THROUGHUT ANALYSIS OF DISCRETE EVENT SYSTEMS BASED ON STOCHASTIC PETRI NETS*

JIACUN WANG† , MENGCHU ZHOU†† AND YI DENG†

†*School of Computer Science, Florida International University*

Miami, FL33199, Email: (wangji, deng)@cs.fiu.edu

††*Department of Electrical and Computer Engineering, New Jersey Institute of Technology*

Newark, NJ07102, Email: zhou@ece.njit.edu

Received (February 1999)

Revised (June 1999)

This paper presents an algorithm of polynomial complexity to derive the throughput of a discrete event system via stochastic Petri net (SPN) models. The concept of flow nets, a subclass of SPN, is introduced to model a class of discrete event systems. The mathematical model for the throughput of flow nets is given. For a structurally non-competitive and acyclic flow net, the solution algorithm proceeds in four steps. First, divide the places and transitions into groups according to some rules. Next, list the flow equilibrium equation for each place, which shows the relation among the average flows of its input and output transitions. Then, deduce the relation of the average flow of any non-source transition to that of all source transitions. Finally, determine the throughput of the model. By so doing, we significantly reduce the size of the linear programming problems. For a structurally competitive and cyclic flow net, a procedure is proposed to convert it to a structurally non-competitive and acyclic one in the sense of equivalent throughput. Besides, the paper also shows how to transform an SPN with shared resource to a flow net. An assembly system is used to illustrate the application of the technique for the analysis of throughput.

Keywords: Stochastic Petri nets, throughputs, discrete event systems.

1. Introduction

Petri nets have been used to model various discrete event systems.[1,2,3,4,5] Because of their ease to model such features as parallelism, contention and synchronization, they have gained more and more applications. Basic Petri nets lack a temporal description and therefore fail to represent any performance measures. Introduction of time into a transition or a place increased both the modeling power and the complexity of the net analysis. When exponentially distributed times are associated with transitions, the extended Petri nets are called *Stochastic Petri Nets* (SPN's). It has been proved that SPN's are isomorphic to Markovian processes. [4,6,7]

In general, the solution techniques for an SPN rely on simulation of the net operation or enumeration of all reachable states, thus the state explosion problems are quickly encountered as the net size or the number of initial tokens representing, e.g., jobs and

* This work was supported in part by the Army Research Office under grant No. DAAG55-98-1-0428, by the NSF under grant No. HDR-9707076, and by the Air Force Office of Scientific Research under grant No. F49620-96-1-0221. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official polices or endorsements either expressed or implied by the above named Agencies.

resources, in the places increases. This has been a major drawback and has limited the application of such models for practical system specification, modeling and analysis.

SPN models of real systems often have tokens representing jobs, users, processors, processes, or stations in a system.[8,9,10] Knowing the maximum rate at which such entities can cycle through the system is a critical measure of the system's performance. A fast (with polynomial complexity) determination of a tight upper bound on the throughput would significantly increase the usability and applicability of the SPN model.[11]

Some work on determining the maximum throughput of an SPN has been done. In the work by Florin and Natkin[4] a special network structure was assumed. This structure limited the action of some sub-network by a limit place, much as the method of stages or the single server model uses a control place with a single token to limit the number of tokens in the subnet to one. In their model, the transitions that start the subnet are enabled by a special control place that limits the number of times the transitions can fire. Thus, until the tokens go through the subnet and a token is placed back into the control place, the input to the subnet is blocked. This is then used to show that such nets have a maximum average throughput.

Campos et al.[12,13] analyzed the ergodicity and characterized the throughput bound for a subclass of timed and stochastic Petri nets, interleaving qualitative and quantitative theories. The considered nets represent an extension to marked graphs, and are defined as nets having a unique consistent firing count vector, independently of the stochastic interpretation of the net models. Upper and lower throughput bounds are computed using linear programming. The bounds depend on the initial marking and the mean values of the delays but not on the probability distribution.

All the previously published work deriving the performance bounds using Petri nets has two common features: (1) The net model is strongly connected, or non-strongly connected marked graph; and (2) The solution is based on linear programming.

Different from the previous work, this paper deals with the performance bounds of a discrete event systems via a particular structure SPN models, called *flow nets*. Flow nets are a special case of the well-behaved blocks[14] in terms of the net structures. Such flow nets can better model certain discrete event systems for throughput analysis than strongly connected SPNs and marked graphs. In a flow net, transitions are divided into two subsets, i.e., source transitions and non-source transitions, the former representing external task arrivals while the latter representing internal task processing. Since the throughput is determined by the structure and each unit's processing rate of the system, independent of the rates of the arrivals of external tasks, source transitions are defined to be unconditionally firable or having infinite liveness bound. A fast algorithm for determining the throughput of a flow net is proposed. For a structurally non-competitive and acyclic flow net, the algorithm proceeds in four steps: (1) Divide the places and transitions into groups according to some rules; (2) List the flow equilibrium equation for each place, which shows the relation among the average flows of its input and output transitions; (3) Simplify the constraints; And (4) solve a linear programming problem for the throughput of the model. If a system's SPN model is structurally competitive and cyclic flow net, or has initial tokens, we show how to equivalently transform it to a structurally non-competitive and acyclic one in terms of the same throughput.

The paper is organized as follows. Section 2 introduces the concept of a flow net, and presents the general description of the throughput of a flow net. In Section 3, we present our fast throughput analysis technique. Section 4 gives an example to illustrate the practical use of our technique.

2. Throughput of Flow Nets

In this section, we first briefly introduce stochastic Petri nets, then present the concept of flow nets, and finally give a general throughput model based flow nets.

2.1. Stochastic Petri Nets

A *Petri net*[1] is a bipartite directed graph in which the nodes are called *places* and *transitions*. A Petri net is a 4-tuple (P, T, I, O), where P is the set of places ($|P| = m$), T is the set of transitions ($|T| = n$, $P \cap T = \emptyset$, $P \cup T \neq \emptyset$), I (O) is the pre- (post-) incidence function representing the input (output) arcs I: $P \times T$ N = $\{0, 1, 2, . . .\}$ (O: $P \times T$ N). The pre- and post-sets of a transition $t \in T$ are defined as ${}^{\bullet}t = \{p \mid I(p, t) > 0\}$ and $t^{\bullet} = \{p \mid O(p, t) > 0\}$, where $I(p, t)$ and $O(p, t)$ are the multiplicities of arcs (p, t) and (t, p), respectively. The pre- and post-sets of place $p \in P$ are defined as ${}^{\bullet}p = \{t \mid O(p, t) > 0\}$ and $p^{\bullet} = \{ t \mid I(p, t) > 0\}$, respectively. $A = \{(x, y) \in (P \times T) \cup (T \times P)$: $I(x, y) > 0$ or $O(x, y) > 0)\}$ is the set of all the directed arcs in the net.

A function M: P N (usually represented in a column vector form) is called a *marking*. Markings represent (distributed) states. A *marked Petri net* (P, T, I, O, M_0) is a Petri net with an initial marking M_0. A transition $t \in T$ is *enabled* in M if and only if for any $p \in P$, $M(p) \geq I(p, t)$. A transition t enabled in M can *fire* and thus yield a new marking $M'(p) = M(p) - I(p, t) + O(p, t)$ for any $p \in P$. This is denoted by $M [t > M'$.

A *stochastic Petri net*[11] is obtained by associating each transition with an exponentially distributed *firing time*, which represents the duration of event or activity modeled by the transition, to the underlying Petri net. Formally, a marked stochastic Petri net is a 6-tuple $Z = (P, T, I, O, M_0, \Lambda)$, where Λ: $T \to R^+$ (the set of non-negative real numbers) and $\lambda_k \in \Lambda$ is the firing rate of transition $t_k \in T$.

Two transition firing policies are discussed by Ajmone Marsan *et al.*[15] Under the *preselection* policy, when a new marking is entered, the firing transition is selected according to a specified distribution. Once the choice is made, the firing delay is sampled from the distribution associated with the selection transition, and the marking changes after the delay. Under the *race* policy, all transitions sample a firing delay when a new marking is entered, and the minimum sampled delay determines both the transition that fires and the sojourn time in the marking. In this paper, the firing of transitions that do not compete for tokens from places is assumed to obey the race policy, and the firing of transitions that compete for tokens from places is assumed to obey the preselection policy.

2.2. Flow Nets

As shown in Figure 1, a *flow net* is an SPN such that

1. There are $m + q$ transitions and $n + s$ places, where t_1, t_2, ..., and t_q are unconditionally firable transitions, called *source transitions*, and others called *non-source transitions*; and places p_1, p_2, ..., and p_q have no output transitions, called *destination places*, and other places called *non-destination places*. Denote $T^0 = \{t_1, t_2, ..., t_q\}$ and $P^0 = \{ p_1, p_2, ..., p_q\}$.
2. Denote $M = (M^1, M^2)$, where $M^1 = (m(p_1), m(p_2), \dots, m(p_s))^T$ and $M^2 = (m(p_{s+1}), m(p_{s+2}), \dots, m(p_{s+n}))^T$. In initial marking, $M_0^1 = \mathbf{0}$. And no matter how many times source transitions fire, $M^1 = \mathbf{0}$ when no source transition is enabled.

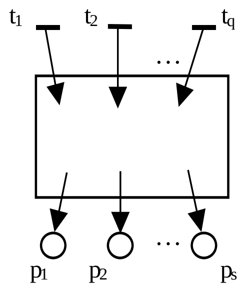


Figure 1. A multiple-input-transition-multiple-output-place block.

Flow nets are confined to the discrete event systems in which the internal task processing can be modeled without the help of the initially marked places that often model explicitly availability of resources. In many applications, the firing speed of a transition can reflect not only the quality of a resource but also the quantity of the resources. The structure of a flow net can include structures of choice, synchronization, and concurrent operations. One example is given in Figure 2.

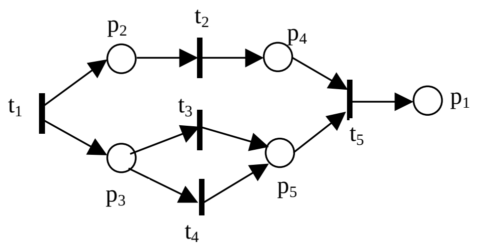


Figure 2. A flow net example with choice, synchronization and concurrency.

2.3. Throughput of Flow Nets

Consider an SPN with the properties of flow nets. Suppose that the firing rates of non-source transitions are constant. If the firing rates of all non-source transitions are high enough compared with that of any source transition, the input tokens which are generated by the firing of source transitions will be moved into destination places in time by the firings of non-source transitions, thus no token accumulates in any non-destination place. But with the increase of the firing rates of some source transitions the inputs will inevitably queue in pre-place(s) of some non-source transition whose firing rate is low, and the queue will grow infinitely over time. The transition is said to be *saturated*, i.e., it is enabled continuously as long as its input queue forms. When a non-source transition is saturated with reference to some specific source transition, the token processing rate of the whole net, corresponding to a system's throughput, keeps constant, even through the source transition's firing rate continues to increase while other source transitions' firing rates keep invariant.

Denote by $f(a, b)$ the average token flow rate on arc $(a, b) \in A$, where A is the set of all arcs in the net. Let $f(a, b) = 0$ for all $(a, b) \notin A$. Then the equilibrium of token flow at each place is of the form:

$$\sum_{t\in T} O(p,t)\, f(p,t) = \sum_{t\in T} I(p,t)\, f(p,t), \quad \forall p \in P \setminus P^0 \tag{1}$$

Denote the average token flow rate through transition $t_i \in T$ by f_i. It is easy to find that the flow on arc (t_i, p) or (p, t_i) is in accord with that through transition t_i. Thus we have

$$f(p,t_k) = \begin{cases} 0 & (p,t_k) \notin A, \\ f_k & (p,t_k) \in A. \end{cases} \tag{2}$$

$$f(t_k, p) = \begin{cases} 0 & (p,t_k) \notin A, \\ f_k & (p,t_k) \in A. \end{cases} \tag{3}$$

f_k is subject to

$$f_k \le \lambda_k, \qquad \forall t_k \in T^1. \tag{4}$$

Because source transitions are unconditionally firable, their average token flow rates are just their firing rates. According to the above analysis, the solution for the throughput of a flow net is converted into the following optimization problem:

$$\begin{aligned} &\max\ \lambda_1 + \lambda_2 + \dots + \lambda_n \\ &\ s.t.\ \sum_{t_j\in T} O(p_i,t_j) f_j = \sum_{t_k\in T} I(p_i,t_k) f_k,\ i = s+1, s+2, \dots, s+n, \\ &\qquad f_i \le \lambda_i,\ i = q+1, q+2, \dots, m. \end{aligned} \tag{5}$$

This is a linear programming problem with $n + m - s$ constraints.

3. Fast Throughput Analysis

In this section, we first present a fast technique for the throughput analysis of structurally non-competitive and acyclic flow nets, then show how to equivalently transform a structurally competitive and cyclic flow net, or an SPN with initial tokens, to a structurally non-competitive and acyclic flow net in terms of the same throughput.

3.1. Structurally Non-competitive and Acyclic Flow Nets

A flow net is said to be *structurally non-competitive* if there is at most one output transition for any place, i.e., $\forall p_i \in P$, $|p_i^\bullet| \le 1$. Otherwise, it is said to be *structurally competitive*.

A flow net is said to be *structurally cyclic* if there exists a set of transitions $x_1, x_2, \ldots,$ and x_h and a set of places $y_1, y_2, \ldots,$ and y_h, $h > 1$, such that

$$x_i \in y_i^\bullet,\ i = 1, 2, \ldots, h,$$
$$y_{i+1} \in x_i^\bullet,\ i = 1, 2, \ldots, h-1,$$
$$\ldots \ldots$$
$$y_1 \in x_h^\bullet.$$

Otherwise, it is said to be *structurally acyclic*. In a structurally acyclic flow net, no impact on a place can be made by places in its post-set.

For a structurally non-competitive and acyclic flow net, a structural partition can be obtained according to the following equations:

$$\begin{aligned}
&P_{j1} = \{p_i \in P \mid p_i^\bullet = \varnothing\},\\
&T_{j1} = \{ t_i \in T \mid t_i^\bullet \subseteq P_{j1}\},\\
&P_{j2} = \{ p_i \in P \setminus P_{j1} \mid p_i^\bullet \subseteq T_{j1}\},\\
&T_{j2} = \{t_i \in T \setminus T_{j1} \mid t_i^\bullet \subseteq (P_{j1} \cup P_{j2})\},\\
&P_{j3} = \{p_i \in P \setminus (P_{j1} \cup P_{j2}) \mid p_i^\bullet \subseteq T_{j2}\}, \qquad (6)\\
&T_{j3} = \{t_i \in T \setminus (T_{j1} \cup P_{j2}) \mid t_i^\bullet \subseteq (P_{j1} \cup P_{j2} \cup P_{j3})\},\\
&\ldots \ldots \ldots\\
&P_{jr} = \{p_i \in P \setminus (P_{j1} \cup P_{j2} \cup \ldots \cup P_{j,r-1}) \mid p_i^\bullet \subseteq T_{j,r-1}\},\\
&T_{jr} = \{t_i \in T \setminus (T_{j1} \cup T_{j2} \cup \ldots \cup T_{j,r-1}) \mid t_i^\bullet \subseteq (P_{j1} \cup P_{j2} \cup \ldots \cup P_{j,r-1})\}.
\end{aligned}$$

where, r is the minimum integer that satisfies

$$P = \bigcup_{k=1}^{r} P_{jk} \qquad (7)$$

Let

$$P_i = P_{j,r-i+1},\ i = 1, 2, \ldots, r, \qquad (8)$$

$$T_i = T_{j,r-i+1},\ i = 1, 2, \ldots, r. \qquad (9)$$

Then we can describe a structurally non-competitive and acyclic flow net simply as shown in Figure 3, where $P^0 = P_r$, $T^0 \subseteq T_1$. In order to obtain the flow equilibrium equation we reshow the relationship between P_i ($i > 1$) and each T_j in Figure 4:

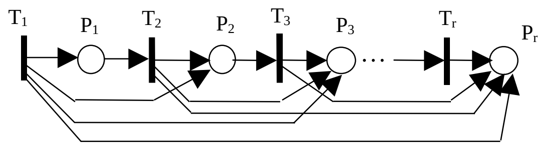


Figure 3. Sketch of a structurally non-competitive and acyclic flow net.

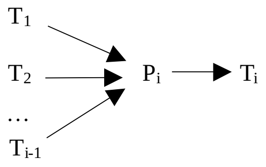


Figure 4. Relationship between a P_i ($i > 1$) and each T_j.

For any $t_k \in T_i$, if $p_j \in P_i$ and $I(p_j, t_k) > 0$ then t_k must be the sole output transition of p_j. Let $T^* = \bigcup_{v=1}^{i-1} T_v$. Thus we have the following equations:

$$I(p_j, t_k) f_k = \sum_{t_u \in T^*} O(p_j, t_u) f_u , \tag{10}$$

$$f_k = I(p_j, t_k)^{-1} \sum_{t_u \in T^*} O(p_j, t_u) f_u \, |_{I(p_j, t_k) > 0} . \tag{11}$$

For $i = 2$, the relationship between the average flow rate of any transition in set $T_2 \setminus T^0$ and the firing rates of transitions in set T_1 ($\subseteq T^0$) can be determined by Eq. (10). For $i = 3$, the same holds for $T_3 \setminus T^0$ and T_2. Continue this way till $i = r$. We thus obtain the relationship between the average token flow of each non-source transition and the firing rates of source transitions within a limited number of steps. Therefore, an algorithm for the throughput analysis of a structurally non-competitive acyclic flow net is derived as follows:

1. Divide the structure of the flow net according to Eqs. (6) and (7) and find P_1, T_1, P_2, T_2, …, P_q and T_q according to Eqs. (8) and (9).
2. List the flow equilibrium equation for each place in set P_1, P_2,…, and P_{q-1} according to Eq.(10). These equations can be formatted as the following recursive equations:

   $f_i = C_{i1}f_1 + C_{i2}f_2 + \cdots + C_{iq}f_q$, $i = q + 1, q + 2, \ldots, q + n$.

where, C_{i1}, C_{i2}, …, and C_{iq} are constant coefficients. Since source transitions are unconditionally firable ones, tokens flow through them at their firing rates. Thus the above equations can be written as

$$f_i = C_{i1}\lambda_1 + C_{i2}\lambda_2 + \cdots + C_{iq}\lambda_q\,,\ i = q + 1, q + 2, \ldots, q + n. \tag{12}$$

3. It follows from Eq.(12) and $f_i \le \lambda_i$, $i = q + 1, q + 2, \ldots, q + n$ that

$$\begin{bmatrix} C_{q+11} & C_{q+12} & \cdots & C_{q+1q} \\ C_{q+21} & C_{q+22} & \cdots & C_{q+2q} \\ & \cdots & \cdots & \\ C_{q+n1} & C_{q+n2} & \cdots & C_{q+nq} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \cdots \\ \lambda_q \end{bmatrix} \le \begin{bmatrix} \lambda_{q+1} \\ \lambda_{q+2} \\ \cdots \\ \lambda_{q+n} \end{bmatrix} \tag{13}$$

   Reduce all the redundant equations. If any two or more rows of the matrix C are identical, then just preserve the equation with the least firing rate in the column vector on the right. Suppose the number of inequalities is n_c ($\le n$).

4. Solve the linear program problem that has n_c constraints with the objective function defined in Eq. (5) to find the throughput of the flow net.

Remark: As we know, for most PN models of practical systems, $\forall$ ($p \in P$, $t \in T$) $I(t, p)$ is either 1 or 0, so is $O(p, t)$. Thus in Eq. (13), each coefficient C_{ij}, $i = q + 1, \ldots, q + n$, $j = 1, \ldots, q$, is either 1 or 0. It results that $n_c \le \min\{2^{q-1}, n\}$. Since q indicates the number of independent input sources of a system, usually we have $q << n$ and $2^{q-1} < n$. Particularly, in the case of $q = 1$ (i.e., the modeled system is a single input system. Many systems drop into this case.), we have $n_c = 1$, then the maximum processing capacity is obtained immediately. Therefore, compared with the conventional algorithm given by Eq. (5), our algorithm is more efficient.

3.2. Analysis of a Class of Structurally Competitive and Cyclic Flow Nets

Consider an SPN with a loop (subnet) L shown as Figure 6(a). In loop L, transitions x_1, x_2, …, and x_h are connected in series through places w_1, w_2, …, and w_h to form a loop. Tokens generated by the firing of the source transition x_0 move into subnet L, and depart from L by the firing of transition x_a. When a token arrives in place w_h, both x_a and x_b are enabled immediately. In other words, they compete for tokens in place w_h, thus there exists a problem of decision-making. The firing of x_b results in the same token being processed again by the subnet, so we call transition x_b a *feedback* transition, and call other elements (places and transitions) *forward transitions*. Without loss of generality, we assume that the firings of x_a and x_b obey the preselection policy, and denote the firing probability of x_a by α, and that of x_b by $1-\alpha$.

In order to analyze the processing capacity of such an SPN, a practical method is to make an equivalent transformation to turn the SPN into one with non-competitive and acyclic structure. Since x_1, x_2, …, and x_h are connected in series, the processing capacity is limited by the one with the smallest firing rate, say x_k, for example. Thus in the sense of throughput, subnet L in Fig. 5(a) can be transformed equivalently to that in Figure 5(b). From the flow equilibrium principle hold the following equations for places w_0 and w_h:

$$f(w_0, x_k) = f(x_0, w_0) + f(x_b, w_0),$$

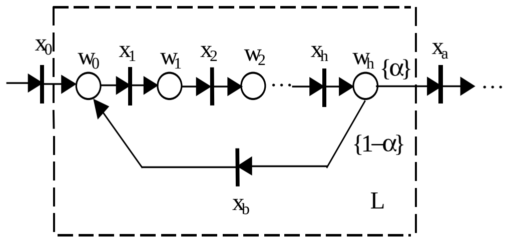
$$f(x_k, w_h) = f(w_0, x_k) = f(w_h, x_a) + f(w_h, x_b),$$

$$f(w_h, x_a) / f(w_h, x_b) = \alpha.$$

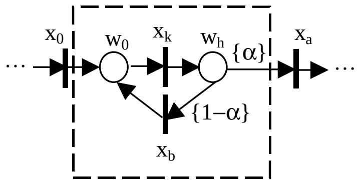
Then by noting $f(x_b, w_0) = f(w_h, x_b)$, we can obtain:

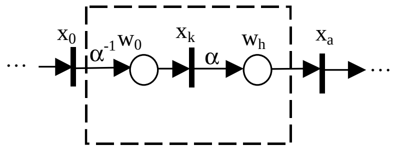
$$f(w_0, x_k) = \alpha^{-1} f(x_0, w_0), \qquad (14)$$

$$f(w_h, x_a) = \alpha f(x_k, w_0). \qquad (15)$$



(a) Original net.

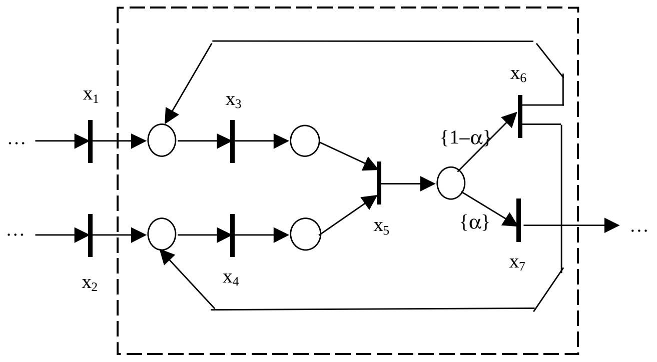




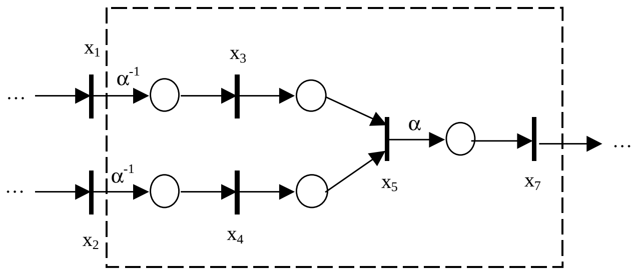
(b) Equivalent net 1. **(c) Equivalent net 2.**

Figure 5. Reduction of a flow net with a loop.

Eq. (14) shows that the existence of feedback transition x_b results in the increase of the processing burden of x_k, and thus decreases the throughput of the SPN. According to Eqs. (14) and (15), Fig. 5(b) can be converted into Fig. 5(c) from the analysis view point. In Fig. 6(c) $O(w_0, x_0)= \alpha^{-1}$ and $O(w_h, x_h) = \alpha$, which are not integers. Strictly speaking, Fig. 6(c) is no longer a Petri net. However, this way allows a structurally competitive and cyclic net to be transformed into a structurally non-competitive and acyclic one. Then the algorithm can apply to it without difficulty.



(a) Original net.



(b) Equivalent net

Figure 6. Reduction of a flow net with two loops sharing a feedback transition.

Using the similar method, we can reduce the net shown in Figure 6(a) where two loops share the same feedback transition, to the acyclic net shown in Figure 6(b).

The method can also be extended to flow nets with many other kinds of multiple loops. As examples, we consider the three cases shown in Figures 7–9. Figure 7 shows a flow net with two totally independent loops. We can apply the equivalent transformation technique twice to open the two loops. Figure 8 shows a flow net with two loops sharing forward elements, in which loop L1 is embedded in loop L2. We can first make an equivalent transformation for L1, and then do the same for L2. Figure 9 shows a flow net with two loops not only sharing the same forward elements but also sharing the same competitive resource. In this case, we replace the shared place with a simple net which is composed of two places and an immediate transition, as shown in Figure 10(b). Then the modified model takes the form of model shown in Figure 8.

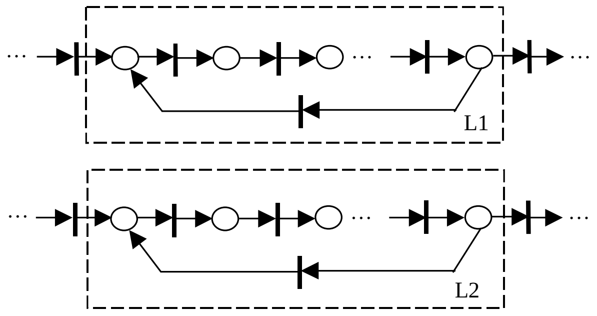


Figure 7. An SPN with two independent loops.

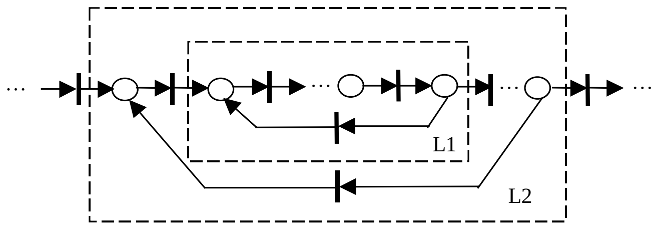


Figure 8. An SPN with two loops sharing the same forward elements.

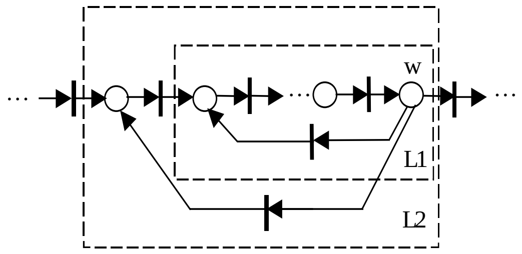


Figure 9. An SPN with two loops sharing the same competitive resource.

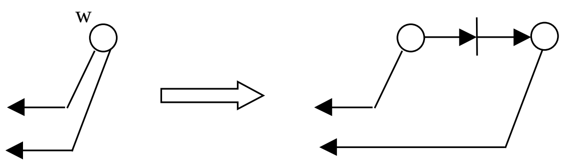
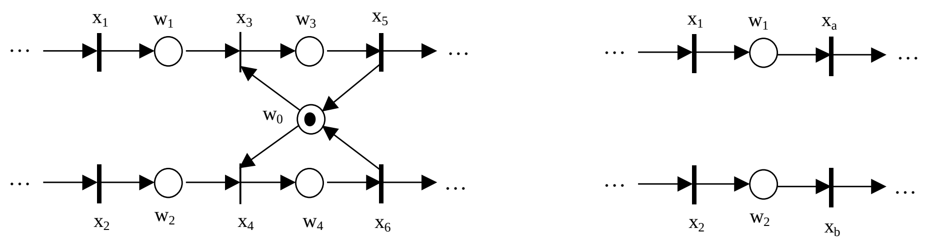


Figure 10. Separate the shared place in Figure 9 to two places connected by an immediate transition.

3.3. *Stochastic Petri Nets with Initially Shared Resources*

As we pointed out in Section 2.2, flow nets are confined to the discrete event systems in which the internal task processing can be modeled without the help of the initially marked places. However, there are many systems to be modeled with the help of the initially marked places to reflect explicitly availability of resources. In order to analyze the throughput of this class of systems, we show how to transform an SPN with initially shared resource to a flow net in this section.



(a) Original net. **(b) Equivalent net.**

Figure 11. Reduction of an SPN with shared resource.

Figure 11(a) shows a typical case of SPN's with initially shared resources. Two events, modeled by transitions x_5 and x_6, share a resource modeled by the token in place w_0. Immediate transitions x_3 and x_4, model the beginning of the two events. Let the firing rates of transitions x_5 and x_6 be λ_5 and λ_6, respectively. When the equilibrium of token flow is reached, the firing probability of transition x_3, denoted by α, is

$$\alpha = f(w_1, x_3) / (f(w_1, x_3) + f(w_2, x_4)),$$

and the firing probability of transition x_4 is $1 - \alpha$. Therefore, we can equivalently transform the net shown in Figure 11(a) to the net shown in Figure 11(b) in terms of the same throughput, where the firing rate of transition x_a is $\alpha\lambda_5$ and that of transition x_b is $(1 - \alpha)\lambda_6$. After the transformation, we can apply the technique given in Section 3.1 to analyze the throughput bound of the SPN.

4. Application to a Discrete Event System

Consider a system composed of 3 machines, 2 inspectors, 1 assembler and 2 disassemblers (Figure 12). The system receives two types of parts (A and B) as inputs, and after processing the input parts, one A-part and two B-parts are assembled into a final product. The process is described as follows: raw parts arrive in pairs, A-parts are processed by machines 1 and 2 in series, while B-parts are processed by machine 3. Processed A- and B-parts are finally assembled by an assembler. Two inspectors are responsible for quality control. Inspector 1 examines A-parts after they are processed by machines 1 and 2. If they don't satisfy the quality requirements, they are sent back to be re-processed by machines 1 and 2 in series. If so, they sent for assembly. Inspector 2 examines the assembled products. If an assembled product satisfies the quality requirements, it is unloaded from the system as a final product; otherwise, it is disassembled either by disassembler 1 or 2 depending upon their status. Disassembler 1 generates A-parts to be sent back to machines 1 and 2 and two B-parts to assembler, respectively. Disassembler 2 generates A-parts to be sent back to machines 1 and 2 and two B-parts to machine 3.

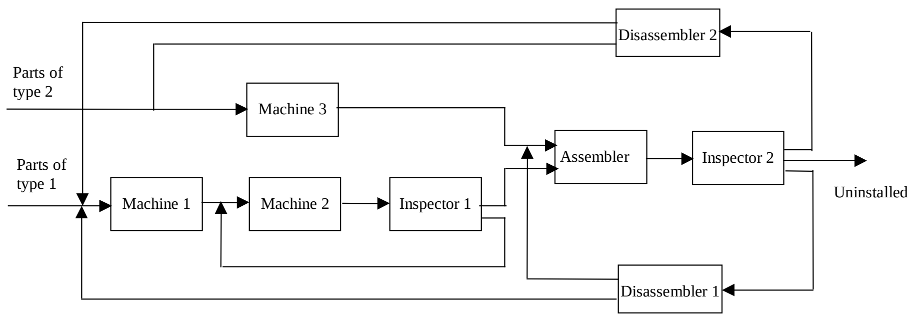


Figure 12. A discrete manufacturing system.

The durations of all events are assumed to be exponentially distributed random times. More specifically, the average times needed to process a part are: 2 minutes for machine 1, 2 minutes for machine 2, and 4 minutes for machine 3. The average time needed to assemble a final part at the assembler is 3 minutes. The average time needed to inspect a part for inspector 1 or 2 is 1 minute. 80% of A-parts pass inspector 1’s test. 20% of assembled parts cannot pass Inspector 2’s test and they are evenly sent back to disassemblers 1 and 2. The average time for disassemblers 1 and 2 to disassemble a part are same, i.e., 6 minutes.

Figure 13 shows the Petri net model of the manufacturing system with descriptions of places and transitions shown in Table 1. The net is also a flow net with one source transition t1 denoting the arrival of parts. There are 5 loops in the model:

Loop 1: $p_1t_2p_2t_3p_3t_4p_4t_6p_1$;

Loop 2: $p_1\ t_2p_2t_3p_3t_4p_4t_5p_5t_8p_8t_9p_9t_{11}p_1$;

Loop 3: $p_7t_8p_8t_9p_9t_{11}p_7$;

Loop 4: $p_1t_2p_2t_3p_3t_4p_4t_5p_5t_8p_8t_9p_9t_{12}p_1$;

Loop 5: $p_6t_7p_7t_8p_8t_9p_9t_{12}p_6$.

Using the technique given in Section 3.2, we convert the model into an acyclic one as shown in Figure 14 in terms of equivalent throughput. The structural division according to Eq. (6) for the equivalent model gives that

$T_1 = \{t_1\}$, $T_2 = \{t_2\}$, $T_3 = \{t_3\}$, $T_4 = \{t_4\}$, $T_5 = \{t_5, t_7\}$, $T_6 = \{t_8\}$,
$T_7 = \{t_9\}$, $T_8 = \{t_{10}\}$; $P_1 = \{p_1\}$, $P_2 = \{p_2\}$, $P_3 = \{p_3\}$, $P_4 = \{p_4, p_6\}$,
$P_5 = \{p_5, p_7\}$, $P_6 = \{p_8\}$, $P_7 = \{p_9\}$, $P_8 = \{p_{10}\}$.

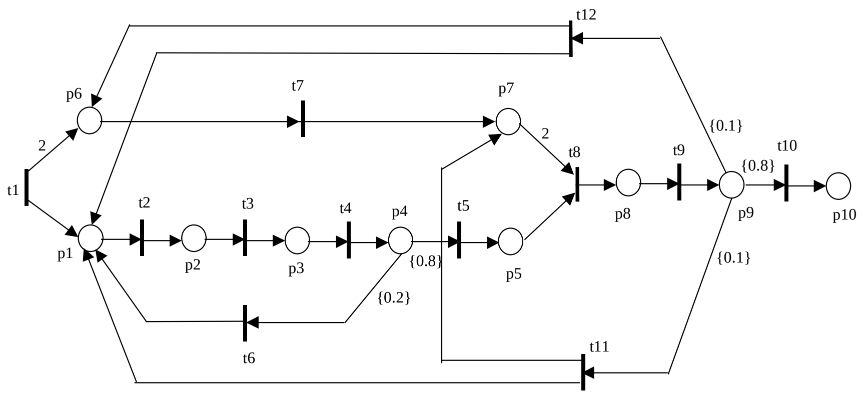


Figure 13. SPN model of an assembly system.

Table 1. Legend for Figure 13.

Place	Description	
p_1	Part of type 1	
p_2	A-part after the processing of machine 1	
p_3	B-part after the processing of machine 2	
p_4	Examining result of inspector 1	
p_5	A-part for assembly	
p_6	B-part	
p_7	B-part for assembly	
p_8	Assembled product	
p_9	Examining result of inspector 2	
p_{10}	Final product	
Transition	Description	Firing rate (1/hour)
t_1	One A-part and two B-parts arrive	λ_1
t_2	Machine 1 works on an A-part	λ_2=30
t_3	Machine 2 works on an A-part	λ_3=30
t_4	Inspector 1 examines an A-part	λ_4=60
t_5	A-part satisfies quality requirements	Immediate ($\lambda_5=\infty$)
t_6	A-part doesn't satisfy quality requirements	Immediate ($\lambda_6=\infty$)
t_7	Machine 3 works on B-part	λ_7=15
t_8	Assembler works	λ_8=20
t_9	Inspector 2 examines assembled product	λ_9=60
t_{10}	Final product is unloaded	λ_{10}=60
t_{11}	Disassembler 1 works	λ_{11}=10
t_{12}	Disassembler 2 works	λ_{12}=10

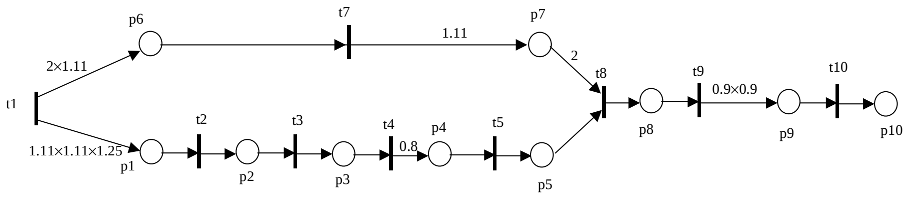


Figure 14. Equivalent model of the net shown in Figure 13.

Based on the algorithm given in Section 3.1, we list the relationships between the average token flow rate of each non-source transition and the firing rates of source transitions as following

$f_2 = 1.11\times1.11\times1.25\lambda_1$, (for P_1)
$f_2 = f_3$, (for P_2)
$f_4 = f_3$, (for P_3)
$f_5 = 0.8f_4$, $f_7 = 2\times1.11\lambda_1$, (for P_4)
$f_8 = f_5$, $2f_8 = 0.9\lambda_7$, (for P_5)
$f_9 = f_8$, (for P_6)
$f_{10} = 0.9\times0.9f_9$, (for P_7)

Rewrite these equalities as follows:

$\lambda_1 = 0.648f_2$, $\lambda_1 = 0.648f_3$, $\lambda_1 = 0.648f_4$, $\lambda_1 = 0.81f_5$,
$\lambda_1 = 0.45f_7$, $\lambda_1 = 0.81f_8$, $\lambda_1 = 0.81f_9$, $\lambda_1 = f_{10}$.

Because $f_i = \lambda_i$ for i = 2, 3, 4, 5, 7, 8, 9, and 10, so

$$\max \lambda_1 = \min \{0.648\lambda_2, 0.648\lambda_3, 0.648\lambda_4, 0.81\lambda_5, 0.45\lambda_7, 0.81\lambda_8, 0.81\lambda_9, \lambda_{10}\}$$
$$= 0.45\lambda_7 = 6.25.$$

Thus, the upper bound of the system throughput is 6.25 per hour.

5. Conclusions

This paper aims at developing a simple and fast algorithm for determining the throughput of discrete event systems. To the end, we introduced the concept of flow nets that are more suitable for the throughput analysis of many systems for which strongly connected marked graphs are not fit for their modeling or SPN's suffer from state explosion problems. In a flow net, transitions are divided into two subsets, i.e., source and non-source transitions, the former representing external task arrivals while the latter representing internal task processing. The fast algorithm is simpler than those based on linear programming. For a structurally non-competitive and acyclic flow net, the algorithm proceeds in four steps: First, divide the places and transitions into groups according to some rules. Next, list the flow equilibrium equation for each place, which shows the relationships among the average flows of the input transitions and the output transitions of any place. Then, deduce the relationships between the average flow of any non-source transition and that of all source transitions. Finally, determine the throughput of the model. For a class of structurally competitive and cyclic flow net, we showed how to convert it to a structurally non-competitive and acyclic one in the sense of equivalent throughput. Besides, we also presented a technique to transform an SPN with shared resource to a flow net.

Our future work includes the development of a software package to handle the throughput analysis for discrete event systems. The work needs also to extend to bottleneck analysis that is a more difficult problem for general cases.

6. References

1. Murata T., Petri nets: properties, analysis and applications, *Proc. of the IEEE*, **Vol.77, No. 4**, (April 1989) 561-580.
2. Zhou M.C. and Venkatesh K., *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*, (World Scientific, 1998)
3. Ajmone Marsan M., Balbo G., and Conte G., A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems, *ACM Trans. Computer Systems*, **Vol.2, No.1,** (May 1984) 93-122.
4. Florin G. and Natkin S., Evaluation based upon stochastic Petri nets of the maximum throughput of a full duplex protocol, *Informatik-Facherichte*, **Vol. 52,** (1982) 208-288.
5. Wang J., *Timed Petri Nets: Theory and Application*, (Kluwer Academic Publishers, Boston, 1998)
6. Molloy M., Performance analysis using stochastic Petri nets, *IEEE Trans. on Computers*, **Vol. C-31, No. 9,** (Sept. 1982) 913-917.
7. Florin G., Fraize C. and Natkin S., Stochastic Petri nets: properties, applications and tools, *Electronical Reliability*, **Vol. 31, No. 4,** (1991) 669-695.
8. Wang J. and Jiang S., Stochastic Petri net models of communication and flexible systems, in *Petri Nets in Flexible and Automation*, Zhou M. C., Editor, (Kluwer Academic Publishers, Boston, 1995)
9. Zhou M.C., Leu M.C., Modeling and performance analysis of a flexible PCB assembly station using Petri nets, *Trans. of the ASME, J. of Electronic Packaging*, **Vol. 113,** (1991) 410-416.
10. Zhou M.C., Guo D.L., and Dicesare F., Integration of Petri nets and moment generating function approaches for system performance evaluation, *J. of Systems Integration*, **Vol. 3,** (1993) 43-62.
11. Molley M., Fast bounds for stochastic Petri nets, *Proc. 1st Int. Workshop Petri Nets and Performance Models*, Turin, (July 1985) 244-249.
12. Campos J., Chiola G., and Silva M., Ergodicity and throughput bounds of the Petri nets with unique consistent firing count vector. *IEEE Trans. Software Eng.*, **Vol. 17, No. 2,** (Feb. 1991) 117-125.
13. Campos J., Chiola G., and Silva M., Properties and performance bound for closed free choice synchronized monoclass queueing networks, *IEEE Trans. Automatic Control*, **Vol. 36, No. 12,** (Dec. 1991) 1368-1381.
14. Suzuki I. and Muruta T., A method for stepwise refinement and abstraction of Petri nets, *J. Computer and System Sciences*, **Vol. 27,** (1983) 51-76.
15. Ajmone Marsan M., Balbo G., Bobbio A., Chiola G., Conte G., Cumani A., The effect of execution policies on the semantics and analysis of stochastic Petri nets, *IEEE Transactions on Software Engineering*, **Vol. SE-15, No. 7,** (July 1989) 832-846.