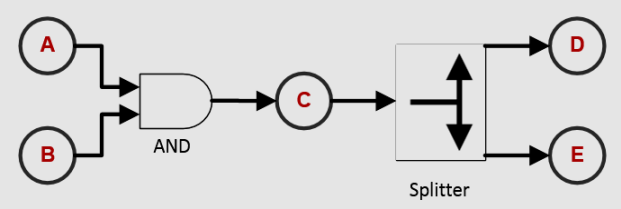


Figure 1. The dependency graph showing the precedence relationship between the tasks.

The stations (machines): there are three stations M1 – M3 available for production. The Table-I given below shows the tasks that can be performed by these machines. According to the Table-I, the tasks A, B, D, and E can be performed by either M1 or M2. Whereas, task C can be performed only by M3.

TABLE I. STATIONS AND THE TASKS THAT CAN BE PERFORMED.

Task	A	B	C	D	E
Station	M1/M2	M1/M2	M3	M1/M2	M1/M2

The products:

- To produce a class-D product: tasks A and B must be performed (in any order) and then task C must be performed. Finally, task D completes production of a class-D product.
- To produce a class-E product: as in class-D, tasks A and B must be performed (in any order) and then task C must be performed. Task E completes production of a class-E product.

B. The Buffers

Let us assume that the stations M1 to M3 are equipped with output buffers on them. These output buffers are for holding the items produced by the respective machines. Let us also assume that there are no input buffers available on these machines. This means, when a station is ready to machine, it triggers a robot or AGV to transport input material from the relevant output buffer of another station or from the input conveyor belt.

IV. PETRI NET MODEL USING AOPN APPROACH

A. The Static Petri net Model

The Petri net model obtained after the phase-I of the AOPN approach is shown in the Fig.2. The Petri net is basically a translation of the dependency graph shown in the Fig.1: the tasks in the dependency graph are represented as transitions in the Petri net; In addition, the precedence relations between the tasks are preserved in the structure of the Petri net. For example, the transitions A1 and A2 represents the task A, and the transitions B1 and B2 represents the task B; the parallel paths "A1-pA-B1" and "B2-pB-A2" are to allow the tasks A and B to occur in parallel or one after the other.

The places pA and pB are the output buffers for the semi-products resulting by the tasks A and B, respectively. However, the place pAB make sure that the tasks A and B has occurred, before the production move to the tasks C and D or E. The parallel paths "pC-D-pD" and "pC-E-pE" is to allow the product flow to follow one of the alternative production line, either the class-D involving the task D or the or the class-E involving the task E. The output buffers of the tasks D and E are pD and pE, respectively.

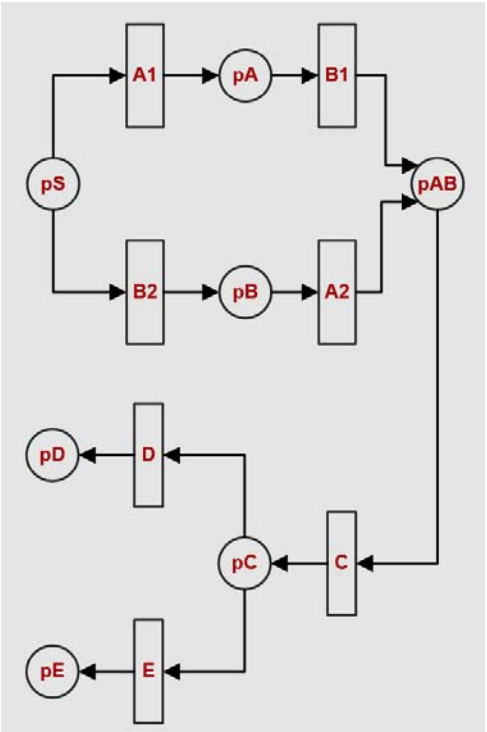


Figure 2. The Petri net model obtained by the AOPN approach.

B. The Run-Time Petri Net Model

In phase-II of the AOPN approach, the relationship between the resources (stations) and the activities (tasks) are considered. Hence, in this subsection, allocating stations to the tasks (or scheduling tasks on the stations) are considered in a step-by-step manner. To make things easier, let us

consider making only one class of products (e.g. class-D) using only one path of product flow (e.g. "A1-pA-B1").

To make one single product of class-D, there are four possibilities (2 × 1 × 1 × 2) for the task-station combination: for example, task A can be performed by either M1 or M2, and then B can be performed by the station that was not used in A. C can be performed only by M3. Finally, D can be performed by either M1 or M2.

No matter what the task-station combination is, the place pS always represent the ICB; since task C can be performed only by M3, place pC always represent the output buffer of M3.

Taking one instance of the four possibilities, let us assume that in making of a product of class-D, the tasks A, B, and D were performed by M1, M2, and M1, respectively, as shown in the Fig. 3.

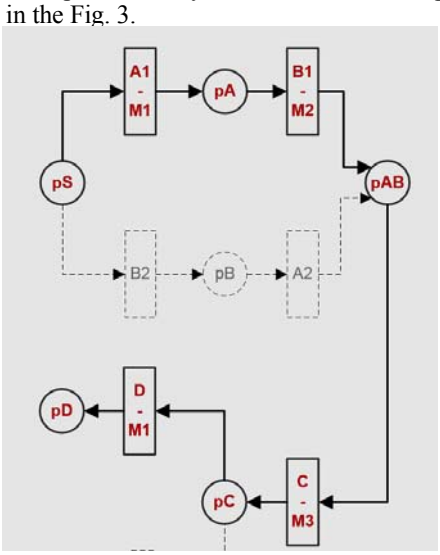


Figure 3. Sample production line for making product type class-D.

In this case (case-1), Table-II given below shows the association of the Petri net places with the real world buffers.

TABLE II. ASSOCIATION BETWEEN PETRI NET PLACES AND THE REAL-WORLD BUFFERS FOR CASE-1.

Petri net place	pA	pAB	pD
Real-world buffer	OB_{M1}	OB_{M2}	OB_{M1}

(OB stands for output buffer)

Let us take another instance of the four possibilities for making a product of class-D; let us assume that in this case (case-2), the tasks A, B, and D were performed by M1, M2, and M2, respectively. The Table-III given below shows the association of the Petri net places with the real world buffers for the case-2.

TABLE III. ASSOCIATION BETWEEN PETRI NET PLACES AND THE REAL-WORLD BUFFERS FOR CASE-2.

Petri net place	pA	pAB	pD
Real-world buffer	OB_{M1}	OB_{M2}	OB_{M2}

Finally, the Table-IV shown below, the association of the places and the buffers for all the four cases.

TABLE IV. ASSOCIATION BETWEEN PETRI NET PLACES AND THE REAL-WORLD BUFFERS FOR ALL THE FOUR CASES.

	A	**pA**	**B**	**pAB**	**D**	**pD**
Case-1	M1	OB_{M1}	M2	OB_{M2}	M1	OB_{M1}
Case-2	M1	OB_{M1}	M2	OB_{M2}	M2	OB_{M2}
Case-3	M2	OB_{M2}	M1	OB_{M1}	M1	OB_{M1}
Case-4	M2	OB_{M2}	M1	OB_{M1}	M2	OB_{M2}

From the Table-IV shown above, it is clear that the places represent different buffers depending on which station the tasks were performed. In other words, the alignment between the places of the Petri net and the buffers of the FMS are dynamic, depending on the run-time situation: which station was used to perform the tasks.

It is important to note that the transitions in the Petri net always represent the respective tasks of the real-world system, as the AOPN approach created the Petri net model focusing on the tasks of the FMS.

Coming back to the Table-IV, it represents the situation when only one path of the product flow "pS-A1-pA-B1-pAB" is considered in making only one class (class-D) of the product. If the whole production line is considered (both paths and both product class), then there will be sixteen cases (2 × (2 × 1 × 1 × 2) × 2) for places to represent real-world buffers of the FMS example.

V. GPENSIM

General purpose Petri net simulator (GenSIM) is a toolbox of functions that run on MATLAB platform [5]. Recently, a team of Australian researchers commended GPenSIM as the ideal tool for modeling and simulation of Event Graphs [6]; Event graph is the main class of Petri nets for modeling manufacturing systems. In addition to its usefulness as a simulator, GPenSIM can also be used a real-time controller [7]. However, one of the main attributes of GPenSIM is that it implements AOPN on the MATLAB platform.

A. Aligning Places with Output Buffers

As the Fig.2 shows, the Petri net model obtained by the AOPN approach is connected with its real-world system only through the activities. The places in the Petri net model represent virtual buffers and are not directly related to the real-world buffers. However, GPenSIM, during the simulation runs, will track the tokens (products) to see which real-world buffer a Petri net place represents during the different points of time. For example: in the FMS problem described in the previous section, the place pA is the output place of the transition A, thus represents the output of the task A. Task A can be performed by either station M1 or station M2. Considering all the products there were machined by M1 or M2 for the task A, pA would represent either the output buffer of M1 or the output buffer of M2, during different periods of the simulation run. From the simulation results, GPenSIM function 'plotOutBuffer' will extract these details (which machine/station was used to perform a specific task) and then plot the dynamic

connection between the places and the real-world buffers, graphically.

B. Aligning with the help of Coloring

It is possible to align the places with the output buffers using two type of extensions to Petri net: the *logical extensions* and the *color extension*. Some of the logical extensions are inhibitor arcs, transition priorities, and enabling functions (enabling functions associated with the transitions) [8]. With inhibitor arcs, we can achieve modeling power equivalent to the same representational power as Turing machines [9]; thus, aligning is possible with the use of inhibitor arcs. However, in GPenSIM, the color extension is used. This is because the color extension is simple and more elegant. GPenSIM function 'plotOutBuffer' uses colors to trace the tokens in the Petri net, in order to map which token belong to which buffer.

VI. Simulations with GPenSIM

Implementing a Petri net model with GPenSIM usually requires four M-files [10]:

1. Petri net Definition file (PDF for short): PDF defines the static Petri net. This means, the PDF simply declares the set of places, the set of transitions, and the set of connections between the places and the transitions.
2. Main simulation file (MSF for short): MSF declares the initial dynamics (e.g. initial tokens, firing times of the transitions, available system resources, etc.) and runs the simulations. The code for plotting and printing the simulations results also are coded in this file.
3. COMMON_PRE file: This file defines the pre-conditions (*aka* 'guard functions') that an enabled transition must satisfy before firing.
4. COMMON_POST file: This file declares all the activities that are to be done after a transition completes firing. For example, this file may instruct a transition to release the resources it was holding during the firing.

The following subsections present some details of these four files for simulation of the FMS example. Due to brevity, the complete code for simulation is not given in this paper. The complete code for simulations with GPenSIM (M-files) is given on the web page [11]; interested readers are encouraged to visit the web page for downloading the codes and experimenting with them.

A. The Petri Net Definition file (PDF)

The PDF shown below declares the sets of places, transitions, and the connections.

```
function [png] = amc2_pdf()
png.PN_name = 'Aligning places with Buffers';
% The set of places
png.set_of_Ps = ...
   {'pS','pA','pB','pAB','pC','pD','pE'};
% the set of transitions
png.set_of_Ts = {'A1','B1','B2','A2','C','D','E'};
% the set of connections (arcs)
png.set_of_As = {...
    'pS','A1',1, 'A1','pA',1, ...  % A1
    'pA','B1',1, 'B1','pAB',1, ... % B1
    'pS','B2',1, 'B2','pB',1, ...  % B2
    'pB','A2',1, 'A2','pAB',1, ... % A2
    'pAB','C',1, 'C','pC',1, ...   % C
    'pC','D',1,  'D','pD',1, ...   % D
    'pC','E',1,  'E','pE',1};      % E
```

B. The main simulation file (MSF)

The MSF shown below declares the initial dynamics and then start the simulations; the code for plotting the simulation results are also coded in this file.

For simulation, the following data is used in the MSF:

1. Ten products are to be made; three are of class-D and the rest 7 are of class-E.
2. The cycle time for the tasks: A1, A2: 5, B1, B2: 7, C: 10, D:3, and E: 7.

```
pns = pnstruct('amc2_pdf'); % the PDF
dyn.m0 = {'pS', 10};  % 10 products are to be made
% cycle time of the tasks
dyn.ft = {'A1',5,'A2',5, 'B1',7,'B2',7, …
     'C',10, 'D',3, 'E',7}; %
% declare availability of the system resources
dyn.re = {'M1',1,inf, 'M2',1,inf, 'M3',1,inf};
pni = initialdynamics(pns, dyn);
% run the simutions with the "gpensim" function
sim = gpensim(pni);
prnschedule(sim); % plot the overall results
% plot place-buffer connection
plotOutBuffer(sim,{{'A1','A2'},{'B1','B2'}});
```

C. The COMMON_PRE file

In this file, the transitions (tasks) request the appropriate resource (station) for execution; the code snippet for this is shown below:

```
function [fire, trans] = COMMON_PRE (trans)
switch trans.name, % name of enabled transition
    case {'A1', 'B2'}
        resource = 'M1';  % first, ask for M1
       granted = requestSR({res, 1}); % request M1
        if not(granted), % if M1 not available
            res = 'M2';  % then, ask for M2
            granted = requestSR({res, 1}); % M2
        end;
…
```

D. The COMMON_POST file

The file is usually compact. The main function of this file is to release the resource (station) that was used by a task when the task is complete.

```
function [] = COMMON_POST(transition)
% after firing, release resource used
release();
```

E. The Simulation Results

Though a number of graphs are plotted as the simulation results, due to brevity, only three of them are shown in this paper. The Fig.5 is the Gantt chart that shows for the tasks and the stations that were used by the tasks. The Fig.6 is two of the eight plots that show the alignment of the places with the buffers during different points in time. Fig.6a shows the output buffer of M1 that was holding the number of semi-products that were made by the task A; similarly, the Fig.6b

shows the output buffer of M1 that was holding the number of semi-products that were made by the task B. These two figures shows that the output buffer of M1 must have the capacity to hold at least 5 products made by task of A and 2 made by B. Similar information on the output buffer dimension of M1, M2, and M3 can be gathered from the other plots not shown in this paper.

VII. Conclusion

Activity-Oriented Petri Nets (AOPN) is a convenient way of modeling and simulation of discrete event systems when a large number of resources are involved. However, the Petri net obtained by the AOPN approach works on a virtual platform and the places of the Petri nets seemingly has no relevance to the real-world buffers. In this paper, a solution is given for aligning the virtual places with the real-world buffers, using colors.

References

[1] R. Davidrajuh. "Activity-Oriented Petri Net for Scheduling of Resources". IEEE International Conference on Systems, Man, and Cybernetics (SMC 2012), October 14-17, 2012, Seoul, Korea.

[2] B. Skolud, D. Krenczyk, and R. Davidrajuh. "Solving Repetitive Production Planning Problems. An Approach Based on Activity-oriented Petri Nets." International Conference on EUropean Transnational Education. Springer International Publishing, 2016.

[3] R. Davidrajuh and B. Lin. "Exploring Airport Traffic Capability Using Petri Net based Model". Expert Systems With Applications (ESWA), 38 (2011) pp. 10923-10931.

[4] R. Davidrajuh, "A New Two-Phase Approach for Petri Net Based Modeling of Scheduling Problems." Industrial Engineering, Management Science and Applications 2015. Springer Berlin Heidelberg, 2015. 125-134.

[5] GPenSIM User Manual. Available at: http://davidrajuh.net/gpensim.

[6] A. Cameron, M. Stumptner, N. Nandagopal, W. Mayor, and T. Mansell, "Rule-based peer-to-peer framework for decentralised real-time service oriented architectures." Science of Computer Programming 97 (2015): 202-234.

[7] R. Davidrajuh, "Developing a Petri Nets based Real-Time Control Simulator," International Journal of Simulation, Systems, Science & Technology (IJSSST), vol. 12, issue. 3, pp. 28-38, November 2012.

[8] G. Ciardo, "Toward a Definition of Modeling Power for Stochastic Petri Net Models," Proc. International Workshop on Petri Nets and Performance Models, 1987, pp. 54-62.

[9] T. Agerwala, A complete model for representing the coordination of asynchronous processes. Baltimore, Maryland: Hopkins Computer Research Report 32, Johns Hopkins University, 1974.

[10] R. Davidrajuh, "Developing a new Petri net tool for simulation of discrete event systems." 2008 Second Asia International Conference on Modelling & Simulation (AMS). IEEE, 2008.

[11] GPenSIM Source Code for the Case Study. Available: http://davidrajuh.net/gpensim/2016-AMC-2-Places-Buffers.

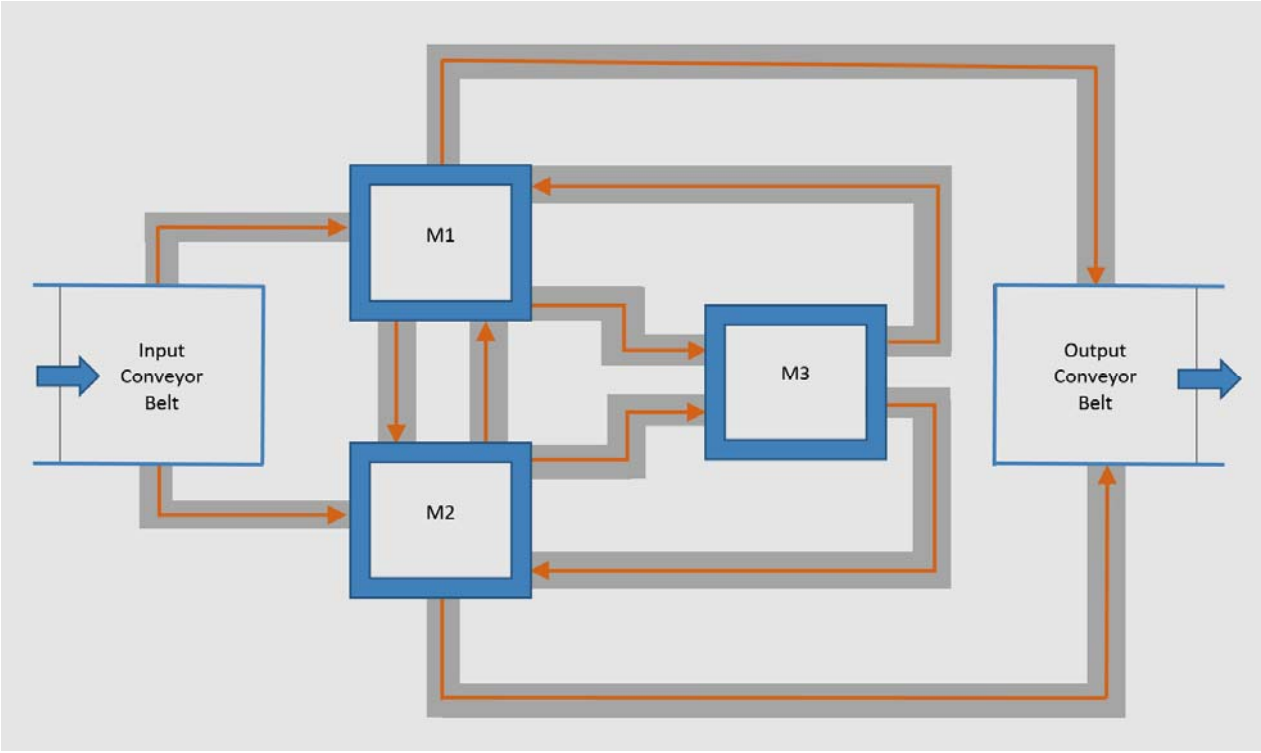


Figure 4. The layour of the Flexible Manufacturing System.

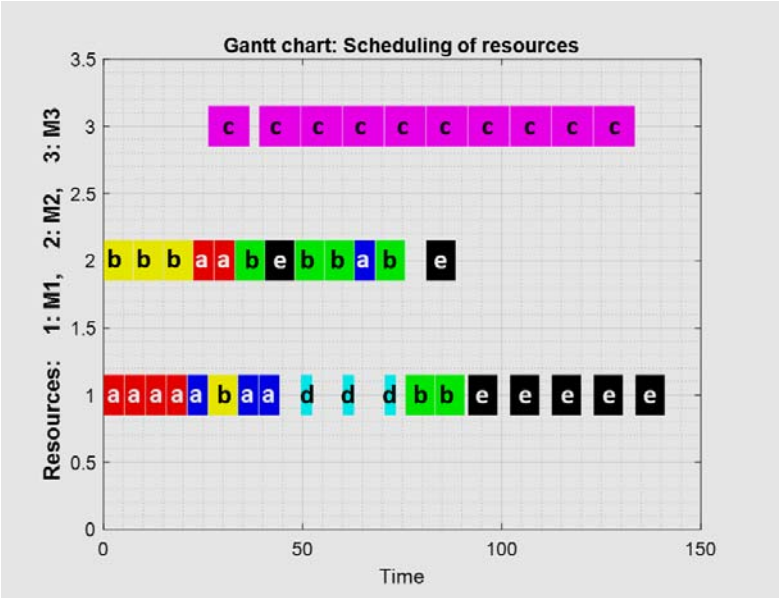
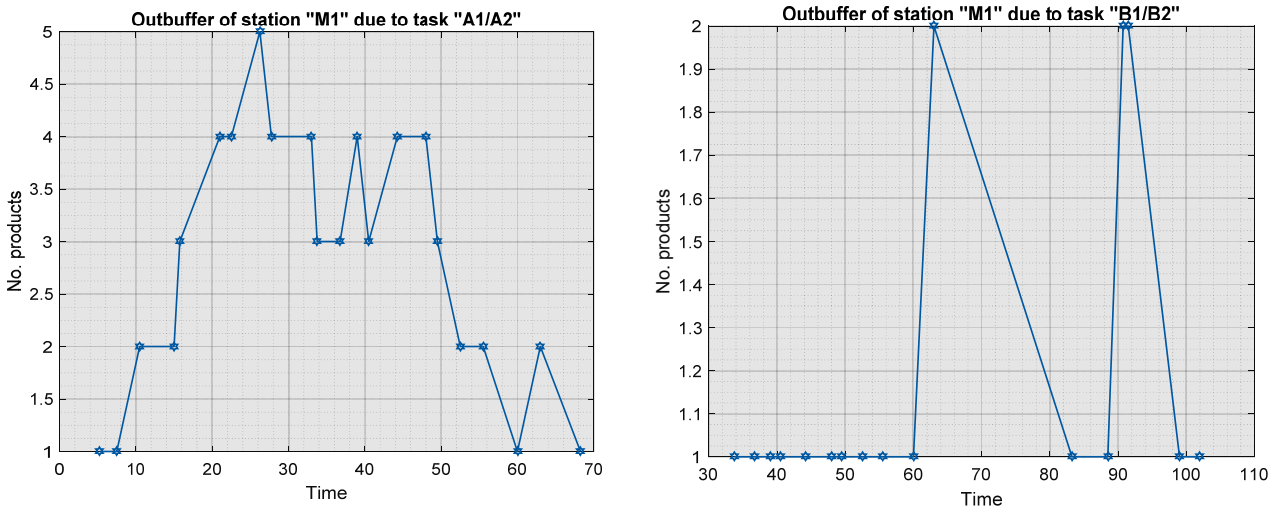


Figure 5. The maximum products in the output buffers of the stations due to different tasks.



6a. Number of products in the output buffer of M1 that were made by task A. 6b. Number of products in the output buffer of M1 that were made by task A

Figure 6. The maximum products in the output buffers of the stations due to different tasks.