Addendum-3 to the book **Davidrajuh, R. "Modeling Discrete-Event Systems with GPenSIM: An Introduction", Springer, 2018**

Cost Calculations in Petri Nets

Measuring the costs of activities and the resource usage costs

---

By Reggie Davidrajuh, reggie.davidrajuh@uis.no, © 14 August 2018.

No simulator for discrete-event systems is complete unless it provides some in-built support for cost calculations. This is because, in engineering and industry, we do study about the performance of discrete-event systems to save time and resources which ultimately results in saving money (costs). GPenSIM provides a simple and effective mechanism for cost calculation in simulations of Petri Net models.

1. Measuring the Costs of Activities

In GPenSIM, only transitions are active, and only the activities (firing of transitions) can cost money. Places are passive, and their sole purpose is to keep tokens. In GPenSIM, places and their storage of tokens cannot cost money. (However, resource usage cost money too, as explained in section-2.)

When a transition fires:

- At the start of firing, the transition consumes input tokens: the cost of the firing (***Cost_of_Firing***) at this stage is the summation of individual costs of all the input tokens; this means, every token carries the costs with it. Let us denote the summation of individual costs of all the input tokens as ***Total_Cost_of_Input_Tokens***

  Again, at the start of firing: ***Cost_of_Firing = Total_Cost_of_Input_Tokens***
- At the end of firing, the cost of the firing becomes more as the firing cost of the transition has to be added to the total cost. However, the **firing cost** of the transition is two folded: 1) fixed cost **(*fc_fixed*)**, and variable cost (***fc_variable***). Hence, the total cost at the end of firing becomes:

  Cost_of_Firing = Total_Cost_of_Input_Tokens + fc_fixed + (firing_time * fc_variable)

- When the firing completes, the input tokens are transformed into some output tokens that will be deposited into the output places. The cost of firing will be equally distributed to the output tokens. Thus, cost of each output token becomes:

  Cost_of_each_output_token = Cost_of_Firing / number_of_output_tokens

In addition to transitions, resource usage will also cost money. In the next section-2, we shall study how to compute costs induced by transitions without using resources. Davidrajuh (2012 and 2013) presents some limited cost calculations based on the extensions of the reachability tree; reachability tree cannot include resource usage costs.

1.1 Firing Costs of Transition

As explained above, a transition can have up to two firing costs:

1. **Fixed cost (*fc_fixed*)**: this is one time cost, as the transition fires, this cost will be added to the total firing cost.
2. **Variable cost (*fc_variable*)**: this is the firing cost per a unit of time. Thus, this cost will be multiplied with firing time before added to the total firing cost.

The firing costs are declared in the MSF, as a part of initial dynamics. For example,

```
% first costs analysis
…
…
pns = pnstruct('abc3_pdf');

dyn.m0 = {'p1',4, 'p2',3, 'p3',5};
dyn.ft = {'t1',2, 't2',5,'allothers',1};
dyn.fc_fixed = {'t1',50, 't2',100};
dyn.fc_variable = {'t1',20, 't1',25};
pni = initialdynamics(pns, dyn);

sim = gpensim(pni);
```

In the code snippet shown above, the fixed firing cost of **t1** is 50 **currency units** (**CU**). The variable firing cost of **t1** is 20 CU per unit of firing time. Whereas the fixed firing cost of **t2** is 100 CU, and the variable firing cost of **t2** is 20 CU (per unit of firing time).

1.1.1 Example-57: Firing Costs of a Transition

Figure-1-1 shows a Petri Net with four transitions.

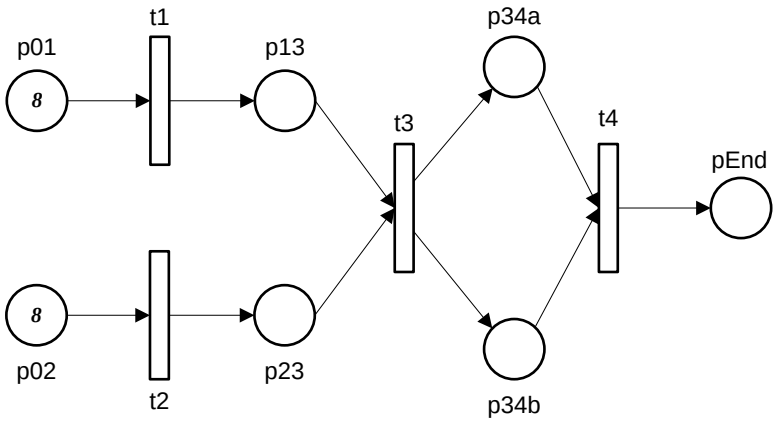

Figure 1-1: Petri Net for simple cost calculation.

The firing times and the firing costs of the transitions are given below:

Table 1-1: Properties of the transitions

Transition	Firing time (ft)	Firing Cost	
		Fixed (*fc_fixed*)	Varbiable (*fc_variable*)
t1	10	3	5
t2	10	250	50
t3	10	5000	500
t4	10	10	10

Also, let us limit the number of firings of the transitions:

- transition **t1** and **t2** will be allowed to fire only six times (so that they will leave two initial tokens each in **p01** and **p02**),
- **t3** will be allowed to fire only four times (so that **t3** will leave two tokens each in **p13** and **p23**), and
- **t4** will be allowed to fire only two times (so that **t4** will leave two tokens each in **p34a** and **p34b**).

At the end of the simulation, there will be two tokens each in all the places, as shown in figure-1-2:

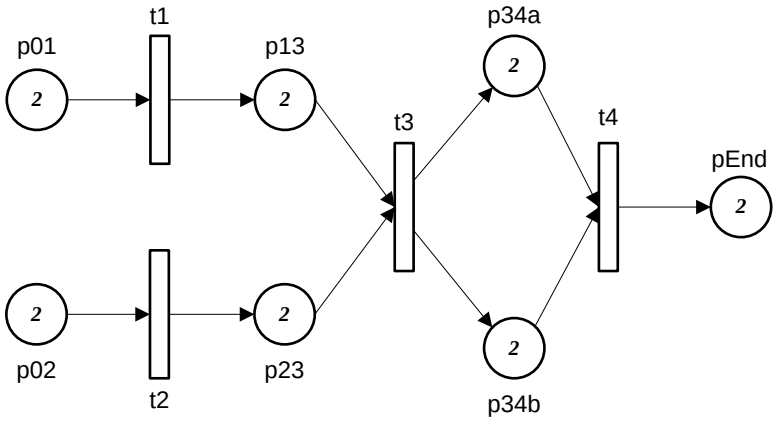

Figure 1-2: At the end of the simulation run, two tokens will be left left in each place.

Let us inspect the cost of each token, with the help of the function **`prnfinalcolors`**:

```
**** **** Colors of Final Tokens ...
No. of final tokens: 14

Place: p01
Time: 0 *** NO COLOR ***
Time: 0 *** NO COLOR ***

Place: p02
Time: 0 *** NO COLOR ***
Time: 0 *** NO COLOR ***

Place: p13
Time: 50 *** NO COLOR ***   Cost: 53
Time: 60 *** NO COLOR ***   Cost: 53

Place: p23
```

```
Time: 50 *** NO COLOR ***   Cost: 750
Time: 60 *** NO COLOR ***   Cost: 750

Place: p34a
Time: 40 *** NO COLOR ***   Cost: 5401.5
Time: 50 *** NO COLOR ***   Cost: 5401.5

Place: p34b
Time: 40 *** NO COLOR ***   Cost: 5401.5
Time: 50 *** NO COLOR ***   Cost: 5401.5

Place: pEnd
Time: 30 *** NO COLOR ***   Cost: 10913
Time: 40 *** NO COLOR ***   Cost: 10913
>>
```

After simulation run, there are two tokens left in each place.

- The tokens in places **p01** and **p02** are initial tokens, thus have zero costs.
- The tokens in **p13** have a cost of 53. This is correct as, for **t1**,
  Total_Cost_of_Input_Tokens + ***fc_fixed*** + ***(fc_variable * ft)*** = 0 + 3 + 5*10 = 53
- The tokens in **p23** have a cost of 750. This is also correct as, for **t2**,
  Total_Cost_of_Input_Tokens + ***fc_fixed*** + ***(fc_variable * ft)*** = 0 + 250 + 50*10 = 750
- The tokens in **p34a** and **p34b** have a cost of 5401.5. This is correct as for **t3**,
  Cost_of_Firing = ***Total_Cost_of_Input_Tokens*** + ***fc_fixed*** + ***(fc_variable * ft)***
  = (53+750) + 5000 + 500*10 = 10803. This cost must be equally divided between the two output tokens. Thus, cost of a token in **p34a** or **p34b** is = 10803/2 = 5401.5
- The tokens in **pEnd** have a cost of 10913. This is because, for **t4**,
  Cost_of_Firing = ***Total_Cost_of_Input_Tokens*** + ***fc_fixed*** + ***(fc_variable * ft)***
  = (2*5401.5) + 10 + 10*10 = 10913.

MSF:

```
% Example-57: Firing Costs of Transition
global global_info;
global_info.STOP_AT = 80;

pns = pnstruct('firing_costs_pdf');

dyn.m0 = {'p01',8, 'p02',8}; % tokens initially
dyn.ft = {'allothers',10};
dyn.fc_fixed = {'t1',3, 't2',250, 't3',5000, 't4',10};
dyn.fc_variable = {'t1',5, 't2',50, 't3',500, 't4',10};

pni = initialdynamics(pns, dyn);
sim = gpensim(pni);
prnfinalcolors(sim);
```

PDF:

```
% Example-57: Firing Costs of Transition
function [png] = firing_costs_pdf()
png.PN_name = 'Simple cost calculation';
png.set_of_Ps = {'p01', 'p02','p13', 'p23', 'p34a', 'p34b', 'pEnd'};
png.set_of_Ts = {'t1', 't2', 't3', 't4'};
png.set_of_As = {'p01','t1',1, 't1','p13',1, ... % t1
    'p02','t2',1, 't2','p23',1, ... % t2
```

```
    'p13','t3',1, 'p23','t3',1, 't3','p34a',1, 't3','p34b',1, ... % T3
    'p34a','t4',1, 'p34b','t4',1, 't4','pEnd',1};
```

COMMON_PRE: For limiting the number of firings of the transitions:

```
% Example-57: AOPN model of a Flexible Manufacturing System
function [fire, transition] = COMMON_PRE(transition)

% get the number of times the trans has fired
n = timesfired(transition.name);

fire = 1;   % initial assumption
switch transition.name
    case {'t1', 't2'}
        if eq(n, 6)  % t1 and t2 are allowed to fire only 6 times
            fire = 0;
        end
    case 't3'
        if eq(n, 4)  % t3 is allowed to fire only 4 times
            fire = 0;
        end
    case 't4'
        if eq(n, 2)  % t4 is allowed to fire only 2 times
            fire = 0;
        end
end
```

1.1.2 Example-58: Repeated Firings of a Transition

This example is yet another simple example, just to experiment with the accumulated costs when transitions fire repeatedly. The model shown in figure-1-3 depicts that the only transition **t1** will fire three times exactly. We will study how the costs of the tokens evolve with time.

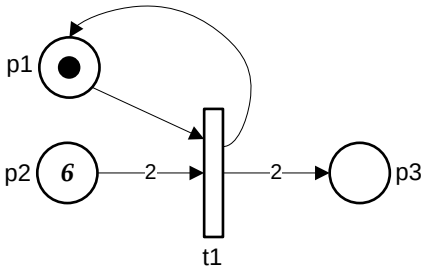

Figure 1-3: Iterative firing.

Let the fixed costs of **t1** be 9 CU, the variable cost be 18 CU per unit of firing time, and the firing time is one TU. The simulation result is shown below:

```
**** **** Colors of Final Tokens ...
No. of final tokens: 7

Place: p1
Time: 3 *** NO COLOR ***   Cost: 13
```

```
Place: p3
Time: 1 *** NO COLOR ***   Cost: 9
Time: 1 *** NO COLOR ***   Cost: 9
Time: 2 *** NO COLOR ***   Cost: 12
Time: 2 *** NO COLOR ***   Cost: 12
Time: 3 *** NO COLOR ***   Cost: 13
Time: 3 *** NO COLOR ***   Cost: 13
>>
```

The simulation result shows that the tokens in **p3** have cost from 9 to 13. Let us verify:

- Initially, **p1** has one token, and **p2** has 6 tokens. The costs of all these initial tokens are zero.
- When **t1** fires for the first time, it takes one token from **p1** and two from **p2**. ***Total_Cost_of_Input_Tokenens = 0.*** The firing of **t1** incurs the following costs:
  Cost_of_Firing = Total_Cost_of_Input_Tokens + fc_fixed + (fc_variable * ft)
  = 0 + 9 + 1*18 = 27
  This cost has to divided equally between 3 output tokens, thus cost of an output token (one into **p1** and two into **p3**) becomes **9**.
- When **t1** fires for the second time, it again takes one token from **p1** (cost is 9) and two from **p2** (cost is 0). Thus, ***Total_Cost_of_Input_Tokens*** is 9. Firing of **t1** costs the following:
  Cost_of_Firing = Total_Cost_of_Input_Tokens + fc_fixed + (fc_variable * ft)
  = 9 + 9 + 1*18 = 36
  This cost has to be divided equally between 3 output tokens, thus cost of an output token (one into **p1** and two into **p3**) becomes **12**.
- When **t1** fires for the third time, it again takes one token from **p1** (cost is 12) and two from **p2** (cost is 0). Thus, ***Total_Cost_of_Input_Tokenens*** is 12. Firing of **t1** costs the following:
  Cost_of_Firing = Total_Cost_of_Input_Tokenens + fc_fixed + (fc_variable * ft)
  = 12 + 9 + 1*18 = 39
  This cost has to be divided equally between 3 output tokens, thus cost of an output token (one into **p1** and two into **p3**) becomes **13**.

Thus, we find six tokens in **p3**, two with a cost of 9 CU, another two with 12 CU, and the final two with 13 CU.

MSF:

```
% Example-58: Itertaive Firing Costs of Transition
global global_info
global_info.STOP_AT = 10;

pns = pnstruct('iterative_fc_pdf');

dyn.m0 = {'p1',1, 'p2',6}; % tokens initially
dyn.ft = {'allothers',1};
dyn.fc_fixed = {'t1',9};
dyn.fc_variable = {'t1',18};

pni = initialdynamics(pns, dyn);
sim = gpensim(pni);
prnfinalcolors(sim);
```

1.2 Token Selection based on Cost

As we have seen in the previous subsection, costs are generated by transitions, and it is the tokens that carry the costs around the net as they flow around. The table below shows the select functions that deal with the cost of a token in a specific place:

Table 1-2: GPenSIM functions for selection of tokens based on cost

Function	*Description*
`tokenCheap`	Select tokens that are the cheapest in a specific place.
`tokenCostBetween`	Select tokens that cost between two limits.
`tokenExpensive`	Select tokens that are the most expensive in a specific place.

A transition may select input tokens based on cost. Selection can be done by three ways:

- Tokens that are the cheapest in the place,
  [set_of_tokID, nr_token_av] = tokenCheap(place, nr_tokens_wanted)

- Tokens that are the most expensive in the place,
  [set_of_tokID, nr_token_av] = tokenExpensive(place, nr_tokens_wanted)

- Tokens that cost between two limits lower cost (lc) and upper cost (uc).
  [set_of_tokID, nr_token_av] = tokenCostBetween(place, nr_tokens_wanted, lc, uc)

The output parameters of the functions are a set of IDs of the selected tokens (set of tokID 'set_of_tokID') and the actual number of valid tokID in the set ('nr_token_av').

E.g., if a transition wants **4** **cheapest** tokens from the input place **pBUFF**, then the transition will execute the following statement:

```
function [fire, transition] = tLR_A_pre (transition)
selected_tokens = tokenCheap('pBUFF',4);
fire = all(selected_tokens);
```

If **pBUFF** has more than or equal to four tokens, then the returned value **`selected_tokens`** will have tokIDs of the four cheapest tokens. Otherwise (if **pBUFF** has less than 4 tokens), then **`selected_tokens`** will have tokIDs of all the tokens, padded with 0s.

E.g., if a transition wants **4** **most expensive** tokens from the input place **pBUFF**, where **pBUFF** has only two tokens at that time:

```
function [fire, transition] = tLR_A_pre (transition)
selected_tokens = tokenExpensive('pBUFF',4);
fire = 1;
```

Composition of **`selected_tokens`** will be [tokIDi tokIDj 0 0], where tokIDi and tokIDj are valid tokIDs.

1.2.1 Example-59: Token selection based on cost

Figure-1-4 depicts a simple Petri Net where **tAddCost** is allowed to fire ten times, depositing ten tokens into **pBuffer**. Once **tAddCost** has completed ten firings, **tDelay** fires, depositing ten enabling tokens into **pReady**. These enabling tokens in **pReady** enable the other three transitions (**tChp**, **tMid**, and **tExp**) to start filtering the tokens in **pBuffer**. Filtering means, **tChp** takes the three cheapest tokens from **pBuffer**, **tExp** takes the three most expensive tokens, and **tMid** takes tokens that cost in between.

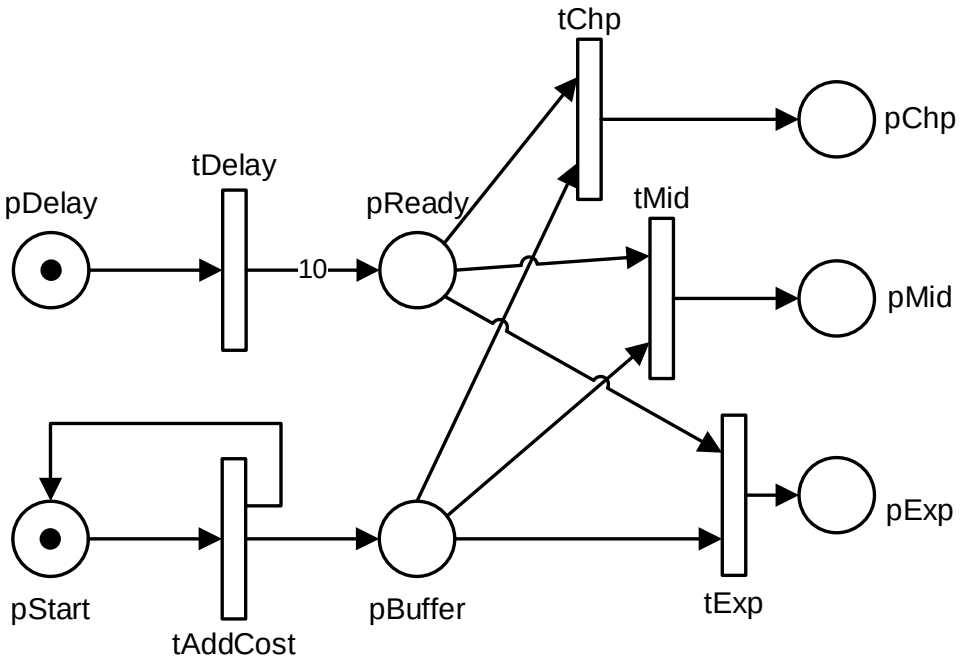

Figure 1-4: Token selection based on cost.

Let us say that the three filtering transitions **tChp**, **tMid**, and **tExp** do not incur any firing cost (fixed and variable firing costs of these transitions are zero). Thus, filtering incurs no additional costs. The tokens that end up in places **pBuffer**, **pChp**, **pMid**, and **pExp**, have costs that are incurred by the firings of **tAddCost** alone. **tAddCost** has a fixed and variable firing costs of 200 and 100 CU, respectively. The firing time of **tAddCost** is 1 TU.

When **tAddCost** fires for the first time, the input token cost nothing, thus the cost of output tokens deposited into **pBuffer** and **pStart** become (200+100)/2 = 300/2. When **tBuffer** fires for the second time, it takes the token from **pStart** (cost 300/2), and add firing cost (200+100 = 300), thus the total cost becomes (300/2 + 300), which has to be divided by 2 as there are two output tokens. Subsequently, the cost of ouput token deposited into **pBuffer** become = 300/2, 300(1/2 + ¼), 300 (1/2 + ¼ + 1/8) + …. and so on.

The simulation results show that **pChp** has the three cheapest tokens, **pMid** has tokens that cost between 270 and 290 CU, and **pExp** has the three most expensive tokens. The rest of the token is left in **pBuffer**.

```
**** **** Colors of Final Tokens ...
No. of final tokens: 14

Place: pBuffer
Time: 5 *** NO COLOR ***   Cost: 290.625
Time: 6 *** NO COLOR ***   Cost: 295.3125
Time: 7 *** NO COLOR ***   Cost: 297.6563

Place: pChp
Time: 12 *** NO COLOR ***   Cost: 150
Time: 13 *** NO COLOR ***   Cost: 225
```

```
Time: 14 *** NO COLOR ***   Cost: 262.5

Place: pExp
Time: 12 *** NO COLOR ***   Cost: 299.707
Time: 13 *** NO COLOR ***   Cost: 299.4141
Time: 14 *** NO COLOR ***   Cost: 298.8281

Place: pMid
Time: 12 *** NO COLOR ***   Cost: 281.25

Place: pReady
Time: 11 *** NO COLOR ***
Time: 11 *** NO COLOR ***
Time: 11 *** NO COLOR ***

Place: pStart
Time: 10 *** NO COLOR ***   Cost: 299.707
>>
```

MSF:

```
% Example-59: token selection based on Costs
global global_info
global_info.STOP_AT = 100;

pns = pnstruct('cost_selection_pdf');
dyn.m0 = {'pStart',1, 'pDelay',1};
dyn.ft = {'allothers',1};

dyn.fc_fixed = {'tAddCost', 200};
dyn.fc_variable = {'tAddCost', 100};

pni = initialdynamics(pns, dyn);
sim = gpensim(pni);
prnfinalcolors(sim);
```

PDF:

```
% Example-59: token selection based on Costs
function [png] = cost_selection_pdf()
png.PN_name = 'Ex-59: token selection based on Costs';
png.set_of_Ps = {'pStart','pDelay','pBuffer','pReady', ...
                 'pChp', 'pMid','pExp'};
png.set_of_Ts = {'tAddCost','tDelay', 'tChp', 'tMid','tExp'};
png.set_of_As = {'pStart','tAddCost',1,'tAddCost','pStart',1,...
    'tAddCost','pBuffer',1, ... % tAddCost
    'pDelay','tDelay',1, 'tDelay','pReady',10,...     % tDelay
    'pBuffer','tChp',1,'pReady','tChp',1,'tChp','pChp',1,... % tChp
    'pBuffer','tMid',1,'pReady','tMid',1, 'tMid','pMid',1,... % tMid
    'pBuffer','tExp',1,'pReady','tExp',1,'tExp','pExp',1, ... % tExp
                };
```

COMMON_PRE: this Pre-processor performs the operations:

- stops **tAddCost** after ten firing,

- allows **tDelay** to fire right after the ten firings of **tAddCost**,
- allows the filtering transitions to fire three times only: **tChp**, **tMid**, and **tExp**,
- allows **tChp** to select the cheapest,
- allows **tExp** to select the most expensive, and
- allows **tMid** to select token cost between 270 and 290 CU.

```
function [fire, transition] = COMMON_PRE(transition)

timesFired = timesfired(transition.name);
switch transition.name
    case 'tAddCost'
        fire = lt(timesFired, 10); return
    case 'tDelay' % tDelay fires after 10 firings of tAddCost
        timesfired_tAddCost = timesfired('tAddCost');
        fire = eq(timesfired_tAddCost,10); return
    case {'tChp','tMid','tExp'}
        % allow 'tChp', 'tMid', and 'tExp' to fire ONLY 3 times
        if eq(timesFired, 3), fire = 0; return; end

        % we are here for the three transitions tChp,tMid,tExp
        if strcmp(transition.name, 'tChp') % tChp selects cheap
            tokID = tokenCheap('pBuffer',1);
        elseif strcmp(transition.name,'tExp')%tExp wants expensive
            tokID = tokenExpensive('pBuffer', 1);
        else % 'tMid' selects tokens that cost between (270, 290)
            tokID = tokenCostBetween('pBuffer', 1, 270, 290);
        end % switch transition.name
        transition.selected_tokens = tokID;
        fire = tokID;
end
```

2. Measuring the Costs of Resource Usage

An activity may be very lightweight as its firing costs are negligible. However, this activity may use an expensive resource to perform. In this case, the total costs of the activity (transition costs + the resource usage costs) will be dominated by the resources used alone. Hence, in this section, we look into the ways to calculate the resource costs of discrete-event systems. GPenSIM provides a simple and effective mechanism for calculating the costs of resource usage.

2.1 The costs of resource usage

In the previous section, we studied the costs due to transition firing. However, if the transitions use resources, then resource cost will also come into the cost calculations, in addition to the transition firing costs. Davidrajuh (2012 and 2013) presents some limited cost calculations based on the extensions of the reachability tree. However, reachability tree cannot include resource usage costs. This section discusses how we can include the cost of using resources.

Just like transition firing costs, each resource instance can also incur two types of costs:

1. **Fixed cost (*rc_fixed*)**: this is one-time cost as each time a transition start using a resource, this cost will be added to the total cost.
2. **Variable cost (*rc_variable*)**: this cost is per unit of time the resource is used. Thus, this cost will be multiplied by the firing time before added to the total resource cost.

The resource costs are declared in the MSF, as a part of initial dynamics. For example:

```
% first costs analysis
…
…

dyn.ft = {'t1',2, 't2',5,'allothers',1};
dyn.fc_fixed = {'t1',50, 't2',100};
dyn.fc_variable = {'t1',20, 't1',25};

dyn.re = {'R1',1,inf, 'R2',1,inf};
dyn.rc_fixed = {'R1',750, 'R2',10};
dyn.rc_variable = {'R1',300, 'R2',250};

pni = initialdynamics(pns, dyn);

sim = gpensim(pni);
```

In the code snippet shown above, there are two resources (‘**R1**’ and ‘**R2**’) and each resource has only one instance. For **R1**, the fixed cost is 750 CU, and the variable cost is 10 CU per unit of firing time. Whereas for **R2**, the fixed cost is 300 CU, and the variable cost is 250 CU (per unit of firing time).

2.1.1 Example-60: Machine and Resource Usage Costs

In this example, we shall work on a simple problem of finding the total cost of production when machines and resources add costs to the production. Davidrajuh (2014) presents a crude version of this problem.

A commercial printing facility is planning to buy a new printing machine. There are two alternatives to consider as two different types of printers can be used for the same job:

- Printer-A is faster and seems to cost more.
- Printer-B is slower, but it seems it is also cheaper to operate.
- These two printers also use a different amount of resources (inks).

The fact sheet from the manufactures of the printers is given below in the table-2-1.

Table 7-1: Properties of the printers for a typical job.

	Production time in TU	Production Cost in CU	
		Fixed	Varbiable
Printer-A	10	1000	100
Printer-B	20	750	75

During the printing operations, the two printers use different types of inks:

- Printer-A uses one tube of ink “Premia” and one tube of ink “Rainbow.”
- Printer-B uses two tubes of “Rainbow.”

The table-2-2 shows the properties of the inks.

Table 7-2: Properties of the inks.

Resource type	Number of tubes	Resource Cost in CU	
		Fixed	Varbiable
Ink “Premia”	1	1000	100
Ink “Rainbow”	3	800	80

Which printer to choose?

A quick look into the table-1-1 reveals that Printer-A is two times faster than Printer-B, but Printer-A is also more expensive than Printer-B to operate. Also, Printer-A uses one tube of Premium ink which is more expensive than the Rainbow ink. Hence, it seems that the choice is between fast and expensive Printer-A or slow and cheap Printer-B. However, as the company wants numerical answers, we are going to make a Petri Net model of the problem and then run simulations to get numerical answers.

The Model:

Figure-2-1 shows the Petri Net model of the production facility where both printing machines are employed to run one sample job (test job) each. In the model, the resource usages of the two machines are also indicated.

PDF:

```
% Example-60: resource Usage Cost
function [png] = ruc_pdf()
png.PN_name = 'Ex-60: Resource Usage Cost';
png.set_of_Ps = {'pInBUFF', 'pA','pB'};
png.set_of_Ts = {'tPrint_A', 'tPrint_B'};
png.set_of_As = {'pInBUFF','tPrint_A',1,'tPrint_A','pA',1,...% prn-A
            'pInBUFF','tPrint_B',1, 'tPrint_B','pB',1};    % prn-B
```

MSF:

```
% Example-60: resource Usage Cost
global global_info
```

```
global_info.STOP_AT = 200;

pns = pnstruct('ruc_pdf');
dyn.m0 = {'pInBUFF', 2}; % two test samples
dyn.ft = {'tPrint_A',10, 'tPrint_B',20};
dyn.fc_fixed =  {'tPrint_A',1000, 'tPrint_B',750};
dyn.fc_variable = {'tPrint_A',100, 'tPrint_B',75};

dyn.re = {'Premia',1,inf, 'Rainbow',3,inf};
dyn.rc_fixed = {'Premia',1000, 'Rainbow',800};
dyn.rc_variable = {'Premia',100, 'Rainbow',80};

pni = initialdynamics(pns, dyn);
sim = gpensim(pni);
prnfinalcolors(sim);
prnschedule(sim);
```

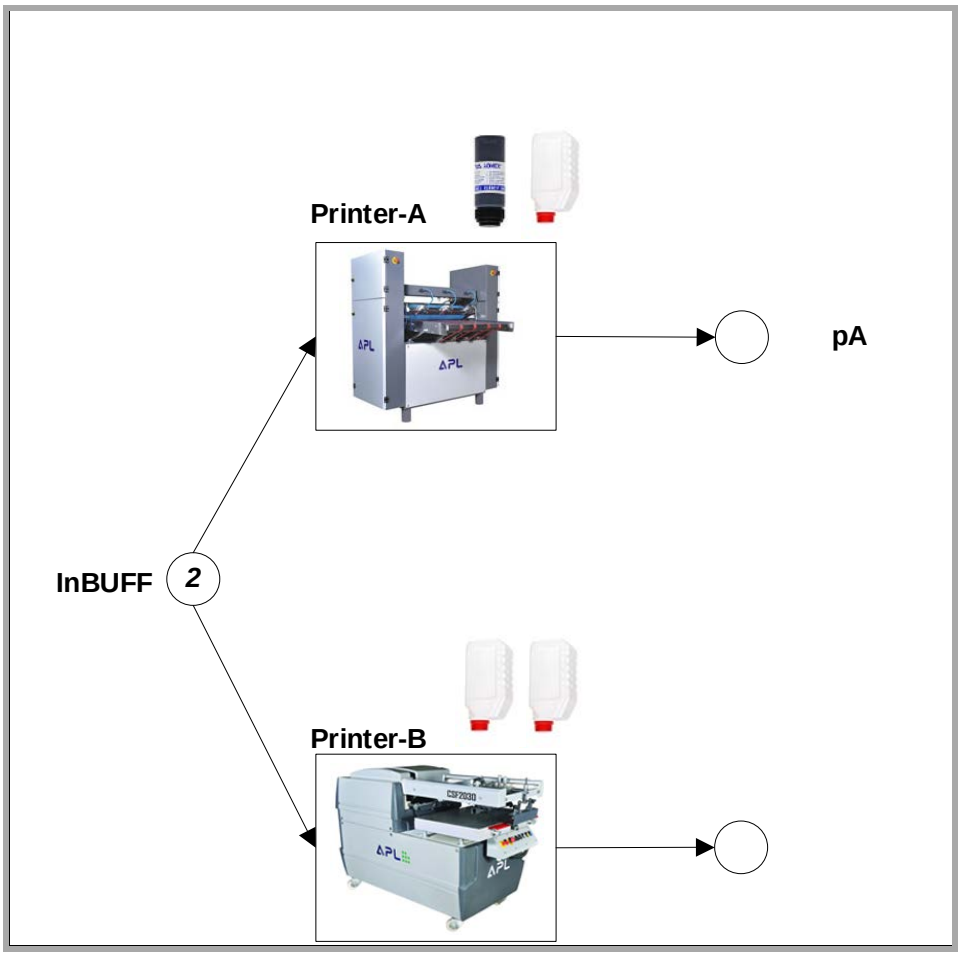

Figure 2-1: Petri Net model of the production facility to calculate production costs.

COMMON_PRE:
In COMMON_PRE, we let **tPrint_A** (Printer-A) and **tPrint_B** (Printer-B) fire only once. Also, **tPrint_A** requests once instance of each Premia and Rainbow, whereas **tPrint_B** requests two instances of Rainbow:

```
function [fire, transition] = COMMON_PRE(transition)

% tPrint_A and tPrint_B can fire only once
if timesfired(transition.name) % quit, if already fired
    fire = 0; return
end

if strcmpi(transition.name, 'tPrint_A')
    % tA needs one tube of 'Premia' and one tube og 'Rainbow'
    granted = requestSR({'Premia',1, 'Rainbow',1});

elseif strcmpi(transition.name, 'tPrint_B')
    % tB needs two tubes of 'Rainbow'
    granted = requestSR({'Rainbow',2});
end
fire = granted;
```

COMMON_POST:
As usual, the main use of COMMON_POST is to release the resources used by a fired transition. Also, COMMON_POST can be used for another important function: when the number of tokens in the places **pA** and **pB** become one each, it is better to stop the simulations; otherwise, simulation runs unnecessarily longer, up to the global_info.STOP_AT time, resulting in lower value for the efficiency of resource usage.

```
function [] = COMMON_POST(transition)
global global_info

release;  % release the resources used by the fired transition

% if pA and pB has one token each,
% then stop simulations immediately
pA = ntokens('pA');
pB = ntokens('pB');

if eq(pA+pB, 2)
    global_info.STOP_SIMULATION = 1;
end
```

Simulation results:
'**prnfinalcolors**' gives the following result, which clearly shows that using Printer-A is not only faster, but it is also cheaper. Printer-B is slower and costs more to produce. A unit sample produced by Printer-A costs only 5600 CU, whereas the costs of producing one sample by Printer-B is 7050 CU. The cost of production by Printer-B is higher because it uses the resources longer, incurring larger variable costs.

```
**** **** Colors of Final Tokens ...
No. of final tokens: 2

Place: pA
Time: 10 *** NO COLOR ***   Cost: 5600
```

```
Place: pB
Time: 20 *** NO COLOR ***   Cost: 7050
```

Since there are no other tokens anywhere, the total cost of production is the total costs of the tokens in **pA** and **pB**: 5600 + 7050 = 12650 CU. The total production cost (12650 CU) is confirmed by the print out of **`prnschedule`**:

```
RESOURCE USAGE SUMMARY:
Premia:  Total occasions: 1   Total Time spent: 10
Premia:  Total resource usage cost: 2000
Rainbow:  Total occasions: 3   Total Time spent: 50
Rainbow:  Total resource usage cost: 6400

*****  LINE EFFICIENCY AND COST CALCULATIONS: *****
  Number of servers:  k = 2
  Total number of server instances:  K = 4
  Completion = 20
  LT = 80
  Total time at Stations: 60
  LE = 75 %
  **
  Sum resource usage costs: 8400   (66.4032% of total)
  Sum firing costs: 4250   (33.5968% of total)
  Total costs:  12650
  **
```

The print out is shown above, while confirming the total production cost (12650), it also states that the total cost can be divided into 8400 CU as resource usage cost and 4250 CU as machining (firing) costs. This is because:

- **tPrint_A** fired once costing 1000 + 10 * 100 = 2000 CU
- **tPrint_B** fired once costing 750 + 20 * 75 = 2250 CU
- Thus, total machining costs (firing costs) is 4250 CU

From the resource usage information given below, it is clear that:

- One instance of Premium ink is used in one firing (by **tPrint_A**) for a total time of 10 TU, amounting to costs: (1000 + 10 * 100) = 2000 CU
- Three instances of Rainbow ink is used (one instance by **tPrint_A** and two instances by **tPrint_B**), for a total time of 50 TU, amounting to costs: (3 * 800 + 50 * 80) = 6400 CU.
- Thus, total resource costs are 8400 CU!

```
RESOURCE USAGE:

RESOURCE INSTANCES OF ***** Premia *****
tPrint_A [0 : 10]
Resource Instance: Premia:: Used 1 times. Utilization time: 10

RESOURCE INSTANCES OF ***** Rainbow *****
(Rainbow-1):   tPrint_A [0 : 10]
(Rainbow-2):   tPrint_B [0 : 20]
(Rainbow-3):   tPrint_B [0 : 20]
Resource Instance: (Rainbow-1):: Used 1 times.  Utilization time: 10
Resource Instance: (Rainbow-2):: Used 1 times.  Utilization time: 20
Resource Instance: (Rainbow-3):: Used 1 times.  Utilization time: 20
```

```
RESOURCE USAGE SUMMARY:
Premia:  Total occasions: 1   Total Time spent: 10
Premia:  Total resource usage cost: 2000
Rainbow:  Total occasions: 3   Total Time spent: 50
Rainbow:  Total resource usage cost: 6400
```

References

- R. Davidrajuh, “Activity Based Costing with Petri Nets,” 2012 UKSim-AMSS 6th European Modelling Symposium, 2012, DOI 10.1109/EMS.2012.64
- R. Davidrajuh, “Extended Reachability Graph of Petri Net for Cost Estimation,” 2013 8th EUROSIM Congress on Modelling and Simulation, 2013, DOI 10.1109/EUROSIM.2013.72
- R. Davidrajuh, “Efficient data structures for a new Petri Net based simulator”, IEEE European Modelling Symposium (EMS), Pisa, Italy, October 21-23, 2014.