

# On Throughput Approximation of Resource-Allocation Systems by Bottleneck Regrowing

Ricardo J. Rodríguez<sup>1b</sup> and Javier Campos

**Abstract**—Complex systems, such as manufacturing, logistics, or Web services, are commonly modeled as discrete event systems dealing with the resource-allocation problem. In particular, Petri nets (PNs) are a widely used formalism to model these systems. Although their functional properties have been extensively studied in the literature, their nonfunctional properties (such as throughput) have usually been ignored. In this brief, we focus on a PN subclass useful for modeling concurrent sequential processes with shared resources, termed  $S^4PR$  nets. For these nets, we present an iterative strategy that makes intensive use of mathematical programming problems to approximate system throughput. Initially, our strategy selects the slowest part (a subsystem) of the net. Then, the next slowest parts are considered. In each step, the throughput is computed solving analytically the underlying continuous-time Markov chain when feasible (or by simulation, otherwise). Since only certain subsystems are considered, the state-explosion problem inherent to the increasing net size is mitigated. We evaluate our strategy in a set of randomly generated  $S^4PR$  nets. Our findings show that the throughput improves the upper throughput bound computation by almost 20% and that small portions of the net are enough to approximate system throughput.

**Index Terms**—Discrete event systems (DESs), linear programming (LP), performance evaluation, Petri nets (PNs), resource-allocation systems (RASs).

## I. INTRODUCTION

MANUFACTURING, logistics, or Web services, to name a few, are usually complex systems using shared resources. The use of shared resources introduces synchronization issues among parties. Once a party allocates a resource, others must wait until the former ends its activity and releases the resource when no free resource instances are available to complete their activities.

Many of these artificial systems can be naturally modeled as discrete event systems (DESs) dealing with the resource-allocation problem, also called resource-allocation systems (RASs).<sup>1</sup> An RAS is a DES in which a set of concurrent processes coexist, which must compete in order to allocate some shared resources [1]. Petri nets (PNs) have been widely

Manuscript received March 14, 2017; revised August 1, 2017; accepted October 11, 2017. Date of publication November 15, 2017; date of current version December 12, 2018. Manuscript received in final form October 26, 2017. This work was supported in part by the EU H2020 through the DICE Project under Grant 644869 and in part by the Spanish MINECO through the CyCriSec Project under Grant TIN2014-58457-R. Recommended by Associate Editor O. Stursberg. (Corresponding author: Ricardo J. Rodríguez.)

R. J. Rodríguez is with the Centro Universitario de la Defensa, Academia General Militar, 50090 Zaragoza, Spain (e-mail: rjrodriguez@unizar.es).

J. Campos is with the Department of Information Technology and System Engineering, Universidad de Zaragoza, 50018 Zaragoza, Spain (e-mail: jcampos@unizar.es).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCST.2017.2768512

<sup>1</sup>In this brief, we use the RAS interchangeably as a singular and plural acronym.

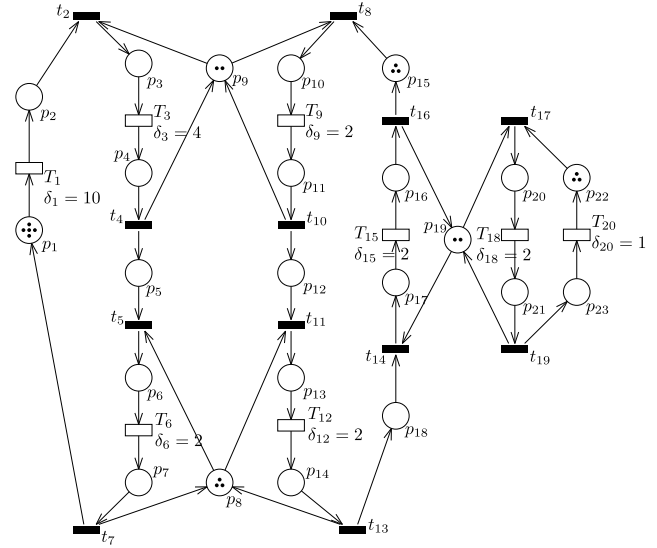


Fig. 1. Particular example of an  $S^4PR$  net.

used to model the RAS. In this regard, a modular methodology based on three steps is usually followed [2]: 1) to characterize production plans as a PN, being processes modeled as tokens within the net; 2) to add resources into each production plan, represented as a place with an initial number of tokens (available copies of the resource); and 3) to build the global model by the composition of production plans through resources, fusing resource places representing the same resource type in different production plans. Following the above methodology, we concentrate on the subclass of RAS that can be modeled by means of  $S^4PR$ , a PN subclass allowing the modeling of concurrent sequential processes with routing decisions and a general conservative use of resources [2].

In these nets, workpieces undergo successive transformations, which may also have independent processing steps, until reaching their final state. A production plan is represented by means of a strongly connected state machine (with no internal cycles), in which availability of different routings in the system may be restricted to the use of nonconsumable, reusable resources.

Fig. 1 depicts a particular example of  $S^4PR$ . It represents three sequential processes without any routing decisions and three resources: the initial marking of  $p_8$  and  $p_9$  is the number of idle resources of each type shared by the left-hand side and the central process, while  $p_{19}$  represents the resource shared by the central and the right-hand side process.

$S^4PR$  nets have been widely studied in the literature regarding their functional properties (e.g., liveness, boundedness, siphon computation, deadlock prevention, and avoidance techniques) [3]–[5]. Evaluating liveness in  $S^4PR$  nets may

help, for instance, to detect errors in the manufacturing system design rather than in the operational stage, thus saving costs.

However, to the best of our knowledge, no techniques have been proposed regarding their nonfunctional properties—more specifically, regarding their throughput. Achieving a good throughput (defined as jobs completed per unit of time) is usually one of the most important requirements of these systems: the better the throughput is, the greater the number of cars assembled (or the greater the number of packages delivered, or the greater the number of users attended to; recalling the aforementioned domains). Furthermore, knowing how a system performs (i.e., what its throughput is) is a mandatory step for predicting deliveries, item storage, or even optimizing associated industrial processes [6]. Evaluating system throughput also helps to evaluate how system efficiency varies with respect to manufacturing flexibility dimensions and to choose the most suitable, effective design among several alternatives [6]–[9].

In practice, the increasing size of systems makes the exact computation of their performance (i.e., throughput) a highly complex computational task. The main reason for this complexity is the well-known state explosion problem. Hence, a task that requires an exhaustive state-space exploration becomes unachievable in a reasonable time for large systems. To avoid the necessity of calculating the whole state space, approaches that provide performance bounds can be used.

In this brief, we propose an iterative strategy to approximate the throughput of an RAS, modeled by  $S^4PR$  nets. The strategy makes use of mathematical programming problems for which polynomial complexity algorithms exist—thus offering a good tradeoff between accuracy and computational complexity—and of the exact solution of underlying continuous-time Markov chain (CTMC) models of subsystems much smaller than the whole original model.

Based on previous works regarding the computation of upper throughput bounds [10], [11] and similar to ideas already introduced to calculate upper throughput bounds in certain subtypes of PNs such as marked graphs (MGs) [12] and process PNs (PPNs) [9], our strategy works as follows. For a given  $S^4PR$  net and tolerance, our iterative strategy computes as a first step the slowest  $P$ -semiflow of the system (i.e., the *bottleneck*), by computing the upper throughput bound of a subset of transitions. Then, in each iteration step, the next  $P$ -semiflow most likely to be constraining the current bottleneck is computed. This  $P$ -semiflow is taken as the new bottleneck of the system, and the exact throughput of the subnet generated by the new and the previous bottleneck is calculated by solving the underlying CTMC. When tolerance is not achieved, another iteration step will be performed. Note that in each iteration step, the bottleneck of the system is regrown. Note also that the throughput of the transition in the subnet is lower or higher than its real throughput, considering the full system. Hence, the values of throughput computed in each iteration step approximate to the real throughput.

This brief is organized as follows. Section II describes the related work. Section III recalls the basic concepts and definitions regarding PNs. Section IV introduces the iterative heuristic to approximate throughput values in  $S^4PR$  nets.

Section V evaluates the effectiveness of our heuristic by applying it to a set of randomly generated  $S^4PR$  nets. Finally, Section VI sets out the conclusions of this brief and proposes future work.

## II. RELATED WORK

A substantial number of works are found in the literature on  $S^4PR$ , a subclass of PNs suitable for modeling RASs [1]. Nevertheless, research into this class of nets is mainly focused on functional properties, such as liveness, boundedness, and siphon computation [3], [4], or (more widely) deadlock prevention and avoidance techniques [5]. The reader is referred to [2] for a recent survey on the topic. To the best of our knowledge, no specific results on performance evaluation have been proposed for this PN subclass in particular.

Concerning performance estimation of general stochastic PN classes (with negative exponentially distributed service times of transitions), the first results were obtained by means of numerical solutions of the isomorphic CTMC [7]. Later, other exact techniques were achieved, such as those based on *product-form solutions*, but these are valid only for very restricted net subclasses [13], [14]. An alternative approach for the exact analysis was based on compact matrix representations of the infinitesimal generator matrix of the CTMC [15].

In any way, in most cases, the exact computation of performance indexes suffers from the well-known *state explosion problem* that makes the evaluation of large systems intractable; thus, alternative and more efficient bounding or approximation techniques have been proposed. There are techniques strongly inspired by exact solutions, such as approximate *mean value analysis* based on product-form solution [13], [16], or iterative approximation techniques based on compact matrix representations of the CTMC [17]–[19].

Other approximation techniques were inspired by classic queuing networks (QNs) approaches, such as flow equivalent aggregation [20] or iterative Marie’s methods [21]–[23]. More recently, continuous PNs provided new techniques to approximate the performance of discrete models based on *fluidification* [24], [25].

Techniques for computing system performance were also proposed for automated manufacturing systems. In [26], resource-based nets are analyzed assuming known production ratios, structurally enforcing visit ratios. In [27], the minimum cycle time of a  $S^4PR$  is computed assuming that deterministic delays are associated with places and that every event is equally likely. In contrast, this brief is more general, since it deals with stochastic timing associated with transitions and does not assume any equally likely events nor enforces any particular production ratios.

Another approach to performance estimation is that based on bottleneck computation expressed in terms of mathematical programming problems. Putting together structure theory of PNs (basically, the  $P$ -semiflows and the state equation) and queuing systems basics (*Little’s law* and other operational analysis relationships), an efficient technique for the computation of upper and lower bounds for general timed/stochastic PNs was introduced in [28] using only first-order moments of random variables, later improved using also second-order

moments [29], and extended to *interval time PNs* [11], [30]. Second-order moments are also used in [8] to compute performance bounds for a particular problem domain modeled with shared resources nets.

The strong point of this structural approach is its very low computational time (theoretically polynomial on the net size, since it is based on the solution of proper linear programming problems (LPPs) for which polynomial complexity algorithms exist). On the other hand, there is a price to be paid for estimating throughput by these LPPs: the obtained throughput is not so tight (i.e., the upper throughput bound is really distant from real throughput). Therefore, these bounds cannot be used in many cases as a good throughput approximation.

The loose estimation provided by bounds was also detected in the seventies in the area of queuing systems with regard to classic bottleneck bounding techniques. Thus, several schemes for the construction of *hierarchies of bounds* for QNs were developed that guaranteed any level of accuracy (including the exact solution), by investing the necessary computational effort: performance bound hierarchies [31], successively improving bounds [32], and generalized quick bounds [33]. All these techniques were derived from the *mean value theorem*, thus being valid only for product-form QNs and unsuitable for general PNs.

A similar idea (but using different techniques) was proposed in the setting of stochastic PNs for the improvement of LPP-based upper throughput bounds for the particular subclasses of *MGs* [12] and *process PNs* [9]. The basic idea behind this improvement, referred to as *bottleneck regrowing*, was an iterative procedure that initially considered the most constraining subnet (generated by the slowest *P-semiflow*) of the system and then adding other subnets to it in each iteration.

The same idea is developed in this brief, but considering a more general net subclass. Thus, we try to extend the bottleneck regrowing technique, initially presented for MGs or PPNs, to general RASs modeled with  $S^4PR$ .

### III. PRELIMINARY CONCEPTS AND DEFINITIONS

This section first introduces untimed PNs and  $S^4PR$  nets. Then, the notion of time in PNs is presented.

#### A. Untimed Petri Nets

*Definition 1:* A PN [34] is a 4-tuple  $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ , where  $P$  and  $T$  are disjoint nonempty sets of *places* and *transitions* ( $|P| = n$ ,  $|T| = m$ ) and  $\mathbf{Pre}$  ( $\mathbf{Post}$ ) are the pre-(post)-incidence nonnegative integer matrices of size  $|P| \times |T|$ .

The *preset* and *postset* of a node  $v \in P \cup T$  are, respectively, defined as  $\bullet v = \{u \in P \cup T | (u, v) \in F\}$  and  $v \bullet = \{u \in P \cup T | (v, u) \in F\}$ , where  $F \subseteq (P \times T) \cup (T \times P)$  is the set of directed arcs. *Ordinary* nets are PNs whose arcs have weight 1. The *incidence matrix* of a PN is defined as  $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ .

A vector  $\mathbf{m} \in \mathbb{Z}_{\geq 0}^{|P|}$ , which assigns a nonnegative integer to each place, is called *marking vector* or *marking*.

*Definition 2:* A *PN system*, or *marked PN*  $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$ , is a PN  $\mathcal{N}$  with an *initial marking*  $\mathbf{m}_0$ .

A transition  $t \in T$  is *enabled* at marking  $\mathbf{m}$  if  $\mathbf{m} \geq \mathbf{Pre}(\cdot, t)$ , where  $\mathbf{Pre}(\cdot, t)$  is the column of  $\mathbf{Pre}$  corresponding to transition  $t$ . A transition  $t$  enabled at  $\mathbf{m}$  can *fire* yielding a new marking  $\mathbf{m}' = \mathbf{m} + \mathbf{C}(\cdot, t)$  (*reached marking*). This is denoted by  $\mathbf{m} \xrightarrow{t} \mathbf{m}'$ . A sequence of transitions  $\sigma = \{t_i\}_{i=1}^n$  is a *firing sequence* in  $\mathcal{S}$  if there exists a sequence of markings, such that  $\mathbf{m}_0 \xrightarrow{t_1} \mathbf{m}_1 \xrightarrow{t_2} \mathbf{m}_2 \dots \xrightarrow{t_n} \mathbf{m}_n$ . In this case, marking  $\mathbf{m}_n$  is said to be *reachable* from  $\mathbf{m}_0$  by firing  $\sigma$ , and this is denoted by  $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}_n$ . The *firing count vector*  $\boldsymbol{\sigma} \in \mathbb{Z}_{\geq 0}^{|T|}$  of the firable sequence  $\sigma$  is a vector, such that  $\boldsymbol{\sigma}(t)$  represents the number of occurrences of  $t \in T$  in  $\sigma$ . If  $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}$ , then we can write in the vector form  $\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}$ , which is referred to as the *linear* (or *fundamental*) *state equation* of the net. The number of times that a transition  $t$  enabled at marking  $\mathbf{m}$  can fire before becoming disabled is called its *enabling degree* and computed as  $\max\{k \in \mathbb{Z}_+ | \mathbf{m} \geq k \cdot \mathbf{Pre}(\cdot, t)\}$ . A transition  $t$  is *persistent* if once it becomes enabled, it will eventually be fired.

The set of markings *reachable* from  $\mathbf{m}_0$  in  $\mathcal{N}$  is denoted as  $RS(\mathcal{N}, \mathbf{m}_0)$  and is called the *reachability set*. A PN system  $\langle \mathcal{N}, \mathbf{m}_0 \rangle$  is *reversible* if for each marking  $\mathbf{m} \in RS(\mathcal{N}, \mathbf{m}_0)$ ,  $\mathbf{m}_0$  is reachable from  $\mathbf{m}$ .

Two transitions  $t$  and  $t'$  are said to be in *structural conflict* if they share, at least, one input place, i.e.,  $\bullet t \cap \bullet t' \neq \emptyset$ . Two transitions  $t$  and  $t'$  are in *free conflict* if  $\mathbf{Pre}(\cdot, t) = \mathbf{Pre}(\cdot, t') \neq \mathbf{0}$ , where  $\mathbf{0}$  is a vector with all entries equal to zero. Two transitions  $t$  and  $t'$  are said to be in (*effective*) *conflict for a marking*  $\mathbf{m}$  if the firing of  $t$  decreases the enabling degree of  $t'$  in  $\mathbf{m}$  (i.e., each token must decide which way to go) [35].

A transition  $t$  is *live* if it can be fired from every reachable marking. A system is *live* when every transition is live. A net is *structurally live* if there exists an initial marking making it live. A system is *bounded* if and only if its reachability set is finite. A net is *structurally bounded* if and only if it is bounded, regardless of the initial marking.

A *P-semiflow* (*T-semiflow*) is a nonnegative integer vector  $\mathbf{y} \geq \mathbf{0}$  ( $\mathbf{x} \geq \mathbf{0}$ ), such that it is a left (right) anuller of the net's incidence matrix,  $\mathbf{y}^\top \cdot \mathbf{C} = \mathbf{0}$  ( $\mathbf{C} \cdot \mathbf{x} = \mathbf{0}$ ). A *P-semiflow* implies a token conservation law independent of any firing of transitions. A *P-* (or *T-*)*semiflow*  $\mathbf{v}$  is *minimal* when its support,  $\|\mathbf{v}\| = \{i | v(i) \neq 0\}$ , is not a proper superset of the support of any other *P-* (or *T-*)*semiflow*, and the greatest common divisor of its elements is one.

A *state machine* is a particular type of ordinary PN where each transition has exactly one input arc and exactly one output arc, that is,  $|\bullet t| = |t \bullet| = 1, \forall t \in T$ .

#### B. $S^4PR$ for the Modeling of RAS

$S^4PR$  nets are a subclass of PN used to model RAS. Production plans are modeled by means of strongly connected state machines with no internal cycles, which share a set of nonconsumable, reusable resources. An example of  $S^4PR$  was described in Section I. A gentle introduction and survey of results for  $S^4PR$  in the context of RAS is given in [2].

*Definition 3:* [2] Let  $I_{\mathcal{N}}$  be a finite set of indices. An  $S^4PR$  net is a connected generalized pure P/T net  $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$  where it has the following.

- 1)  $P = P_0 \cup P_S \cup P_R$  is a partition, such that  $P_0 = \bigcup_{i \in I_N} \{p_{0i}\}$  is the set of *process-idle places*,  $P_S = \bigcup_{i \in I_N} P_i$ , where  $\forall i \in I_N : P_i \neq \emptyset$ , and  $\forall i, j \in I_N, j \neq i, P_i \cap P_j = \emptyset$ , is the set of *process-activity places*, and  $P_R, |P_R| > 0$ , is the set of *resources places*.
- 2)  $T = \bigcup_{i \in I_N} T_i$ , where  $\forall i \in I_N : T_i \neq \emptyset$ , and  $\forall i, j \in I_N, j \neq i, T_i \cap T_j = \emptyset$ ;
- 3) For each  $r \in P_R$ , there exists a unique minimal  $P$ -semiflow  $\mathbf{y}_r \in \mathbb{N}^{|P|}$ , such that  $\|\mathbf{y}_r\| \cap P_R = \{r\}$ ,  $\|\mathbf{y}_r\| \cap P_S \neq \emptyset$ ,  $\|\mathbf{y}_r\| \cap P_0 = \emptyset$ , and  $\mathbf{y}_r(r) = 1$ . This establishes how each resource is reused, that is, they cannot be created nor destroyed.
- 4)  $P_S = \bigcup_{r \in P_R} (\|\mathbf{y}_r\| \setminus \{r\})$ .

Note that there should be an initial marking in which there are enough free resource instances, such that every production plan in the net is realizable. More formally, a vector  $\mathbf{m}_0 \in \mathbb{Z}_{\geq 0}^{|P|}$  is an *acceptable initial marking* of an  $S^4PR$  net iff  $\|\mathbf{m}_0\| = P_0 \cup P_R$  and  $\forall p \in P_S, r \in P_R : \mathbf{m}_0(r) \geq \mathbf{y}_r(p)$  [2].

Let us also note that every  $S^4PR$  net is structurally live and structurally bounded, and if a  $S^4PR$  net is live for an acceptable initial marking, then the system is reversible [2].

### C. Stochastic Petri Nets

Since our goal is to compute performance in  $S^4PR$ , we need to introduce the notion of time. PNs have been extended in the literature with suitable time interpretations for the modeling and timing prediction of real-time systems or for performance evaluation. In this regard, *time PNs* reduce nondeterminism in the duration of activities by associating a time interval with each transition, being also extended with time intervals to reduce the nondeterminism inherent in conflict resolution for sets of transitions in equal-conflict relation [11]. In *timed PNs* [36], the duration of activities is considered as constant, thus they are in some way interpreted as a particular case of (interval) time PNs in which the lower and upper limits become a single value. Similarly, *stochastic PNs* [7] model durations with (usually negative exponential) random variables and firing probabilities of simultaneously enabled transitions are solved either using race policy (between timed transitions) or firing probabilities defined by weights or ratios (between immediate transitions).

In this brief, we consider that an exponentially distributed delay is associated with each transition in the net, as in generalized stochastic PNs [7], since for these models, an underlying CTMC exists that provides powerful numerical analysis techniques. More formally, every transition  $t \in T$  fires following an exponential distribution with mean service time  $\delta(t) \in \mathbb{R}_{\geq 0}$ . If  $\delta(t) > 0$ , then transition  $t$  is a *timed transition*. Otherwise, transition  $t$  is an *immediate transition* (i.e., it fires in zero time).

To decouple activity duration times from resource acquisition/release, we assume that all transitions connected to resources, as well as free conflicts in process-related nets, are immediate and conflict resolution is based on ratios, as in generalized stochastic PNs [7]. An immediate transition  $t$  in conflict will fire with probability  $(\mathbf{r}(t)/\sum_{t' \in A} \mathbf{r}(t'))$ , where  $A$  is the set of enabled immediate transitions in conflict and  $\mathbf{r}(t)$  is the ratio of transition  $t$ .

From a modeling point of view, the above assumption restricts us to modeling the preemption of a given process activity due to the resource consumption by another process. Nonetheless, such a constraint also allows us to use linear programming (LP)-based techniques for the computation of throughput bounds that assume persistent (i.e., nonpreemptive) transitions.

Regarding firing semantics, we assume infinite server semantics, since this is the firing semantics assumed by the linear programming-based techniques used for throughput bound computation. Furthermore, single-server semantics can be modeled by adding a self-loop place with a single token.

In this brief, we assume that the  $S^4PR$  under study is live, and thus, as previously mentioned, it is also reversible. Hence, as stated in [7], the underlying CTMC is ergodic. Therefore, the steady-state distribution serves as a basis for the quantitative evaluation of an  $S^4PR$  in terms of performance indices, particularly the expected value of the number of tokens in a given place (*average marking*) and the mean number of firings of a transition per unit time (*throughput*). We denote the average marking vector as  $\bar{\mathbf{m}}$  and the steady-state throughput vector as  $\chi$ .

The vector of visit ratios expresses the relative throughput of transitions in the steady state. The visit ratio  $\mathbf{v}(t)$  of each transition  $t \in T$  normalized for transition  $t_i$ ,  $\mathbf{v}_{t_i}(t)$ , is expressed as follows:  $\mathbf{v}_{t_i}(t) = \chi(t)/\chi(t_i) = \Gamma(t_i) \cdot \chi(t)$ ,  $\forall t \in T$  where  $\Gamma(t_i) = 1/\chi(t_i)$  represents the *average interfering time* of transition  $t_i$ .

By token flow balance, it is straightforward to see that the vector  $\mathbf{v}_{t_i}$  must be a  $T$ -semiflow, i.e.,  $\mathbf{C} \cdot \mathbf{v}_{t_i} = \mathbf{0}$ . Besides,  $\mathbf{v}_{t_i}$  must also satisfy the routing constraints at free conflicts. Let  $t$  and  $t'$  be two immediate transitions in free conflict. Then,  $\mathbf{r}(t) \cdot \mathbf{v}_{t_i}(t') = \mathbf{r}(t') \cdot \mathbf{v}_{t_i}(t)$ . All these routing constraints can be expressed in matrix form as  $\mathbf{R} \cdot \mathbf{v}_{t_i} = \mathbf{0}$ , where  $\mathbf{R}$ , termed *routing matrix*, has one row per each pair of  $t$  and  $t'$  in free conflict.

## IV. ITERATIVE PERFORMANCE APPROXIMATION TECHNIQUE FOR $S^4PR$ NETS

There are several approaches to computing performance in PNs. The tradeoff between computational complexity and accuracy is reflected in the computation of throughput bounds (fast, but not so accurate) or in the analytical performance computation (very accurate, but probably infeasible for large systems due to the state-space explosion problem).

Our approach falls between both limits. In particular, it consists of an iterative method for approximating throughput for  $S^4PR$  nets based on upper throughput bounds and using first-order moments as in [28], [30], and [36] (it might be extended to use second-order moments, as in [8] and [29]).

Performance bounds as proposed in [11], [28], and [30] are valid for arbitrary (structurally defined) PN subclasses (such as MGs, free choice nets, or  $S^4PR$ , to name a few), as well as for arbitrary time interpretations (timed, time, or stochastic PNs). These bounds are based on structural information (incidence matrix and initial marking) and also on the timing specification of the model, namely, the average duration of activities (timed or stochastic PNs) or the

time interval limits (time PNs), and the weights or rates specifying the resolution of free conflicts (i.e., transitions in equal conflict relation). Using this structural information implies that the obtained bounds are tight (i.e., closer to real throughput) for net subclasses without nonfree conflicts (such as MGs or free choice nets) or for other net subclasses with the presence of nonfree conflicts (such as freely related T-semiflow nets [10] or PPNs [9]), in which the structural interaction between free conflicts and  $T$ -semiflows (fixed by the net incidence matrix) ensures the existence of a unique vector of visit ratios (or relative throughput of transitions, as defined in Section III-C).

Beyond these particular net subclasses, the quality of the structurally computed throughput bounds can be significantly reduced, especially when used as a throughput approximation. The main novelty of this brief is to compute a better throughput approximation for the  $S^4PR$  net subclass, where the vector of visit ratios cannot be efficiently computed from the net structure due to nonfree conflicts involving resource places. Thus, we combine the structural technique for the computation of bounds presented in [11] with the numerical solution of the underlying CTMC of some appropriate subnets, using similar ideas to those presented in [9] for PPNs, that can be nontrivially extended to the more general  $S^4PR$  net subclass.

Specifically, we apply the optimization problem introduced in [11] for general nets to  $S^4PR$  nets, which is as follows:

$$\begin{aligned} \mathcal{P}_1 : \quad & \Gamma(t_i) \geq \Gamma^{\text{lb}}(t_i) \\ & = \min_{\mathbf{v}_i \in \text{dom}_{\mathbf{v}}} \max_{\mathbf{y} \in \text{dom}_{\mathbf{y}}} \mathbf{y} \cdot \mathbf{Pre} \cdot \mathbf{D}^{\mathbf{t}_i} \\ & \text{subject to } \text{dom}_{\mathbf{y}} : \{ \mathbf{y}^{\top} \cdot \mathbf{C} = \mathbf{0}, \mathbf{y} \cdot \mathbf{m}_0 = 1, \mathbf{y} \geq \mathbf{0} \} \\ & \quad \text{dom}_{\mathbf{v}} : \{ \mathbf{R} \cdot \mathbf{v}_i = \mathbf{0}, \mathbf{C} \cdot \mathbf{v}_i = \mathbf{0} \\ & \quad \mathbf{v}_i \geq \mathbf{0}, \mathbf{v}_i(t_i) = 1 \} \end{aligned}$$

where  $\mathbf{D}^{\mathbf{t}_i} = \delta \odot \mathbf{v}_i$  (componentwise product) is the vector of average service demands of transitions and  $\mathbf{R}$  is the routing matrix.

Recall that the lower bound  $\Gamma^{\text{lb}}(t_i)$  computed as a solution of  $\mathcal{P}_1$  is the inverse of the upper throughput bound of  $t_i$ , i.e.,  $\chi^{\text{ub}}(t_i) = \frac{1}{\Gamma^{\text{lb}}(t_i)}$ . Besides,  $\mathcal{P}_1$  has two (free) variables: the  $P$ -semiflow  $\mathbf{y}$  and the vector of visit ratios  $\mathbf{v}$ . Hence, the slowest  $P$ -semiflow of the net  $\mathbf{y}^*$  and its associated vector of visit ratio  $\mathbf{v}_i^*$  are also obtained by solving  $\mathcal{P}_1$ .

The domain of  $\mathbf{y}$ ,  $\text{dom}_{\mathbf{y}}$ , defines the constraints regarding  $\mathbf{y}^*$ . In particular, these constraints impose that  $\mathbf{y}^*$  is in fact a  $P$ -semiflow of the system. The interpretation of these constraints is that the problem is computing the minimum cycle time of all subnets generated by each  $P$ -semiflow, if they are considered in isolation. Similarly,  $\text{dom}_{\mathbf{v}}$  defines the constraints regarding  $\mathbf{v}^*$ . These constraints impose that the vector of visit ratios  $\mathbf{v}^*$  is a solution to both the linear system of equations  $\mathbf{C} \cdot \mathbf{v}_i = \mathbf{0}$  (token-flow balance) and  $\mathbf{R} \cdot \mathbf{v}_i = \mathbf{0}$  (free-conflict routing).

The objective function of  $\mathcal{P}_1$  represents the cycle time of a subsystem generated by a  $P$ -semiflow, considered in

isolation (that is, the cycle time is the weighted sum of the mean service times of transitions that belong to such a subsystem).

Let us compute the upper throughput bound for every transition in the running example using  $\mathcal{P}_1$ :  $\chi^{\text{ub}}(t_2) = \chi^{\text{ub}}(t_5) = 0.3125$ ,  $\chi^{\text{ub}}(t_8) = \chi^{\text{ub}}(t_{11}) = \chi^{\text{ub}}(t_{14}) = 0.5$ , and  $\chi^{\text{ub}}(t_{17}) = 1$ . However, if we analytically compute the throughput of the net (i.e., by solving the CTMC):  $\chi(t_2) = 0.275275$ ,  $\chi(t_5) = 0.275248$ ,  $\chi(t_8) = 0.288146$ ,  $\chi(t_{11}) = 0.288056$ ,  $\chi(t_{14}) = 0.287948$ , and  $\chi(t_{17}) = 0.675903$ . Note that the upper throughput bound ranges from an error of 12% to 42% with regard to the real throughput. Hence, the relative error of the upper throughput bound may be considerable.

As stated before, this (bad) result may be consistent with the fact that structural bounds, as proposed in [11], are unable to incorporate the relative throughput between transitions not related by a  $T$ -semiflow and involved in nonfree conflict, as occurs with transitions acquiring resources in  $S^4PR$  nets (e.g., transitions  $t_2$  and  $t_8$  in the running example).

In this brief, we investigate the idea of *net regrowing* to obtain a better approximation of throughput values in  $S^4PR$  nets. Considering initially  $\mathbf{y}^*$  (obtained as the result of  $\mathcal{P}_1$ ), which is the slowest  $P$ -semiflow of the net (i.e., the bottleneck), we will compute the next  $P$ -semiflow most likely to be constraining the system. The subnet generated by such a  $P$ -semiflow plus the bottleneck is taken as the basis for throughput approximation.

Let  $\mathbf{y}^*$  be a solution of  $\mathcal{P}_1$ . To compute the next  $P$ -semiflow most likely to be constraining the system, we use a similar approach as in [9]. Hence, we add two constraints to  $\text{dom}_{\mathbf{y}}$ :  $\mathbf{y}(p) > 0, \forall p \in \mathcal{Q}, \mathcal{Q} = \|\mathbf{y}^*\|$  and  $\sum_{p \in V} \mathbf{y}(p) > 0$ , where  $V = \{v | v \in \bullet(\mathcal{Q}^*) \setminus \mathcal{Q}\}$ . These new constraints in  $\text{dom}_{\mathbf{y}}$  impose that the solution of the optimization problem will be some (nonminimal)  $P$ -semiflow composed of the previous  $\mathbf{y}^*$  plus another  $P$ -semiflow, which is connected to  $\mathbf{y}^*$  through some transition (we refer to this technique as *bottleneck regrowing*).

The strict inequality  $\sum_{p \in V} \mathbf{y}(p) > 0$  may lead, however, to numerical problems (the lower the value of  $\sum_{p \in V} \mathbf{y}(p)$ , the higher the optimization function value). Hence, this inequality is transformed into  $\sum_{p \in V} \mathbf{y}(p) \geq h^*$ , where  $h^*$  is a strictly positive value that ensures the feasibility of the following optimization problem. A valid value for  $h^*$  is computed by the LPP introduced in [9] as follows:  $h^* = \max\{h | \mathbf{y} \cdot \mathbf{C} = \mathbf{0}, \mathbf{y} \cdot \mathbf{m}_0 = 1, \mathbf{y} \geq h \cdot \mathbf{1}, h > 0\}$ . Putting all together, the optimization problem  $\mathcal{P}_2$  is defined as:

$$\begin{aligned} \mathcal{P}_2 : \quad & \Gamma(t_i) \geq \Gamma^{\text{lb}}(t_i) \\ & = \min_{\mathbf{v}_i \in \text{dom}_{\mathbf{v}}} \max_{\mathbf{y} \in \text{dom}_{\mathbf{y}}} \mathbf{y} \cdot \mathbf{Pre} \cdot \mathbf{D}^{\mathbf{t}_i} \\ & \text{subject to } \text{dom}_{\mathbf{y}} : \{ \mathbf{y}^{\top} \cdot \mathbf{C} = \mathbf{0}, \mathbf{y} \cdot \mathbf{m}_0 = 1, \\ & \quad \mathbf{y}(p) \geq 0, \forall p \in \mathcal{Q}, \\ & \quad \sum_{p \in V} \mathbf{y}(p) \geq h^*, h^* > 0, \mathbf{y} \geq \mathbf{0} \} \\ & \quad \text{dom}_{\mathbf{v}} : \{ \mathbf{R} \cdot \mathbf{v}_i = \mathbf{0}, \mathbf{C} \cdot \mathbf{v}_i = \mathbf{0}, \\ & \quad \mathbf{v}_i \geq \mathbf{0}, \mathbf{v}_i(t_i) = 1 \}. \end{aligned}$$

**Algorithm 1** Performance Approximation in  $S^4PR$  Nets

---

**Input:**  $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle, \varepsilon$   
**Output:**  $\Theta, \mathbf{y}$

- 1 Compute the set  $T' = \{t | t \in p^\bullet = p_r^\bullet, p \in P_S, p_r \in P_R\}$
- 2 Compute  $\langle \Gamma_t^{\text{lb}}(t), \mathbf{y}_t^* \rangle = \mathcal{P}_1(t), \forall t \in T'$
- 3 Select  $t_{\min} \in T'$  such that  $\Gamma_{t_{\min}}^{\text{lb}}(t_{\min}) = \max_{\forall t \in T'} \Gamma_t^{\text{lb}}(t)$
- 4  $\chi(t) = \frac{1}{\Gamma_{t_{\min}}^{\text{lb}}(t)}, \forall t \in T; \mathbf{y}^* = \mathbf{y}_{t_{\min}}^*$ ;  
 $\Theta = \chi(t_{\min}); \Theta' = 0$
- 5 Compute  
 $h^* = \max\{h | \mathbf{y} \cdot \mathbf{C} = \mathbf{0}, \mathbf{y} \cdot \mathbf{m}_0 = 1, \mathbf{y} \geq h \cdot \mathbf{1}, h > 0\}$
- 6 **while**  $\frac{\Theta - \Theta'}{\Theta} \geq \varepsilon$  **and**  $(\|\mathbf{y}^*\| \neq P)$  **do**
- 7     Compute  $\langle \mathbf{y}, \Gamma_{t_{\min}}^{\text{lb}} \rangle$  using  $\mathcal{P}_2(h^*, t_{\min}, \mathbf{y}^*)$
- 8     Compute  $\chi$  of the net generated by the  $P$ -semiflow  $\mathbf{y}$
- 9      $\Theta' = \Theta; \Theta = \chi(t_{\min}); \mathbf{y}^* = \mathbf{y}$
- 10 **end**

---

Hence, given a transition  $t_i$  and a bottleneck  $\mathbf{y}^*$  computed by means of  $\mathcal{P}_1$ , the solution of  $\mathcal{P}_2$  provides a new bottleneck  $\mathbf{y}^{*'}$  of the net, which will be a linear combination of the previous  $\mathbf{y}^*$  and the next  $P$ -semiflow most likely to be constraining the system connected to  $\mathbf{y}^*$  through some transition.

The above process can be iteratively repeated, considering a transition  $t_i$ , to regrow the initial bottleneck until obtaining the full net. Note that  $\chi^{*'}(t_i) \leq \chi^{\text{ub}}(t_i)$ , but the throughput of the subnet generated by  $\mathbf{y}^{*'}$  in each step,  $\chi^{*'}$ , will be an approximation of the real throughput of  $t_i$ ,  $\chi(t_i)$ .

Algorithm 1 implements an iterative strategy for approximating throughput in live, bounded  $S^4PR$  nets. As input, it needs the  $S^4PR$  to be analyzed, i.e.,  $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$ , and the tolerance ( $\varepsilon > 0$ ) to be achieved. As output, it provides the approximate throughput  $\Theta$  and its associated  $P$ -semiflow.

Step 1 computes the set of transitions  $T'$ , such that their firing acquires some resource  $r \in \mathcal{R}$ , i.e.,  $t \in p^\bullet = p_r^\bullet, p \in P_S, p_r \in P_R$ . Step 2 computes, for each transition  $t \in T'$ , the lower bound  $\Gamma_t^{\text{lb}}(t)$  and its associated  $P$ -semiflow  $\mathbf{y}_t^*$  using  $\mathcal{P}_1$ . Then, the transition  $t$ , which has the maximum lower bound  $\Gamma_t^{\text{lb}}(t)$ , is selected as  $t_{\min}$ . The regrowing method will consider  $t_{\min}$  as the reference transition. Note that  $t_{\min}$  may also be previously selected by the user if there is some interest in approximating throughput for a certain transition. Step 4 refers to the setup of local variables. Step 5 computes the value  $h^*$  that ensures feasibility of  $\mathcal{P}_2$ , as in [9].

Steps 6–10 represent the iterative strategy for approximating throughput values. A new iteration is performed when the relative error of throughputs between consecutive iterations is greater than or equal to the tolerance to be achieved and when there are places not covered by the current  $P$ -semiflow  $\mathbf{y}^*$ . Step 7 uses  $\mathcal{P}_2$  to compute the next  $P$ -semiflow  $\mathbf{y}$ , which will be a linear combination of the current  $\mathbf{y}^*$  plus the next  $P$ -semiflow most likely to be constraining the system and connected to  $\mathbf{y}^*$ . Step 8 computes the throughput of the net generated by  $\mathbf{y}$  solving the underlying CTMC analytically, when feasible; or by simulation, otherwise. The last step updates the iteration variables accordingly.

TABLE I

RESULTS OF ALGORITHM 1 APPLIED IN THE RUNNING EXAMPLE (CONSIDERING  $t_{\min} = t_2$  AND  $\varepsilon = 0$ ). THROUGHPUT VALUES ARE PER  $10^{-2}$

$ P $	$ T $	$P$ -semiflow	$\chi(t_2)$	$\%_{\text{real}}$
7	7	$\mathbf{y}_0$	31.2500	-13.77%
10	10	$\mathbf{y}_0 \cup \{p_9, p_{10}, p_{11}\}$	24.3500	11.35%
17	16	$\mathbf{y}_0 \cup \mathbf{y}_1 \cup \{p_9\}$	26.9239	1.98%
20	19	$\mathbf{y}_0 \cup \mathbf{y}_1 \cup \{p_9, p_{19}, p_{20}, p_{21}\}$	27.7512	-1.04%
22	20	$\mathbf{y}_0 \cup \mathbf{y}_1 \cup \mathbf{y}_2 \cup \{p_9, p_{19}\}$	27.5534	-0.32%
23	20	$\mathbf{y}_0 \cup \mathbf{y}_1 \cup \mathbf{y}_2 \cup \{p_8, p_9, p_{19}\}$	27.4665	-

Let us illustrate how Algorithm 1 works by means of the running example. Recall that the net is composed of three processes, whose supports are, respectively:  $\|\mathbf{y}_0\| = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$ ,  $\|\mathbf{y}_1\| = \{p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}\}$ , and  $\|\mathbf{y}_2\| = \{p_{20}, p_{21}, p_{22}, p_{23}\}$ , while the set of resource places is  $\mathcal{R} = \{p_8, p_9, p_{19}\}$ . The set of transitions  $T'$  is equal to  $T' = \{t_2, t_5, t_8, t_{11}, t_{14}, t_{17}\}$  whose lower cycle time bounds are, respectively,  $\Gamma_t^{\text{lb}} = \{3.2, 3.2, 2, 2, 2, 1\}$ . In this case, the transition having the maximum lower cycle time bound is  $t_{\min} = t_2$ , since  $\Gamma^{\text{lb}}(t_2) = 3.2$ . Hence,  $\Theta = 0.3125$  and  $\mathbf{y}^* = \mathbf{y}_0$ . Step 5 computes the value of  $h^* = 0.05555$ .

Consider  $\varepsilon = 10\%$ . Since the iteration condition is fulfilled, a regrowing step is performed. Using  $\mathcal{P}_2(h^*, t_{\min}, \mathbf{y}^*)$ , the next constraining  $P$ -semiflows  $\mathbf{y} = \mathbf{y}_0 \cup \{p_9, p_{10}, p_{11}\}$  and  $\Gamma^{\text{lb}}(t_2) = 3.0667$  are obtained as a solution. The throughput of  $t_2$  is computed in the next step solving the underlying CTMC giving as a solution  $\chi(t_2) = 0.243501$ . Then, the variables are properly updated. Since the relative throughput error is 28.34% and the regrowing can continue (i.e., the current  $P$ -semiflow does not contain all the places), another iteration step is carried out. The second step provides as the solutions of  $\mathcal{P}_2$  the  $P$ -semiflow  $\mathbf{y} = \mathbf{y}_0 \cup \mathbf{y}_1 \cup \{p_9\}$  and  $\Gamma^{\text{lb}}(t_2) = 3.4045$ . The throughput of  $t_2$  in the subnet generated by  $\mathbf{y}$  is  $\chi(t_2) = 0.2692396$ , i.e., the relative throughput error is 9.56%.

Hence, the algorithm stops after a few steps and provides the approximated throughput  $\Theta = 0.2692396$  and its associated  $P$ -semiflow  $\mathbf{y} = \mathbf{y}_0 \cup \mathbf{y}_1 \cup \{p_9\}$ . The size of the subnet generated by  $\mathbf{y}$  is  $|P| = 17$  places and  $|T| = 16$  transitions. Note that the real throughput of  $t_2$  is  $\chi(t_2) = 0.274665$ , i.e., the approximated throughput is 1.98% lower than  $\chi(t_2)$ .

Table I shows the results considering  $t_2$  and  $\varepsilon = 0$ . Each row represents an iteration step, indicating the subnet size (a number of places and transitions), the associated  $P$ -semiflow, its computed throughput, and the relative error with regard to the real throughput value. The last (highlighted) row corresponds to the last step, in which the full net is considered.

Regarding convergence, the algorithm stops, since in each iteration step, the  $P$ -semiflow regrows until all the places are covered. Regarding accuracy, if we keep regrowing until all the places are covered, then the accuracy error is zero (since we consider all the net). However, we cannot reach any conclusions about the accuracy error in general terms. In this regard, in Section V, we randomly create  $S^4PR$  nets to evaluate how Algorithm 1 performs.

TABLE II  
EXPERIMENTAL SETTINGS

Parameter description	Values
No. of processes ( $nPPNs$ )	5, 10
Places per process ( $ P $ )	$\{8, \dots, 12\}, \{20, \dots, 25\}$
Transitions per process ( $ T $ )	$ T  =  P  \cdot \frac{7}{6}$
Timed transition rates	$[1.0, 2.0]$
Initial tokens of process-idle places	$\{2, \dots, 5\}$
No. of resources	$\{nPPNs, nPPNs + \frac{1}{2}\}$
Initial tokens of resource places	$\{2, \dots, 3\}$
Processes in which a resource $r$ is used	$\{2, \dots, 3\}$
Confidence level and simulation accuracy	95%, $\pm 5\%$

## V. EXPERIMENTATION AND DISCUSSION

To evaluate the effectiveness of Algorithm 1, we developed a tool<sup>2</sup> to randomly create  $S^4PR$  nets taking into account various parameters, such as the production plan size, transition rates, number of transitions, resources and available resource copies, and resource sharing between production plans.

We created 24 different  $S^4PR$  nets, classified into small (5 processes) and big (10 processes)  $S^4PR$  nets. Each set was also divided into small (between 8 and 12 places each) and big processes (between 20 and 25 places). The number of transitions of each production plan was one-sixth more than its number of places. The firing time of timed transitions followed an exponential distribution of a randomly selected rate ranging from 1 to 2. The initial marking of processes was randomly selected between 2 and 5. Resources were also randomly added, ranging from the number of processes to one-half more. Similarly, the number of available copies of resources was randomly selected between 2 and 3, and the number of production plans sharing each resource was also randomly selected in the same range. For the sake of readability, Table II summarizes the experimental settings.

Experiments were performed in a GNU/Linux environment running Intel Pentium 4 3.60 GHz with DDR2 RAM 3.0 GiB. Algorithm 1 was implemented as a plug-in of Peabrain [37] (using GLPK v4.55 as an LP solver), whereas throughput computation was carried out using GreatSPN [38]. When feasible, the underlying CTMC was solved; otherwise, the net was simulated with a precision of 5% at the 95% confidence level. The tolerance was set at  $\epsilon = 0.02$ .

Tables III and IV show the experimental results for the small and big  $S^4PR$  nets grouped by the size of the production plan, respectively. For each net, we show its size (places and transitions), the  $\chi(t_{\min})$  throughput of transition  $t_{\min}$  (computed by simulation), the upper throughput bound value (solution of  $\mathcal{P}_1(t_{\min})$ ), the iteration steps performed, the size of the subnet comprising the slowest  $P$ -semiflow given as a solution by Algorithm 1, its throughput, and the relative error with regard to its upper throughput bound.

The results show, on average, an improvement in the upper throughput bound of nearly 20% in almost all cases (in the case of big  $S^4PR$  nets having small processes, the improvement is almost 40%). It should be remarked that the initial upper throughput bound is very tight for some of these nets, whereas it is very distant from the real throughput for others. Note

<sup>2</sup>Released under GPLv3 license and freely available at [http://weediis.unizar.es/~ricardo/?page\\_id=527](http://weediis.unizar.es/~ricardo/?page_id=527).

TABLE III  
EXPERIMENTAL RESULTS: APPROXIMATE THROUGHPUT VALUES (PER  $10^{-2}$ ) OF SMALL  $S^4PR$  NETS, GROUPED BY SIZE OF PPNs.  
(a) SMALL  $S^4PR$  NETS WITH SMALL PROCESSES.  
(b) SMALL  $S^4PR$  NETS WITH BIG PROCESSES

$ P $	$ T $	$\chi(t_{\min})$	$\chi^{ub}(t_{\min})$	Steps	$ P' $	$ T' $	$\Theta$	%
102	127	2.4567	3.7037	6	83	108	2.3850	-55.29%
112	150	5.0247	4.2667	5	70	95	5.0129	14.89%
87	108	6.1678	6.6667	5	54	70	6.2938	-5.92%
92	112	6.8400	9.4787	3	42	53	6.9596	-36.20%
99	138	4.5687	5.4795	2	42	63	4.2352	-29.38%
101	136	2.0001	2.4242	3	40	57	2.0058	-20.86%

(a)

$ P $	$ T $	$\chi(t_{\min})$	$\chi^{ub}(t_{\min})$	Steps	$ P' $	$ T' $	$\Theta$	%
159	244	0.2901	0.5224	3	65	101	0.2799	-86.63%
155	250	4.2619	3.7068	3	63	100	4.3552	14.89%
146	223	4.0878	3.3631	4	62	93	3.8592	12.85%
156	237	1.5909	1.8957	3	64	99	1.5620	-21.37%
158	259	3.9912	3.9683	1	35	54	3.9189	-1.26%
156	236	2.0210	2.4742	6	99	152	1.9586	-26.33%

(b)

TABLE IV  
EXPERIMENTAL RESULTS: APPROXIMATE THROUGHPUT VALUES (PER  $10^{-2}$ ) OF BIG  $S^4PR$  NETS, GROUPED BY SIZE OF PPNs.  
(a) BIG  $S^4PR$  NETS WITH SMALL PROCESSES.  
(b) BIG  $S^4PR$  NETS WITH BIG PROCESSES

$ P $	$ T $	$\chi(t_{\min})$	$\chi^{ub}(t_{\min})$	Steps	$ P' $	$ T' $	$\Theta$	%
181	219	8.0788	8.2759	2	36	46	7.2507	-14.14%
155	202	4.1858	5.3333	3	39	51	4.1076	-29.84%
180	239	3.3381	6.7797	2	22	31	2.9298	-131.40%
170	207	2.3503	3.9216	3	39	47	2.2325	-75.66%
175	219	12.5064	9.1954	3	41	52	10.6163	13.38%
176	216	5.5516	5.4054	2	38	48	5.3302	-1.41%

(a)

$ P $	$ T $	$\chi(t_{\min})$	$\chi^{ub}(t_{\min})$	Steps	$ P' $	$ T' $	$\Theta$	%
301	463	0.7884	0.7113	2	64	98	0.6897	-3.13%
305	456	1.9988	1.4475	5	92	140	1.9301	25.00%
283	441	1.5148	2.2792	3	58	91	1.4933	-52.63%
283	453	1.3327	1.5662	3	64	99	1.3537	-15.70%
298	457	2.4061	2.3392	1	31	49	2.3648	1.08%
285	430	2.3138	3.6980	2	59	91	2.2626	-63.44%

(b)

that the value computed by  $\mathcal{P}_1$  is in fact an upper throughput bound, while the value returned by the Algorithm 1 is approximated, i.e., this value is higher or lower than the real throughput. As future work, we aim to characterize it better with regard to the real throughput.

Note also that in a few cases, the iterative heuristic returns throughput values, which are in fact greater than the initial upper throughput bound. Besides, the real throughput value is also greater than the upper bound in these cases. These situations are caused by simulation errors.

The difference in size between the subnet returned by the  $P$ -semiflow obtained by Algorithm 1 and the original net is nearly 20% in both experiments for big  $S^4PR$  nets, while it reaches a values of almost 40% and 60% for small  $S^4PR$  nets having big and small processes, respectively. Hence, a small portion of the original net is representative enough of the real throughput of the transition. Note also that Algorithm 1 rapidly converges, using a low number of iterations.

Let us finally remark that Algorithm 1 chooses as a reference transition the one with the lowest initial upper throughput bound. However, this transition can be carefully chosen in advance by an engineer considering the system under study.

## VI. CONCLUSION AND FUTURE WORK

In this brief, we have proposed an iterative strategy to approximate throughput values in  $S^4PR$  nets. The strategy

is based on first computing the slowest  $P$ -semiflow, which is then iteratively regrown considering the next  $P$ -semiflow most likely to be constraining the system. Although we have shown that our iterative approximation technique converges, we cannot characterize its convergence speed nor its accuracy. Thus, we have evaluated our strategy in a set of randomly generated  $S^4PR$  nets to give an insight into its usefulness.

From the extensive experiments performed, we concluded that: 1) the throughput obtained with our approach approximates better to the real throughput than (classical) upper throughput bounds, reaching an improvement on average close to 20% and 2) small portions of the net are representative enough to approximate to the real throughput of a transition in big  $S^4PR$  nets. In particular, the experiments showed that roughly 20% of the original net size is enough. Roughly speaking, our results indicate that the bigger the  $S^4PR$  net is, the smaller the net proportion which is sufficiently representative.

As future work, we aim at extending the approach to nets with a more general structure (i.e., general nets) or to interval time PNs. Similarly, we aim at studying other properties of RAS, such as resource optimization, as well as alternative techniques based on response time approximations and matrix compact representations of the infinitesimal generator matrix.

#### REFERENCES

- [1] J. M. Colom, "The resource allocation problem in flexible manufacturing systems," in *Applications and Theory of Petri Nets* (Lecture Notes in Computer Science), vol. 2679. Berlin, Germany: Springer, 2003, pp. 23–35.
- [2] J. López-Grao, J. M. Colom, and F. Tricas, "Structural deadlock prevention policies for flexible manufacturing systems: A Petri net outlook," in *Formal Methods in Manufacturing* (Industrial Information Technology), J. Campos, C. Seatzu, and X. Xie, Eds. Boca Raton, FL, USA: CRC Press, Mar. 2014, ch. 7, pp. 197–228.
- [3] J. Ezpeleta and L. Recalde, "A deadlock avoidance approach for nonsequential resource allocation systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 1, pp. 93–101, Jan. 2004.
- [4] F. Tricas and J. Ezpeleta, "Computing minimal siphons in Petri net models of resource allocation systems: A parallel solution," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 3, pp. 532–539, May 2006.
- [5] J. Ezpeleta and R. Valk, "A polynomial deadlock avoidance method for a class of nonsequential resource allocation systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 6, pp. 1234–1243, Nov. 2006.
- [6] J. Campos, C. Seatzu, and X. Xie, Eds., *Formal Methods in Manufacturing*. Boca Raton, FL, USA: CRC Press, 2014.
- [7] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling With Generalized Stochastic Petri Nets* (Wiley Series in Parallel Computing). Hoboken, NJ, USA: Wiley, 1995.
- [8] M. Jeng, X. Xie, and W. Hung, "Markovian timed Petri nets for performance analysis of semiconductor manufacturing systems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 5, pp. 757–771, Oct. 2000.
- [9] R. J. Rodríguez, J. Júlvez, and J. Merseguer, "On the performance estimation and resource optimization in process Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 6, pp. 1385–1398, Nov. 2013.
- [10] J. Campos and M. Silva, "Structural techniques and performance bounds of stochastic Petri net models," in *Advances in Petri Nets* (Lecture Notes in Computer Science), vol. 609, G. Rozenberg, Ed. Berlin, Germany: Springer, 1992, pp. 352–391.
- [11] S. Bernardi and J. Campos, "A min-max problem for the computation of the cycle time lower bound in interval-based time Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1167–1181, Sep. 2013.
- [12] R. J. Rodríguez and J. Júlvez, "Accurate performance estimation for stochastic marked graphs by bottleneck regrowing," in *Proc. 7th Eur. Perform. Eng. Workshop*, vol. 6342. Sep. 2010, pp. 175–190.
- [13] M. Sereno and G. Balbo, "Mean value analysis of stochastic Petri nets," *Perform. Eval.*, vol. 29, no. 1, pp. 35–62, 1997.
- [14] S. Haddad, P. Moreaux, M. Sereno, and M. Silva, "Product-form and stochastic Petri nets: A structural approach," *Perform. Eval.*, vol. 59, pp. 313–336, Mar. 2005.
- [15] J. Campos, S. Donatelli, and M. Silva, "Structured solution of asynchronously communicating stochastic modules," *IEEE Trans. Softw. Eng.*, vol. 25, no. 2, pp. 147–165, Mar. 1999.
- [16] M. Sereno, "Approximate mean value analysis for stochastic marked graphs," *IEEE Trans. Softw. Eng.*, vol. 22, no. 9, pp. 654–664, Sep. 1996.
- [17] C. J. Perez-Jimenez and J. Campos, "On state space decomposition for the numerical analysis of stochastic Petri nets," in *Proc. 8th Int. Workshop Petri Nets Perform. Models*, 1999, pp. 32–41.
- [18] A. S. Miner, G. Ciardo, and S. Donatelli, "Using the exact state space of a Markov model to compute approximate stationary measures," in *Proc. ACM SIGMETRICS*, 2000, pp. 207–216.
- [19] P. Buchholz, "Adaptive decomposition and approximation for the analysis of stochastic Petri nets," *Perform. Eval.*, vol. 56, nos. 1–4, pp. 23–52, 2004.
- [20] H. Jungnitz and A. A. Desrochers, "Flow equivalent nets for the performance analysis of generalized stochastic Petri nets," in *Proc. IEEE Robot. Autom. Conf.*, Apr. 1991, pp. 122–127.
- [21] Y. Li and C. Woodside, "Complete decomposition of stochastic Petri nets representing generalized service networks," *IEEE Trans. Comput.*, vol. 44, no. 4, pp. 577–592, Apr. 1995.
- [22] J. Freiheit and A. Zimmermann, "A divide and conquer approach for the performance evaluation of large stochastic Petri nets," in *Proc. 9th Int. Workshop Petri Nets Perform. Models*, 2001, pp. 91–100.
- [23] C. Pérez-Jiménez, J. Campos, and M. Silva, "Approximate throughput computation of stochastic weighted T-systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 3, pp. 431–444, May 2007.
- [24] D. Lefebvre, E. Leclercq, N. El Akchioui, E. S. de Cursis, and L. Khalij, "A geometric approach for the homothetic approximation of stochastic Petri nets," in *Proc. 10th Int. Workshop Discrete Event Syst. (WODES)*, 2010, pp. 235–240.
- [25] C. R. Vazquez and M. Silva, "Stochastic continuous Petri nets: An approximation of Markovian net models," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 3, pp. 641–653, May 2012.
- [26] H. Hu, M. Zhou, and Z. Li, "Low-cost and high-performance supervision in ratio-enforced automated manufacturing systems using timed Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 4, pp. 933–944, Oct. 2010.
- [27] H. Hu and Y. Liu, "Supervisor synthesis and performance improvement for automated manufacturing systems by using Petri nets," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 450–458, Apr. 2015.
- [28] G. Chiola, C. Anglano, J. Campos, J. M. Colom, and M. Silva, "Operational analysis of timed Petri nets and application to the computation of performance bounds," in *Proc. IEEE 5th Int. Workshop Petri Nets Perform. Models*, Oct. 1993, pp. 128–137.
- [29] Z. Liu, "Performance analysis of stochastic timed Petri nets using linear programming approach," *IEEE Trans. Softw. Eng.*, vol. 24, no. 11, pp. 1014–1030, Nov. 1998.
- [30] S. Bernardi and J. Campos, "Computation of performance bounds for real-time systems using time Petri nets," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 168–180, May 2009.
- [31] D. L. Eager and K. C. Sevcik, "Bound hierarchies for multiple-class queuing networks," *J. ACM*, vol. 33, no. 1, pp. 179–206, Jan. 1986.
- [32] M. M. Srinivasan, "Successively improving bounds on performance measures for single class product form queueing networks," *IEEE Trans. Comput.*, vol. C-36, no. 9, pp. 1107–1112, Sep. 1987.
- [33] R. Suri, "Generalized quick bounds for performance of queueing networks," *Comput. Perform.*, vol. 5, no. 2, pp. 116–120, Jun. 1984.
- [34] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
- [35] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. New York, NY, USA: Springer-Verlag, 2001.
- [36] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Ph.D. dissertation, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, Feb. 1974.
- [37] R. J. Rodríguez, "A Petri net tool for software performance estimation based on upper throughput bounds," *Automat. Softw. Eng.*, vol. 24, no. 1, pp. 73–99, Jan. 2017.
- [38] S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN tool: Recent enhancements," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 4–9, 2009.