Proc. 2011 Computer Measurement Group Conference, Dec. 5-9, Washington D.C.

A Methodology for Combinining GSPNs and QNs

Daniel A. Menascé
Department of Computer Science
Volgenau School of Engineering
George Mason University
4400 University Dr., Fairfax, VA 22030, USA
menasce@gmu.edu

Abstract

Generalized Stochastic Petri Nets (GSPNs) are powerful mechanisms to model systems that exhibit instances of parallelism, synchronization, blocking, and simultaneous resource possession. For large systems, however, the solution of GSPNs is computationally very expensive due to the combinatorial growth of the state space. Queuing networks (QNs) provide very efficient solutions for the cases were parallelism, synchronization, blocking, and simultaneous resource possession are not present. This paper presents a methodology by which large GSPNs can be efficiently solved by automatically detecting subnetworks that are equivalent to product-form queuing networks (PFQNs). These subnetworks are replaced in the original GSPN by Flow-Equivalent Service Centers (FESCs). Each FESC is composed of a place/transition pair with marking dependent transition rates. These rates are obtained by solving the QN that corresponds to the subnetwork. The reduced GSPN obtained this way has fewer states than the original GSPN. Performance metrics derived from the reduced GSPN match those obtained from the original network with significant accuracy. This paper presents an algorithm to detect the subnetworks that are equivalent to a PFQN and an algorithm to obtain the service demands of the equivalent PFQN. The computational complexity of the algorithms is a function of the structural properties (number of places, transitions, and arcs) of the GSPN and not of the size of its reachability set. A theorem and two lemmas that serve as a basis for the algorithms are also presented.

1 Introduction

Generalized Stochastic Petri Nets (GSPNs) [8] are powerful mechanisms to model systems that exhibit instances of parallelism, synchronization, blocking, and simultaneous resource possession. For large systems, however, the solution of GSPNs is computationally very expensive due to the combinatorial growth of the state space. Product Form Queuing Networks (PFQNs), on the other hand, provide very efficient solutions for cases in which parallelism, synchronization, blocking, and simultaneous resource possession are not present. Vernon, Zahorjan, and Lazowska [14] compared QNs and GSPNs and concluded that while GSPNs have a greater descriptive power than QNs, specially in situations involving parallelism and synchronization, the evaluation techniques for GSPNs exhibit a much higher computational demand than QNs. The study suggests techniques that could be used to obtain approximate results for GSPNs at a reduced evaluation cost. These techniques are based on the decomposition [6] principle. They suggest that subnetworks of a larger GSPN that interact weakly with the rest of the network be solved in isolation, using the same techniques used to solve GSPNs, and that the entire subnetwork be replaced by a Flow Equivalent Service Center (FESC). The new (reduced) GSPN has fewer states and can then be solved more efficiently.

Balbo, Bruell, and Ghanta [3] suggested that PFQNs could be combined with GSPNs as a way to efficiently solve complex computer systems. They suggest that the subnetworks of a GSPN that are amenable to modeling with PFQNs should be modeled as such. Then, each subnetwork should be replaced by a FESC in the higher level GSPN. The solution of the reduced GSPN is an approximation to the solution of the complete GSPN. Their study does not present a methodology for identifying the subnetworks of a GSPN that can be modeled by PFQNs. They do not explain also how the FESC parameters are obtained.

Becker and Szczerbicka [4, 5] introduced PNiQ a combination of PFQNs and GSPNs in which the modeler has

to explicitly model parts of a system as a PFQN and combine them with a GSPN. This approach differs from ours, which automatically detects portions of a GSPN that can be solved using a PFQN. Balbo et al [2] provided arrival theorems for product-form stochastic Petri nets. For this class of networks, a formula for the mean sojourn time of a token in a place can be obtained. This formula serves as the basis of a Mean Value Analysis algorithm for Product-Form Stochastic Petri Nets. Klas and Seidel [7] suggest the combined use of decomposition [6] and response time preservation techniques [1] for the solution of very large GSPNs.

This paper presents a methodology by which large GSPNs can be efficiently solved by automatically detecting its subnetworks that are equivalent to product-form queuing networks (PFQNs). Subnetworks with only one input place and only one output transition are replaced in the original GSPN by Flow-Equivalent Service Centers (FESCs). Each FESC is composed of a place/transition pair with marking dependent transition rates. The marking dependent transition rates of each FESC are obtained by solving the QN that corresponds to the subnetwork. The reduced GSPN obtained this way has fewer states than the original GSPN. Performance metrics derived from the reduced GSPN match those obtained from the original network with significant accuracy. This paper presents an algorithm to detect the subnetworks that are equivalent to a PFQN as well as an algorithm to compute the service demands of these PFQNs. The computational complexity of the algorithms is a function of the structural properties (number of places, transitions, and arcs) of the GSPN and not of the size of its reachability set. The paper also presents a theorem and two lemmas that serve as a basis for the algorithms.

Section two presents the basic concepts and notation used throughout the paper. Section three discusses an example that motivates the need for the methodology. Next section presents the methodology and an example of its use. Finally, section five presents some concluding remarks.

2 Basic Concepts and Notation

A GSPN is a generalization of a Petri Net (PN), which is a model that allows for the easy representation of concurrency and parallelism. Petri Nets are directed graphs that have two types of nodes: places and transitions [10]. Places can only be connected to transitions and transitions can only be connected to places. Places can have zero or more tokens in them. When all input places to a transition have at least one token, the transition *fires* and as a result, one token is removed from each input places of the transition and one token is added to each output place of the transition. Firing of transitions is considered to be instantaneous in Petri Nets. A GSPN allows for both timed and instantaneous transitions as discussed below.

The notation used in this paper is pretty much standard for GSPNs. The reader is referred to [8] for a complete description of GSPNs. A brief description of the notation follows. A GSPN N is described by a tuple (P, T, A, M_0) where P is the set of places, T the set of transitions, $A \in (P \times T) \bigcup (T \times P)$ the set of arcs that connect places to transitions and transitions to places, M_0 is the initial marking of the places. We denote by $A_i \in (P \times T)$ the set of all input arcs to transitions and by $A_o \in (T \times P)$ the set of all output arcs of transitions. We denote by L the array of firing rates of timed transitions. The notation $I_t(p)$ refers to the set of input transitions of place p, $I_p(t)$ to the set of input places of transition t, $O_t(p)$ to the set of output transitions of place p, and $O_p(t)$ to the set of output places of transition t. We represent by $l(t)$ the firing rate of transition t, by $m(p)$ the marking of place p, and by $m_0(p)$ the initial marking of place p. The firing time of a transition t is assumed to be exponentially distributed with an average equal to $1/l(t)$.

3 Motivating Example

This section presents a motivating example that illustrates the purpose of the methodology and prepares the reader for the next section where the methodology is formally presented.

Figure 1 shows a GSPN that represents a computer system with a number of interactive terminals and a number of memory partitions where the submitted transactions execute. A submitted transaction must wait in a queue for memory when all memory partitions are in use. The computer system has one CPU and two disks. Once a transaction uses the CPU it may proceed to disk 1 with probability q_1, to disk 2 with probability q_2, or it may leave the system with probability q_0. The initial number of tokens in place p_1, $m_0(p_1)$, represents the number of terminals and $m_0(p_7)$ represents the number of memory partitions. The firing rate of transition t_1, $l(t_1)$, is marking dependent and is proportional to the number of terminals in the thinking state (i.e., number of tokens in p_1). Thus, $l(t_1) = m(p_1) \times (1/Z)$ where Z is the average think time at the terminals. The firing rates $l(t_3), l(t_6)$, and $l(t_7)$ are equal to μ_{cpu}, μ_{d1}, and μ_{d2} and represent the CPU, disk 1, and disk 2 service rates, respectively. The interactive computer system just described cannot be modeled by a PFQN because of the memory constraint. However, the sub GSPN inside the dashed box in figure 1 does not exhibit population constraints and can be modeled by a PFQN.

A PFQN equivalent to the sub GSPN inside the dashed box can be solved using efficient Mean Value Analysis (MVA) [11] methods. The solution of this PFQN provides the throughput $X_0(k)$ $(k = 0, \cdots, M)$ defined as the rate at which transactions complete from the computer system when the number of transaction in the computer system is equal to k and the number of memory partitions is M.

The next step in the methodology consists of replacing

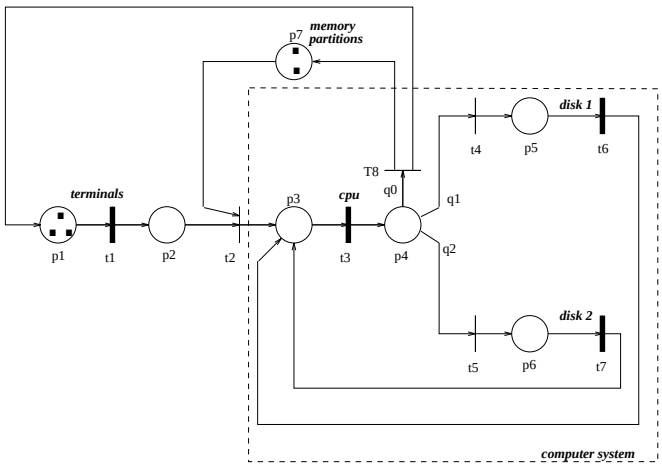

Figure 1: Complete GSPN for the Interactive System

the complete sub GSPN in figure 1 by a place-transition pair such that the place represents the entire computer system and the transition has a firing rate dependent on the marking of the place. Figure 2 shows the reduced GSPN that corresponds to figure 1. The firing rate $l(t_3)$ is a function of the computer system throughput $X_0(k)$. In particular, $l(t_3) = X_0(m(p_3))$.

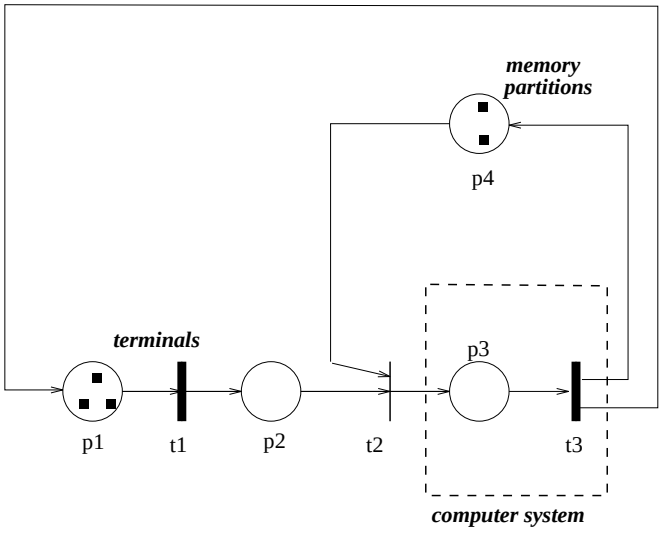

Figure 2: Reduced GSPN for the Interactive System.

To solve the PFQN equivalent to the sub GSPN and find the throughput $X_0(k)$, one must compute D_{cpu}, D_{d1}, and D_{d2}, the service demands at the CPU, disk 1, and disk 2, respectively. The service demand at any device i is equal to the average number of visits V_i to the device multiplied by the average service time S_i per visit [9]. The probability P_i^{cpu} that a transaction visits the CPU i times is

$$P_i^{cpu} = (1 - q_0)^{i-1} \times q_0 \quad (1)$$

Thus,

$$V_{cpu} = \sum_{i=1}^{\infty} i \; P_i^{cpu} = \sum_{i=1}^{\infty} i \; (1 - q_0)^{i-1} \times q_0 = 1/q_0 \quad (2)$$

It is clear that $V_{d1} = q_1 \; V_{cpu}$ and $V_{d2} = q_2 \; V_{cpu}$. The service demands can now be written as a function of the firing rates of the transitions in the dashed box of figure 1 as follows.

$$\begin{aligned} D_{cpu} &= V_{cpu} \; S_{cpu} = (1/q_0) \; (1/l(t_3)) \\ D_{d1} &= V_{d1} \; S_{d1} = (q_1/q_0) \; (1/l(t6)) \\ D_{d2} &= V_{d2} \; S_{d2} = (q_2/q_0) \; (1/l(t7)). \end{aligned}$$

Table 1 shows the number of states for the complete and reduced GSPNs for the interactive system example for various values of the numbers of terminals and memory partitions. It also displays the response time values obtained. They were exactly the same for both the complete and reduced GSPNs. The average response time in both cases is computed by subtracting the average think time at the terminals from the average cycle time. The average cycle time is obtained by applying Little's law to the entire interactive system. Thus, the cycle time is equal to the number of terminals divided by the average system throughput, which is given by the average firing rate of transition t_1. SPNP [12] was used to solve the GSPNs in this and all other examples in this paper. The values of q_0, q_1, q_2 are 0.1, 0.4, and 0.5, respectively, and the values of μ_{cpu}, μ_{d1}, and μ_{d2} are 100, 50, and 50 request/sec, respectively. The average think time Z is equal to 10 sec.

As the table indicates, the number of states of the complete GSPN increases in a combinatorial fashion with the number of terminals and memory partitions. For the reduced GSPN, the number of states grows with the number of terminals and is independent on the number of memory partitions. For example, the complete GSPN has 20,251 states for 100 terminals and 20 memory partitions while the reduced GSPN has only 101 states for the same case.

Next section presents a methodology to detect automatically the subnetworks of a GSPN that are equivalent to a PFQN.

4 The Methodology

The methodology for reducing a GSPN can be described as follows. Let N be a GSPN and let N' be the reduced GSPN obtained by applying the methodology described in this paper.

1. Detect the sub GSPNs in N that are equivalent to a PFQN and that have a single input place and a single output transition. The definitions of input place and

No. Terminals	No. Memory Partitions	Complete GSPN No. States	Reduced GSPN No. States	Avg. Response Time (sec)
5	3	40	6	.29
10	5	161	11	.30
15	10	616	16	.32
25	10	1,276	26	.36
30	10	1,606	31	.38
50	15	5,576	51	.50
100	20	20,251	101	1.50

Table 1: Number of States for the Interactive Computer System Example

output transition of a sub GSPN are given later in this section.

2. Compute the service demands for each of the PFQNs found in Step 1.

3. Replace each of the sub GSPNs detected in Step 1 by a place-transition pair. This pair is called the Flow Equivalent Service Center (FESC) for the sub GSPN. The firing rate of transition t in the FESC is a function of the marking of the place p in the FESC. More specifically, $l(t) = X_0(m(p))$ where $X_0(k)$ is the throughput obtained by solving the PFQN that corresponds to the sub GSPN for a population equal to k for $k = 0, \cdots, K^*$, where K^* is the maximum number of tokens in the sub GSPN. The value of K^* can be easily obtained by inspection of the complete GSPN in many cases. For example, in the GSPN of figure 1, the value of K^* is equal to the number M of memory partitions. When K^* cannot be easily determined by inspection, it can always be obtained by computing the reachability set of the complete GSPN. Note that this does not imply that one needs to solve the complete GSPN. The GSPN obtained by executing this step of the methodology is the reduced GSPN N'.

4. Solve N' and obtain the performance metrics of interest.

This section describes how steps 1 and 2 can be carried out automatically. Some definitions, a theorem, and two lemmas have to be presented first.

Definition 1 [subnetwork]: A GSPN $N' = (P', T', A', M'_0)$ is a *subnetwork* of a GSPN $N = (P, T, A, M_0)$ if

- $P' \subseteq P$
- $T' \subseteq T$
- $(p', t') \in A'_i \implies (p', t') \in A_i$
- $(p, t) \in A_i \implies (p, t) \in A'_i \quad \forall p \in P', t \in T'$
- $(t, p) \in A'_o \implies (t, p) \in A_o$
- $(t, p) \in A_o \implies (t, p) \in A'_o \quad \forall p \in P', t \in T'$
- $m'(p) = m(p) \quad \forall\, p \in P'$
- $l'(t) = l(t) \quad \forall\, t \in T'$ □

Definition 2 [consistent GSPN]: A subnetwork $N' = (P', T', A', M'_0)$ of a GSPN $N = (P, T, A, M_0)$ is said to be *consistent* if it satisfies the following conditions:

- if a transition $t \in T'$, then all its input places $\in P'$.
- if a place $p \in P'$, then all its output transitions $\in T'$.
- there is at least one *output transition* $t \in T'$ defined as a transition t such that at least one of its output places $\notin P'$.
- there is at least one *input place* $p \in P'$ defined as a place p such that at least one of its input transitions $\notin T'$ or is an output transition of N'. □

Note that the subnetwork within the dashed box in figure 1 is consistent. In this subnetwork, t_8 is an output transition and p_3 is an input place.

Definition 3 [equivalence to PFQN]: A GSPN N is said to be *equivalent* to a PFQN if there is a PFQN whose throughput is the same as the one obtained by solving the GSPN. Such GSPNs will be referred to as PF-GSPN. □

Consider now the following theorem.

Theorem: Let $N' = (P', T', A', M'_0)$ be a consistent subnetwork of a GSPN $N = (P, T, A, M_0)$. N' is a PF-GSPN if

1. all transitions in N' have a single input arc,
2. all transitions in N' that are not output transitions have a single output arc, and
3. N' does not have any random switch with a marking dependent firing rate.

Proof: To prove the theorem we need to show i) what are the possible combinations of places and transitions in N', ii) show how these combinations map into queuing network elements, and iii) show that none of these elements violates any of the conditions for product-form solution, namely: one step behavior, flow balance, single resource possession, no blocking, independent customer behavior, local information, fair service, and routing homogeneity. See [1] for further explanation on the conditions for product form solution of QNs. Figure 3 shows the only possibilities of arrivals and departures from transitions and for departures from places allowed within N'. Figures 3 (a) and 3 (b) illustrate conditions 1 and 2 of the Theorem for immediate and timed transitions. Figure 3 (c) illustrates that the output of a place can go to one or more transitions. In the case of more than one output, we have a marking independent random switch.

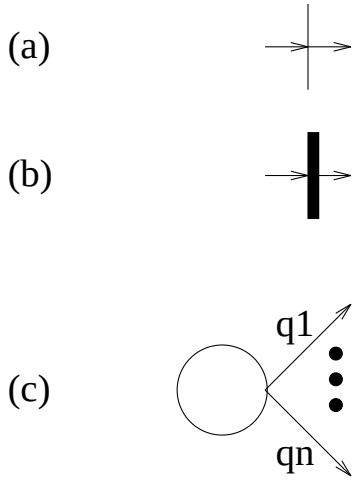

Figure 3: Building blocks for N'.

The building blocks in figure 3 can be combined as shown in figure 4. Figure 4 (e) represents the case where some of the output transitions of a place are immediate transitions and some are timed. The figure also shows the QN elements that correspond to the GSPN elements. It should be noted that places are mapped as queues if the transition that follows them is a timed transition.

It is easy to see that the QN elements of figure 4 do not create any situation that violates product-form conditions. Since transitions have only one input arc and consequently one input place and one output arc and one output place, it is not possible to represent situations where blocking, simultaneous resource possession, fork-join, or any other form of behavior in which customers do not flow independently or have to synchronize in any form. Since we do not allow random switches with marking dependent firing rates, we guarantee routing homogeneity. Q.E.D.

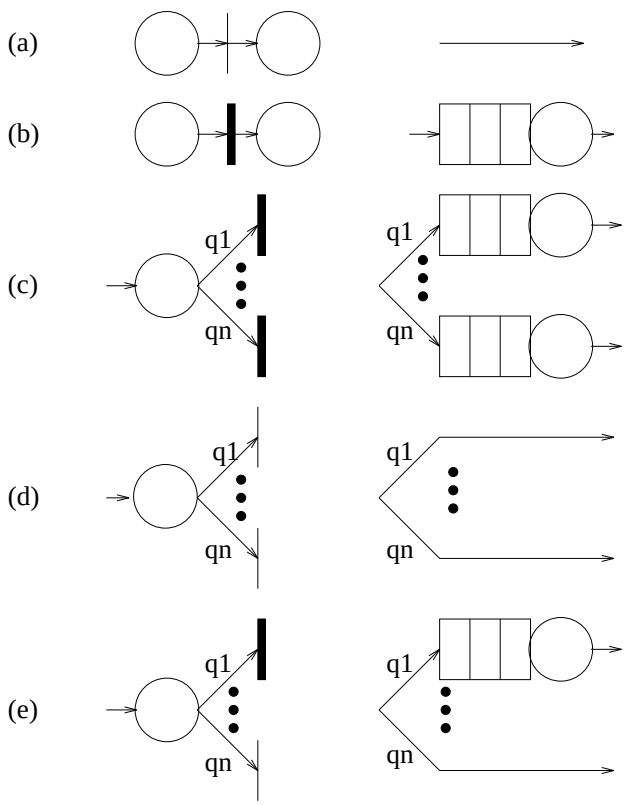

Figure 4: Possible mappings from GSPN into QNs for Theorem.

All GSPNs that satisfy the theorem are PF-GSPNs. We refer to these PF-GSPNs as TPF-GSPNs to indicate that they satisfy the conditions of the theorem.

Lemma 1: Let t be a transition with more than one input arc. Then, t and all of its input places do not belong to a TPF-GSPN.
Proof: Transition t does not satisfy condition 1 of the theorem and therefore is not part of a TPF-GSPN. Let p be an input place to t, and let p belong to a TPF-GSPN N. Since N is consistent by the theorem, t has to be part of network N by definition of consistency. This implies that t belongs to a TPF-GSPN which is a contradiction. So, no input places to t belong to a TPF-GSPN. Q.E.D.

Lemma 2: Let t be a transition in a subnetwork N' of a GSPN N. If t has a single input arc and its input place p is not in N', then N' is not a TPF-GSPN.
Proof: Since the place p is not in N', N' is not consistent by definition. Therefore, by the above theorem, N' is not a TPF-GSPN. Q.E.D.

The theorem and lemmas above are the basis for the following algorithm for detecting subnetworks of a GSPN

N that are PF-GSPNs.

Algorithm for Detecting PF-GSPNs:

1. For each transition t, check if t has more than one input place or one input place with more than one arc from the place to t. Mark the input places for which this is true as not belonging to a PF-GSPN (see Lemma 1).

2. For each place p marked in Step 1 (places that do not belong to a PF-GSPN), mark all output transitions of p as not belonging to a PF-GSPN (see Lemma 2).

3. For each transition t belonging to a PF-GSPN (not marked in Step 2), check if t has more than one output place or a single output place connected to t by more than one arc, or an output place that does not belong to a PF-GSPN (marked in Step 1). If t satisfies any of these conditions, then mark t as an output transition of a PF-GSPN.

4. For each place p belonging to a PF-GSPN (not marked in Step 1), check if p is an output place of a transition not belonging to a PF-GSPN (marked in Step 2) or if p is an output place for an output transition of a PF-GSPN (marked in Step 3). Mark each place p that satisfies one of these two conditions as an input place to a PF-GSPN.

5. Detect the PF-GSPNs as follows:

   (a) for each input place p of a PF-GSPN (marked in Step 4), traverse the GSPN until transitions labeled as output transitions in Step 3 are reached. Any graph traversal algorithm such as depth-first search [13] can be used here. All places and transitions visited this way belong to the same subnetwork as p and so do all output transitions reachable from p.

   (b) compare the sets of input places and output transitions obtained in Step 5 (a). If the same output transition can be reached from input places p_1 and p_2, all places and transitions reachable from p_1 and p_2 are in the same PF-GSPN subnetwork.

Step 2 of the methodology presented at the beginning of this section consists in obtaining the service demands for each PFQN that corresponds to a PF-GSPN. As seen, devices in the PFQN correspond to timed transitions in the corresponding PF-GSPN. The service demand D_i at a device i of a QN is equal to the product $V_i\ S_i$ where V_i is the average number of visits to device i and S_i is the average service time at device i. S_i is given by the inverse of the firing rate of the transition in the PF-GSPN that corresponds to device i. One is then left with the problem of obtaining the average number of visits V_i's.

The algorithm for obtaining the average number of visits is given below for the case in which the PF-GSPN has a single input place and a single output transition. These cases tend to exhibit a smaller degree of interaction between the PF-GSPN being replaced and the rest of the GSPN, yielding a better approximation [6].

Before we explain the algorithm for computing the visit ratios, a few definitions are in order.

Definition 4 [g-node]: The input place of a PF-GSPN N', its output transition, and any place of N' with either more than one input transition or more than one output transition are called *g-nodes* of N'. These g-nodes will be referenced by the name of the corresponding place or transition. □

Consider the PF-GSPN in the dashed box labeled as subnet 2 in figure 5. The g-nodes of this GSPN are the input place p_5, the output transition t_{10}, and the place p_6.

Definition 5 [path of a GSPN]: A *path* (n_i, n_j) of a PF-GSPN is a path that connects g-node n_i to g-node n_j. A path may contain places that are not g-nodes of the PF-GSPN. If there is more than one path connecting g-node n_i to g-node n_j we refer to it as (n_i, n_j, t) where t is any transition in the path in question. □

For example, there are two paths in the PF-GSPN of subnet 2 of figure 5 from g-node p_6 to g-node p_5. One of these paths goes through transition t_8 and the other through transition t_9. These paths are designated as (p_6, p_5, t_8) and (p_6, p_5, t_9), respectively.

Definition 6 [path probability]: The *probability of a path* is defined as the probability that a token flows through the path from its initial g-node. □

The probability of path (p_6, p_5, t_8) in figure 5 is q_5 and the probability of path (p_6, p_5, t_9) is q_6.

Note that all transitions of a path are visited the same number of times. Thus, if one obtains the average number of visits to the paths of a PF-GSPN we obtain the average number of visits to all transitions in the PF-GSPN. If there is more than one path between g-nodes n_i and n_j of a PF-GSPN and g-node n_i is an input place to one or more immediate transitions, then all paths that originate from n_i and have a timed transition as an output transition of n_i can be eliminated since these timed transitions will never fire. In other words, all timed transitions in the eliminated paths will have an average visit ratio equal to zero. Subnet 3 of figure 5 illustrates this situation. Note that the elimination of paths (p_{12}, p_{17}, t_{14}) and (p_{12}, p_{17}, t_{15}) requires that probabilities q_3 and q_4 be adjusted so that they sum to one. The new value for q_3 is adjusted to $q_3/(q_3 + q_4)$ and the adjusted value for q_4 is $q_4/(q_3 + q_4)$. In general,

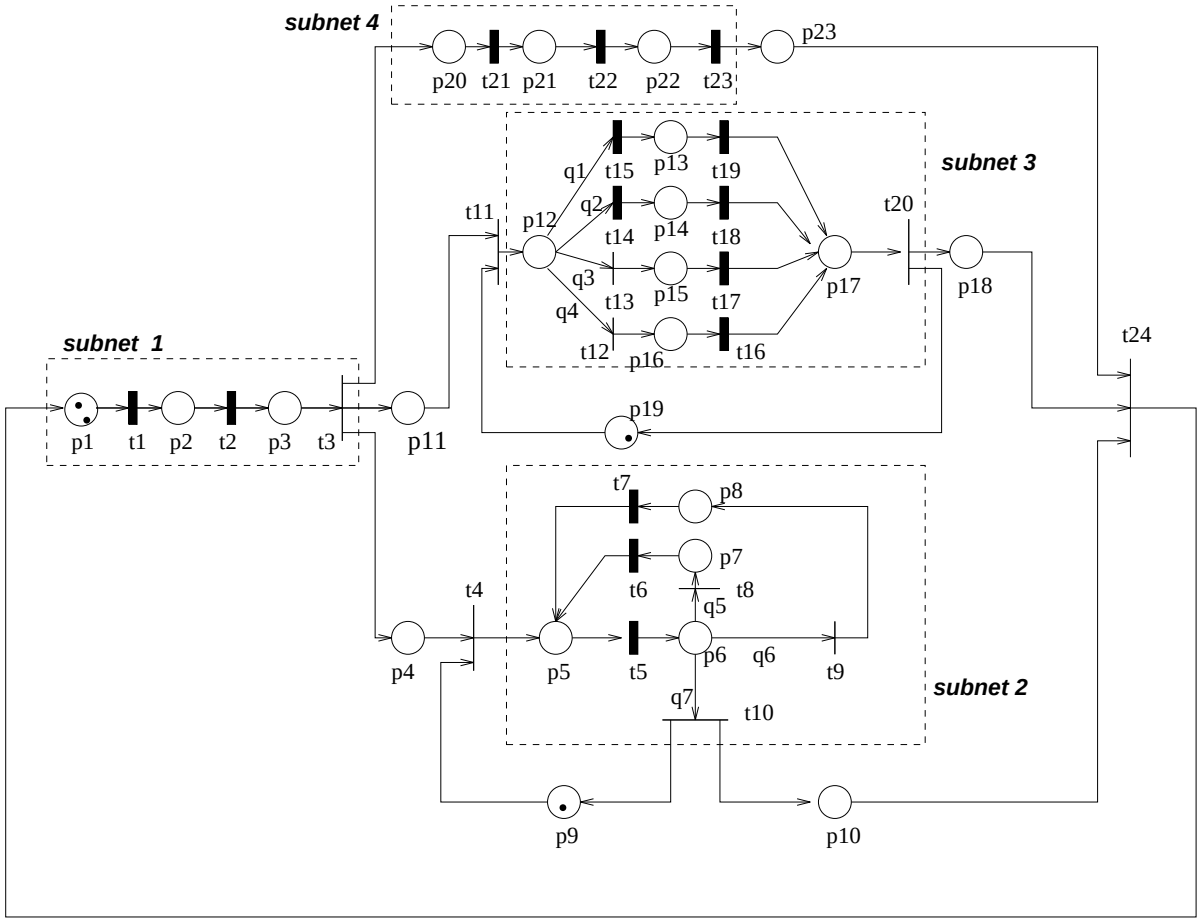

Figure 5: GSPN with several PF-GSPNs.

the adjusted probability of a path that remains is equal to its original value divided by the sum of the probabilities of all paths that remain.

After eliminating the paths with zero probability, one must obtain the average number of visits to the remaining paths in the PF-GSPN. For that purpose we define a condensed graph associated to a PF-GSPN as follows.

Definition 7 [condensed graph of a GSPN]: The *condensed graph* (CG) of a PF-GSPN is a directed graph (V, E) built as follows:

- The set of nodes $V = \{v_0\} \bigcup G$ where G is the set of g-nodes of the PF-GSPN and v_0 is a node connected through a directed arc $a = (v_0, w)$ to g-node w, where w is the g-node of the PF-GSPN associated with its input place.
- The set of arcs $E = \{a\} \bigcup A$ where a is the arc defined above and A is derived from the set of paths of the PF-GSPN as follows.
  - $(v_i, v_j) \in A$ if there is a single path in the PF-GSPN connecting g-node v_i to g-node v_j.
  - if there is more than one path in the PF-GSPN connecting g-node v_i to g-node v_j, then create a single arc (v_i, v_j) in A to represent all paths that connect v_i to v_j in the PF-GSPN.
- Probabilities are assigned to the arcs of GC as follows.
  - the probability of arc a defined above is 1.
  - the probability of an arc derived from a single path in the PF-GSPN is the probability of the path.
  - the probability of an arc derived from more than one path in the PF-GSPN, is the sum of the probabilities of the constituent paths. □

Figure 6 shows the CG that corresponds to subnet 2 of figure 5. Nodes v_1 and v_2 correspond to places p_5 and p_6 in the PF-GSPN, as indicated in the figure, and node v_3 corresponds to the output transition t_{10}. Arc (v_2, v_1) is originated from paths (p_6, p_5, t_8) and (p_6, p_5, t_9) in the PF-GSPN of subnet 2 of figure 5. Therefore, the probability of this arc is equal to the sum of the probabilities of the two paths. The average number of visits to a path in the PF-GSPN can now be easily computed from the average number of visits to the nodes of a CG as follows. Let $e = (v_i, v_j)$ be an arc in the CG and let π be a path in the GSPN associated to e. The average number of visits to π is equal to the average number of visits to v_i multiplied by the probability of the path π. For example, the average number of visits to path (p_6, p_5, t_8) is equal to q_5 multiplied by V_2, the average number of visits to node v_2 in the CG

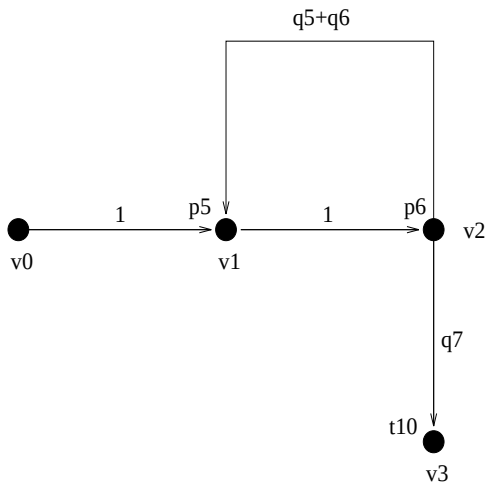

Figure 6: Condensed Graph Example

of figure 6. One is then left with the problem of obtaining the average number of visits to the nodes of the CG.

Let V_i denote the average number of visits to node v_i of the CG. V_i can be written as a function of the average number of visits to all nodes of the CG and as a function of the probabilities $p_{j,i}$ of the arc (v_j, v_i). If an arc (v_j, v_i) does not exist in the CG, its probability $p_{j,i}$ is assumed to be zero. Thus,

$$V_i = \sum_{j=0}^{K} V_j \times p_{j,i} \tag{3}$$

where K is the largest index of a node in the CG. We assume without loss of generality that the output transition of a PF-GSPN corresponds to node v_K in the condensed graph. If one writes such an equation for every node one obtains a system of linear equations that can be written as

$$\vec{V} = \vec{V}\ P \tag{4}$$

where $\vec{V} = [V_0, \cdots, V_K]$ and the matrix $P = [p_{i,j}]$ is the matrix of the arc probabilities. Note that $V_0 = 1$ and $V_K = 1$ since v_0 corresponds to the input place of the GSPN and v_K corresponds to the output transition of the GSPN. With this substitution one can obtain a unique solution to the system of linear equations in Eq. (4).

The process of finding the service demands for the PFQN that corresponds to a PF-GSPN can be summarized as follows:

1. Determine the paths of the PF-GSPN.
2. Remove from the PF-GSPN the paths that are never visited. The service demands associated to transitions in these paths should be set to zero. When paths are eliminated from the PF-GSPN, the probabilities of the remaining paths must be adjusted.

3. Build the Condensed Graph CG that corresponds to the PF-GSPN.

4. Obtain the average number of visits to the g-nodes of CG by solving the system of linear equations $\vec{V} = \vec{V} \times P$.

5. Obtain the average number of visits to the paths of the PF-GSPN from the average number of visits to the nodes of the CG.

6. Compute the service demand associated to each timed transition by multiplying the average number of visits to the path where the transition is located by the inverse of the transition firing rate.

To illustrate the methodology, consider the GSPN of figure 5. This GPSN has four PF-GSPNs shown within dashed boxes. The probabilities $q_1, \cdots, q_7$ have the values 0.1, 0.2, 0.3, 0.4, 0.3, 0.4, and 0.3. The average firing times of the timed transitions are given in Table 2.

Transition	Avg. Firing Time (sec)	Avg. No. Visits	Avg. Service Demand (sec)
t_1	2.0	1.00	2.00
t_2	3.0	1.00	3.00
t_5	2.0	3.33	6.66
t_6	1.0	1.00	1.00
t_7	3.0	1.33	4.00
t_{14}	3.0	0.00	0.00
t_{15}	4.0	0.00	0.00
t_{16}	7.0	0.57	3.99
t_{17}	6.0	0.43	2.58
t_{18}	5.0	0.00	0.00
t_{19}	4.0	0.00	0.00
t_{21}	6.0	1.00	6.00
t_{22}	5.0	1.00	5.00
t_{23}	4.0	1.00	4.00

Table 2: Service Demands for GSPN of Figure 5.

The average number of visits to the nodes of the CG of figure 6 are given by the equations:

$$\begin{aligned} V_0 &= 1 \\ V_1 &= V_0 + V_2\ (q_5 + q_6) \\ V_2 &= V_1 \\ V_3 &= 1 \end{aligned}$$

Solving these equations one gets $V_1 = V_2 = 1/(1 - q_5 - q_6) = 1/q_7$. The average number of visits to t_5 is equal to V_1, the average number of visits to t_6 is $V_2 \times q_5 = q_5/q_7$, and the average number of visits to t_7 is $V_2 \times q_6 = q_6/q_7$. Table 2 shows the average number of visits and the average service demands for all timed transitions of the GSPN of figure 5.

To illustrate the importance of the methodology presented in this paper, we solved the GSPN of figure 5 with SPNP. The underlying Markov Chain has 59,669 states. It took approximately 90 minutes to solve this GSPN on a Sun Ultra 1 workstation. The reduced GSPN that corresponds to the one in figure 5 has 216 states and was solved on the same workstation in less than one second. In both cases we used as initial markings $m_0(p_1) = 5, m_0(p_9) = 3$, and $m_0(p_19) = 3$. All other places had zero tokens in the initial marking.

5 Concluding Remarks

The methodology presented in this paper provides an efficient manner to detect the subnetworks of a GSPN that can be replaced by a marking dependent place-transition pair. The firing function of the transition in the pair can be obtained by solving a PFQN equivalent to the replaced subnetwork. The algorithms given in the paper show how the service demand parameters for the PFQNs are computed. The computational demand of the methodology is $\mathcal{O}(\mid T \mid + \mid P \mid + \mid A \mid)$ for detecting the PF-GSPNs plus a factor due to computing the service demands. This factor is dominated by the time to solve k system of linear equations where k is the number of PF-GSPNs detected (note that $k < (\mid P \mid + \mid T \mid)$). Each of these systems of linear equations has a number of equations bounded above by $\mid P \mid$. It is clear then, that the computational complexity of the methodology is low and is a function of the structural properties of the GSPN (number of places, transitions, and arcs) and is independent of the initial marking. The method presented in this paper can easily be implemented into GSPN solvers such as SPNP and others.

Acknowledgements

Thanks to Prof. Kishor Trivedi of Duke University for providing us with a copy of SPNP.

References

[1] Agrawal, S. C., J. P. Buzen, and A. W. Shum, "Response Time Preservation: A General Technique for Developing Approximate Algorithms for Queuing Networks," Performance Evaluation Review, vol. 12, no. 3, pp. 63-77, 1984.

[2] Balbo, G., S.C. Bruell, and M. Sereno, "Arrival Theorems for Product-Form Stochastic Petri Nets," Proc. of the ACM Sigmetrics, Nashville, Tennessee, May 16-20, 1994, pp. 87-97.

[3] Balbo, G., S. Bruell, and S. Ghanta, "Combining Queuing Networks and Generalized Stochastic Petri Nets for the Solution of Complex Models of System Behavior," IEEE Transactions on Computers, Vol. 37, No. 10, October 1988.

[4] Becker, M. and H. Szczerbicka, H., "PNiQ: integration of queuing networks in generalised stochastic Petri nets," Software, IEE Proceedings - , vol.146, no.1, pp.27-32, Feb 1999.

[5] Becker, M. and H. Szczerbicka , "Integration of multi-class queueing networks in generalized stochastic Petri Nets," 2001 IEEE Intl. Conf. Systems, Man, and Cybernetics, ,vol.2, no., pp. 1137- 1142 vol.2, 2001

[6] Courtois, P. J., "Decomposability: Queuing and Computer System Applications," Academic Press, NY, 1977.

[7] Klas, G. I. and U. G. Seidel, "A Subsystem Identificating Algorithm for the Approximate Solution of Large GSPN Models," Proc. of the Fifth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Torino, Italy, February 13-15, 1991.

[8] Marsan, A., G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," ACM Trans. on Computer Systems, 2(2), May 1984, pp. 93-122.

[9] Menascé, D.A., V.A.F. Almeida, and L.W. Dowdy, *Performance by Design: Capacity Planning by Example*, Prentice Hall, Upper Saddle River, NJ, 2004.

[10] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Upper Saddle River, NJ, 1981.

[11] Reiser, M. and S. Lavenberg, "Mean-Value analysis of closed multi-chain queuing networks," J. of the ACM, Vol. 27, No. 2, 1980.

[12] Ciardo, G., J. Muppala, and K. Trivedi, "Manual for the SPNP Package Version 4.0", Duke University, September 1995.

[13] Hopcroft, J. and R. Tarjan, "Planarity testing in V log V steps," Proc. IFIP Congress, Ljubljana, Booklet Ta-2, August 1971, 18-23.

[14] Vernon, M., J. Zahorjan, and E. Lazowska, "A Comparison of Performance Petri Nets and Queuing Network Models," Proc. of the International Workshop on Modeling Techniques and Performance Evaluation, Paris, March 9-11, 1998.