On approximate performance evaluation of manufacturing systems modelled with weighted T–systems

Carlos J. Pérez-Jiménez, Javier Campos, and Manuel Silva
Departamento de Informática e Ingeniería de Sistemas *
Centro Politécnico Superior, Universidad de Zaragoza
María de Luna 3, E-50015 Zaragoza, Spain
e-mail: cjperez, fjcampos, msilva @mcps.unizar.es.

ABSTRACT

Approximate throughput computation of a class of discrete event systems (DES) modelled with stochastic weighted T–systems is considered. Stochastic weighted T–systems are the weighted extension of well known stochastic marked graphs Petri net subclass and are usually presented as a useful model to deal with bulk consumptions or productions of resources in manufacturing systems working on a cyclic basis. The iterative response time approximation technique that we present is deeply based on a previous structural decomposition and aggregation of the net model. Experimental results on several examples generally have and error of less than 5%. The state space is usually reduced by more than one order of magnitude; therefore, the analysis of otherwise intractable systems is possible.

1 INTRODUCTION

Petri nets have been shown to be an adequate formalism for the modelling, functional analysis and performance evaluation of *discrete event and manufacturing systems* [11, 4]. In this paper we concentrate in steady-state performance analysis of systems with a cyclic behaviour modelled with a subclass of *stochastic Petri nets*, assuming exponentially distributed service times for transitions [5]. In particular, we study the productivity measure defined in terms of the *throughput* of transitions of the net model. The general approach for the computation of throughput of transitions of bounded stochastic Petri nets (with exponential timing) is based on the solution of the underlying continuous time Markov chain and suffers from the well-known *state explosion* problem. In order to deal with this problem, we propose an approximation technique based on the solution of several subsystems with smaller state spaces, practically reducing the execution time of the solution technique in one or even more orders of magnitude. The presented technique is strongly based in a structural decomposition and aggregation of the net model, and we are not able yet to perform it for a general Petri net model. Therefore, we restrict ourselves to the particular net subclass of *stochastic weighted T–systems* (SWTS's, in the sequel) [12]. SWTS's are the stochastic interpretation of the weighted extension of classical *marked graphs* and are usually presented as a useful model to deal with *bulk* consumptions or productions of resources in *assembly systems* working on a cyclic basis [7].

In a previous paper [2], we considered the particular case of stochastic marked graphs (thus, *ordinary* models, i.e., models without bulk movements). For those models we showed how a structural decomposition into two *aggregated subsystems*, that leads to an exact aggregation from a functional point of view[1], allows to get a very accurate throughput approximation by means of a response time approximation algorithm that iteratively solves both subsystems and a basic skeleton.

In this paper we extend the technique to the non-ordinary case and we also consider a more general decomposition into several (more than two) subsystems, thus allowing the solution of larger models. The extension is far from being straightforward. We use the concept of *gain* between two transitions and we elaborate on the concept of *resistance* also between two transitions —introduced in [9]— that help to derive the aggregated subsystems preserving in a reasonable way the functional behaviour of the original system.

Other authors have considered performance analysis of weighted T–systems in the case of deterministic timing (see for instance [7]) but, to the best of our knowledge, nobody dealt with the stochastic case.

The paper is organized as follows. In Section 2 the main notation and definitions are given. After that, we define the concepts of gain and resistance between transitions and give an intuitive idea of their meaning. We also present a dynamic programming algorithm for their computation. Section 3 includes the throughput approximation technique in two steps: (1) an structural decomposition of the model and construction of the aggregated subsystems and basic skeleton, making use of the concepts introduced in Section 2, and (2) a brief explanation of the iterative response time approximation method. In Section 4 we present an example and some numerical results to illustrate the technique. Finally, some concluding remarks are summarized in Section 5.

*This work has been developed within the projects PRONTIC 94-0242 of the Spanish CICYT and HCM CT94-0452 (MATCH) of the European Community

[1] In this context, exact aggregation means that the state space and firing sequences of the whole system can be reconstructed from those of the aggregated subsystems.

2 AGGREGATION OF SUBNETS IN WEIGHTED T–SYSTEMS

Petri nets notations and definitions
We assume the reader is familiar with concepts of P/T nets. Here we only present notations used later. For further extensions the reader is referred to [8, 4].
A P/T *net* is a 4–tuple $\mathcal{N} = (P, T, Pre, Post)$, where P and T are disjoint sets of *places* and *transitions* ($|P| = n$, $|T| = m$), and Pre ($Post$) is the *pre- (post-) incidence function* representing the input (output) arcs: $Pre: P \times T \to \mathbb{N} = \{0, 1, 2, \ldots\}$ ($Post: P \times T \to \mathbb{N}$). *Ordinary* nets are Petri nets whose pre- and post-incidence functions take values in $\{0, 1\}$. The incidence function of a given arc in a non-ordinary net is called *weight* or *multiplicity*. The *pre-* and *post-set* of a transition $t \in T$ are defined respectively as ${}^\bullet t = \{p | Pre(p, t) > 0\}$ and $t^\bullet = \{p | Post(p, t) > 0\}$. The *pre-* and *post-set* of a place $p \in P$ are defined respectively as ${}^\bullet p = \{t | Post(p, t) > 0\}$ and $p^\bullet = \{t | Pre(p, t) > 0\}$. The *incidence matrix* of the net is defined as $C = [Post(p_i, t_j) - Pre(p_i, t_j)]$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. *Flows* (*semiflows*) are integer (natural) annihilators of C. Right and left annihilators are called T- and P-(semi)flows respectively. A semiflow is called *minimal* when its support, $\|X\|$, is not a proper superset of the support of any other, and the greatest common divisor of its elements is one. A net is *consistent* if it has a T-semiflow $X \geq \mathbb{1}$. A net is *conservative* if it has a P-semiflow $Y \geq \mathbb{1}$.
A function $M: P \to \mathbb{N}$ (usually represented in vector form) is called *marking*. A P/T *system*, or *marked Petri net*, $(\mathcal{N}, M_0)$, is a P/T net $\mathcal{N}$ with an *initial marking* M_0. A transition $t \in T$ is *enabled* at marking M if $\forall p \in P$: $M(p) \geq Pre(p, t)$. A transition t enabled at M can *fire* yielding a new marking M' (*reached* marking) defined by $M'(p) = M(p) - Pre(p, t) + Post(p, t)$ (it is denoted by $M[t\rangle M'$). A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is a *firing sequence* in $(\mathcal{N}, M_0)$ if there exists a sequence of markings such that $M_0[t_1\rangle M_1[t_2\rangle M_2 \ldots [t_n\rangle M_n$. In this case, marking M_n is said to be *reachable* from M_0 by firing σ, and this is denoted by $M_0[\sigma\rangle M_n$. The *reachability set* $R(\mathcal{N}, M_0)$ is the set of all markings reachable from the initial marking. The function $\vec{\sigma}: T \to \mathbb{N}$ is the *firing count vector* of the firable sequence σ, i.e., $\vec{\sigma}[t]$ represents the number of occurrences of $t \in T$ in σ. If $M_0[\sigma\rangle M$, then we can write in vector form $M = M_0 + C \cdot \vec{\sigma}$, which is referred to as the *linear state equation* of the net.
A place $p \in P$ is said to be k*–bounded* if $\forall M \in R(\mathcal{N}, M_0)$, $M(p) \leq k$. A P/T system is said to be (marking) k–bounded if every place is k–bounded, and bounded if there exists some k for which it is k–bounded. A P/T system is *live* when every transition can ultimately occur from every reachable marking. An *implicit place* never is the unique restricting the firing of its output transitions. M is a *home state* in $(\mathcal{N}, M_0)$ if it is reachable from every reachable marking.
Now, we recall the definition of the net subclass that is considered in the sequel. *Weighted T–systems* (WTS's) are the weighted generalization of *marked graphs* (see an example in Fig. 4.a). Even if some results for WTS are essentially parallel to those for the ordinary (non-weighted) case [12], there are interesting differences that play an important role in the decomposition of WTS models.

Definition 2.1 (Stochastic weighted T–system) *A P/T system $(\mathcal{N}, M_0) = (P, T, Pre, Post, M_0)$ is a weighted T–system (WTS) if $\forall p \in P : |{}^\bullet p| = |p^\bullet| = 1$. A stochastic WTS (SWTS) is obtained from $(\mathcal{N}, M_0)$ if independent and exponentially distributed random variables are associated to the firing time of transitions.*

From a queueing network perspective, SWTS's are a mild generalization of *Fork-Join Queuing Networks with Blocking* where bulk movements of jobs are allowed.
In this paper, a general decomposition technique is developed for this class of nets. The following properties will be preserved after the reduction: boundedness (essential to the finiteness of the underlying CTMC of the stochastically timed models), liveness (for strongly connected WTS's equivalent to deadlock-freeness [12], thus non-null throughput preservation) and the existence of home states (to preserve the CTMC ergodicity). To improve the accuracy of the numerical results we will define three additional functional properties related to paths (gain, resistance and weighted marking). We will also preserve the gain, maximum resistance and minimum weighted marking among *interface transitions*. The decomposition phase will be performed in *polynomial time* on the net size.

Gain, resistance and weighted marking of paths
The reduction technique that we propose is a generalization of that presented in [2] for marked graphs. A multiple cut is defined through places whose deletion generates a partition of the system into several unconnected subsystems (like in [10]).
Once a multiple cut has been decided, the main problem is to derive the *aggregated subsystems* where we need to reduce some subnets of the original system to a "minimum number" of nodes preserving some interesting functional properties (the *basic skeleton* will be obtained after reduction of all the parts). In Fig. 4.a, the lower part (the subnet formed by all the nodes below the eight places of the cut) is replaced with a set of nodes such that we can preserve the functional properties previously mentioned (see Fig. 4.b).
The first functional property that we must preserve is the *gain* of any path joining interface transitions. The gain of a path represents the average number of firings of the final transition per each single firing of the first transition. This functional property depends only on the weights of the arcs of the path and we must preserve it to maintain the consistency and conservativeness of the reduced subsystems. The preservation of the gain of a path can be done substituting it by the implicit place joining the first and last transitions (place $p12$ in Fig. 1.).

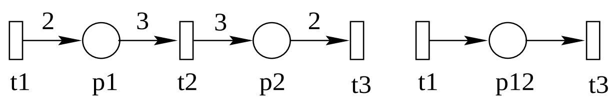

Figure 1: A path and its reduction preserving the gain

But the preservation of the gain of a path is not enough to achieve good numerical results after the decomposition of a general SWTS. The problem is that inside of the weighted arcs of a path there are implicit synchronizations. For example, in Fig. 2.a, we have a path of gain equal to one (the average number of firings of the first transition is the same that the last one), but the weights of the arcs of the path induce a synchronization at transition $t2$. We need to fire ten times the first transition for the last transition to be fired. In general, if we reduce the path as in Fig. 2.b (with its implicit place preserving only the gain of the path), we obtain poor numerical results.

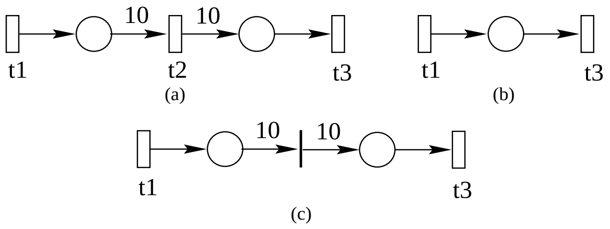

Figure 2: (a) A path, (b) its reduction preserving the gain and (c) its reduction preserving the gain and the resistance.

So, we need to introduce another concept, that we call *resistance* of a path. The resistance of a path is a property related with the average number of firings of the first transition that are needed to fire the last one. In the case of Fig. 2 the resistance of the path is 10. If we consider for a given path all the subpaths from the first transition to the rest, it is clear that there is a transition such that the resistance of the corresponding path is maximum (it is $t2$ in the case of Fig. 2). Then, we can preserve the gain and resistance of a path with an immediate transition representing the transition of maximum resistance of the path and the two implicit places summarizing the gain of the subpaths in which this transition divides the original path (see Fig. 2.c). It is obvious that depending on the weights of the arcs of a path the implicit place preserving the gain can also summarize the resistance. So, this reduced paths are implicit with respect to the original path.

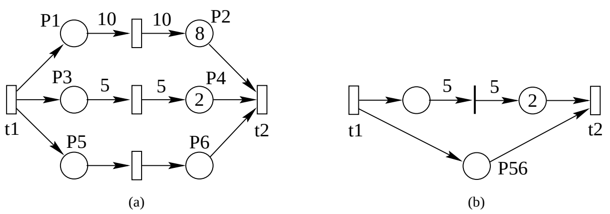

Figure 3: Reduction of several paths between two transitions.

In a general SWTS we may have several paths joining two given transitions. For example, in Fig. 3.a., there are three paths between t_1 ad t_2. If the SWTS is live and bounded all the paths joining the two transitions must have the same gain [12]. But they can have different resistance. If we look only at the weights of the arcs, the first path has resistance 10, the second one 5 and the last one 1. But the resistance is related with the average number of firings of the first transition needed to fire the last one. The 8 tokens of the place $P2$ make smaller the resistance of the first path. Moreover, the 8 tokens never can leave the path, so they are always reducing its resistance. In other words, the concept of resistance depends also on the initial marking of the path. In this case, the first path has resistance 2 (the weight 10 minus the 8 tokens), the second one 3 and the last one 1. Thus, the second path is the path who offers the maximum resistance between the first and the last transition. And the third path is the path which captures the tokens in the other paths. This is the path with minimum *weighted marking*.

The weighted marking of a path is a property related with the number of times that the first transition of the path must be fired to achieve the current marking of the path (the weighted markings of the paths of Fig. 3.a are 8, 2 and 0 respectively). The implicit place corresponding to a given path not only preserves the gain of the path but also preserves its weighted marking.

Thus, the reduction of several paths joining two transitions is as follows. We compute the path with *maximum resistance* and the path with *minimum weighted marking*. The path with maximum resistance is reduced as we explained before and the path with minimum weighted marking is substituted by its implicit place (see Fig. 3.b). Note that, depending on the weights of the paths and the markings, one of the reduced paths can be implicit with respect to the other and in this case we can delete it (that is not the case for the reduction in Fig. 3.b).

Now, we are interested in computing for every pair of transitions in a SWTS, the path joining them with the maximum resistance and the path with minimum weighted marking. To do that we are going to translate the nets language to the graphs language. A SWTS can be represented as a directed graph whose vertices are the transitions of the SWTS and the edges are the places joining them. In order to include the properties of gain, resistance and weighted marking in the graph system, we put a label on each edge of the graph with three components. The first one is the quotient of the weight of the output arc of the corresponding place by the weight of its input arc (this component is the resistance of this edge), the second one is the quotient of the marking of the place by the weight of its input arc (weighted marking) and the third one is the inverse of the first one (is the gain of this edge). We can express for any path in the graph its corresponding label recursively from the labels of the edges of the path. The label of a path represents its resistance, weighted marking and gain. To do that we need to define the appropriate operations.

Definition 2.2 *Let* $S = \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \setminus \{0\} \cup \overline{0}$ *where* $\overline{0} = (\infty, \mathcal{X}_1, 0)^2$ *, and let* $\overline{1} = (0, 0, 1)$. *We define the following operations on S:*
i) $(r_1, w_1, g_1) \odot (r_2, w_2, g_2) = (max\{r_1, r_2/g_1\}, w_1 + w_2/g_1, g_1 g_2)$ $\forall (r_i, w_i, g_i) \in S \setminus \overline{0}$ *and* $\overline{0} \odot s = s \odot \overline{0} = \overline{0} \odot \overline{0} = \overline{0}$ $\forall s \in S$.
ii) $(r_1, w_1, g) \oplus (r_2, w_2, g) = (max\{r_1 - w_1, r_2 - w_2\} + min\{w_1, w_2\}, min\{w_1, w_2\}, g)$ *if* $r_1 - w_1 \neq r_2 - w_2$ *and* $(r_1, w_1, g) \oplus (r_2, w_2, g) = (min\{r_1, r_2\}, min\{w_1, w_2\}, g)$ *if* $r_1 - w_1 = r_2 - w_2$.

The set S is the domain of possible labels, the $\odot$ operation computes the label of a path from the label of subpaths and the $\oplus$ operation selects between two paths joining the same two transitions the path with maximum resistance.

[2] $\mathcal{X}_1$ is the infinite cardinal number in the usual mathematical sense. $\infty - \mathcal{X}_1 = -\mathcal{X}_1$.

If two paths have the same resistance, $\oplus$ selects the one with minimum weighted marking.
In this way we have done a formal *recursive* definition of the gain, resistance and weighted marking of a path. Operations $\oplus$ and $\odot$ give the recursive equations to express them from the initial gain, resistance and weighted marking of the edges.
The next theorem gives us the proof of existence of a dynamic programming algorithm to compute in polynomial time for any two vertices in a graph the path with maximum resistance joining them (see [3, pp. 570–575] for the details).

Theorem 2.1 *The system* $(S, \oplus, \odot, \overline{0}, \overline{1})$ *is a closed semiring satisfying the following eight properties:*
i) $(S, \oplus, \overline{0})$ *is a monoid in the definition domain of* $\oplus$ *(i.e.* $\oplus$ *satisfy the associative property and* $\overline{0}$ *is an identity for* $\oplus$*).*
ii) $(S, \odot, \overline{1})$ *is a monoid.*
iii) $a \odot \overline{0} = \overline{0} \odot a = \overline{0}\ \forall a \in S$.
iv) $a \oplus b = b \oplus a\ \forall a, b \in S$.
v) $a \oplus a = a\ \forall a \in S$.
vi) $\odot$ *distributes over* $\oplus$.
vii) $\oplus$ *is well defined for countable sequences of a finite number of elements of* S *and associativity, commutativity and idempotence apply to infinite summaries. (Thus, such an infinite summary can be rewritten as an finite summary in which each term of the summary is included just once and the order of evaluation is arbitrary.)*
viii) $\odot$ *distributes over infinite summaries of a finite number of elements of* S.

The proof of this theorem is a tedious algebraic exercise. With this algebraic environment, following the general framework for solving path problems in directed graphs [3], we can assure that there exists a dynamic programming algorithm to compute for any pair of vertices in a graph the path with maximum resistance joining them. This algorithm is no more than a generalization of the well-known Floyd's *all pairs shortest path* algorithm.
The algorithm is the following:

n :=number of vertices
for $i := 1$ **to** n **do**
  for $j := 1$ **to** n **do**
    if edge (i, j) in the graph $l_{ij}^{(0)} := \lambda(i, j)$
    else $l_{ij}^{(0)} := \overline{0}$
for $k := 1$ **to** n **do**
  for $i := 1$ **to** n **do**
    for $j := 1$ **to** n **do**
      $l_{ij}^{(k)} := l_{ij}^{(k-1)} \oplus \left(l_{ik}^{(k-1)} \odot l_{kj}^{(k-1)} \right)$
return $L^{(n)}$

The time complexity of this algorithm is $O(n^3(T_{\oplus} + T_{\odot})) = O(n^3)$ (because the computing time of $\oplus$ and $\odot$ is constant). With an additional square matrix we can keep track of the vertices of the optimal paths during the execution of the algorithm. In this way we can obtain the real optimal paths.
Now we have all the structural techniques needed to decompose a SWTS in several subnets.

3 APPLICATION TO THROUGHPUT APPROXIMATION

In this section, we present the method for the approximate throughput computation of the transitions of a general SWTS.
The first subsection deals with the system decomposition. A multiple cut defined through places splits the system in several subsystems. Each of them is reduced preserving the gain, maximum resistance and minimum weighted marking among all the cut interface transitions. Also a basic skeleton is defined by aggregating all the subnets.
The second subsection shows the iterative algorithm for the throughput approximation of the transitions (essentially the same used for marked graphs [2], called *pelota algorithm*).

The decomposition phase
The decomposition of a live and bounded SWTS (like that in Fig. 4.a) is based on the splitting in k pieces by a cut defined on places. Once the multiple cut and the pieces are selected we construct k *aggregated subsystems* ($\mathcal{AS}_1, \ldots, \mathcal{AS}_k$; see Figs. 4.b and 4.c) and a *basic skeleton system* ($\mathcal{BS}$; see Fig. 4.d). First, we formally define the cut.

Definition 3.1 *Let* $(\mathcal{N}, M_0) = (P, T, Pre, Post, M_0)$ *be a live and bounded WTS. A subset* $Q \subseteq P$ *of places is said a* k*-cut of* $\mathcal{N}$ $(k \geq 2)$ *if there exist* k *subnets* $\mathcal{N}_i = (P_{\mathcal{N}_i}, T_{\mathcal{N}_i}, Pre_{\mathcal{N}_i}, Post_{\mathcal{N}_i})$, $i = 1, \ldots, k$, *of* $\mathcal{N}$ *verifying:*
i) $\bigcup_{i=1}^{k} T_{\mathcal{N}_i} = T$, $T_{\mathcal{N}_i} \cap T_{\mathcal{N}_j} = \emptyset$, $\forall i \neq j$, $i, j = 1, \ldots, k$.
ii) $P_{\mathcal{N}_i} = {}^{\bullet}T_{\mathcal{N}_i} \cup T_{\mathcal{N}_i}{}^{\bullet}$, $i = 1, \ldots, k$.
iii) $\bigcup_{i=1}^{k} P_{\mathcal{N}_i} = P$ *and* $\bigcup_{i \neq j} \left(P_{\mathcal{N}_i} \cap P_{\mathcal{N}_j}\right) = Q$, $i, j \in \{1, \ldots, k\}$.
iv) $Pre_{\mathcal{N}_i} = Pre|_{P_{\mathcal{N}_i} \times T_{\mathcal{N}_i}}, Post_{\mathcal{N}_i} = Post|_{P_{\mathcal{N}_i} \times T_{\mathcal{N}_i}}, i \in \{1, \ldots, k\}$.
The transitions $t \in {}^{\bullet}Q \cup Q^{\bullet}$ *are called interface transitions.*

Now we have to build the aggregated subsystems and the basic skeleton obtained from a k-cut. To do that, we must aggregate each subnet to a set of nodes. We take each subnet $\mathcal{N}_i$ and we apply to it the dynamic programming algorithm to compute for any pair of its interface transitions the paths with maximum resistance and minimum weighted marking joining them. As we looked in section 2, the path with mimimum weighted marking can be reduced to its implicit place, and the path with maximum resistance to an immediate transition and two places. We need all this reduced paths to construct the aggregate subsystems.

Definition 3.2 *Let* $(\mathcal{N}, M_0)$ *a live and bounded WTS,* $Q \subseteq P$ *a* k*-cut of* $\mathcal{N}$ *and* $\mathcal{N}_i, i = 1, \ldots, k$ *the* k *subnets defined by* Q. *We call aggregated subnet* $\mathcal{ASN}_i$ *to the net system formed by the interface transitions of* $\mathcal{N}_i$ *(we will denote it by* $IT_i = T_{\mathcal{N}_i} \cap {}^{\bullet}Q \cup Q^{\bullet}$*) and the set of reduced paths to preserve the gain, maximum resistance and minimum weighted marking among the interface transitions in* $\mathcal{N}_i$.

Now we can formally define the aggregated subsystems and the basic skeleton.

Definition 3.3 *Let $(\mathcal{N}, M_0)$ be a live and bounded WTS, $Q \subseteq P$ a k-cut of $\mathcal{N}$ and $\mathcal{N}_i, i = 1, \ldots, k$, the k subnets defined by Q. For each $i = 1, \ldots, k$, the aggregated subsystem $\mathcal{AS}_i$ is the system obtained from $(\mathcal{N}, M_0)$ by substituting $\mathcal{N}_j$ by $\mathcal{ASN}_j$, $\forall j \neq i$. The basic skeleton $\mathcal{BS} = (\mathcal{BN}, M_0^{\mathcal{BN}})$ is the system obtained from $(\mathcal{N}, M_0)$ by substituting $\mathcal{N}_i$ by $\mathcal{ASN}_i$, $\forall i = 1, \ldots, k$.*

After this reduction, the projection of the state space of the original system on the non-reduced part is not preserved in the aggregated subsystems (this fact differs from the marked graph case [2]). In other words, spurious markings can be made reachable after the aggregation. But we can preserve other interesting functional properties (and less strong) and the numerical results obtained are good enough.
The next theorem includes the main result of this section. It relates the behaviour of a WTS with those of the aggregated subsystems and the basic skeleton.

Theorem 3.1 *Let $(\mathcal{N}, M_0)$ be a live and bounded WTS, $Q \subseteq P$ a k-cut of $\mathcal{N}$, $\mathcal{AS}_i, i = 1, \ldots, k$ and $\mathcal{BS}$ the aggregated subsystems and the basic skeleton derived from Q. Then, $\mathcal{AS}_i, i = 1, \ldots, k$ and $\mathcal{BS}$ are live, bounded and have home state.*

Proof: In [12] it is proved that a necessary and sufficient condition for a WTS to be live and bounded is that any weighted cycle has gain equal to one and tokens enough to making it live. To construct the aggregated subsystems and the basic skeleton we substitute several paths by a set of nodes preserving the gain, so all the cycles in the aggregated subsystems and the basic skeleton have gain equal to one. With regard to the marking, all the places that we put in these systems are implicit in $(\mathcal{N}, M_0)$ so the reduced cycles in the aggregated subsystems and the basic skeleton are also live. As any live and bounded WTS has home state, the aggregated subsystems and the basic skeleton have also home state. Q.E.D.

Iterative approximation phase
The technique for an approximate computation of the throughput that we present now is, basically, a *response time approximation* method [2]. The interface transitions of $\mathcal{N}_j$ in $\mathcal{AS}_i$ $(j \neq i)$ approximate the response time of all the subsystem $\mathcal{N}_j$. A direct (non-iterative) method to compute the constant service rates of observable transitions in order to represent the aggregation of the subnet gives, in general, low accuracy. Therefore, we are forced to define a *fixed-point search iterative process*. The proposed algorithm is the following:

select a k-cut Q
derive $\mathcal{AS}_i$, $i = 1, \ldots, k$, and $\mathcal{BS}$
give an initial service rate $\mu_t^{(0)}$ for $t \in IT_i, i = 2, \ldots, k$
$j := 0$ {counter for iteration steps}
repeat
  $j := j + 1$
  for $i := 1, \ldots, k$
    solve the aggregated subsystem $\mathcal{AS}_i$ with
      input: $\mu_t^{(j)}$ for each $t \in IT_l$ $(l < i)$
          $\mu_t^{(j-1)}$ for each $t \in IT_l$ $(i < l \leq k)$
      output: ratios among $\mu_t^{(j)}$ for each $t \in IT_i$, and $\mathcal{X}_i^j$
    solve the basic skeleton system $\mathcal{BS}$ with
      input: $\mu_t^{(j)}$ for each $t \in IT_l$ $(l < i)$
          $\mu_t^{(j-1)}$ for each $t \in IT_l$ $(i < l \leq k)$
          ratios among $\mu_t^{(j)}$ of $t \in IT_i$ and $\mathcal{X}_i^j$
      output: scale factor of $\mu_t^{(j)}$ of $t \in IT_i$
  end for
until convergence of $\mathcal{X}_1^{(j)}, \ldots, \mathcal{X}_k^{(j)}$

In the above procedure, once a k-cut has been selected and given some initial values $\mu_t^{(0)}$ for service rates of all interface transitions except those in $\mathcal{N}_1$, the underlying CTMC of aggregated subsystem $\mathcal{AS}_1$ is solved. The selection of the initial values of interface transitions rates does not affect (under our experience) to the accuracy of the method. A simple option is putting the initial rate of the transitions in the original model. From the solution of that CTMC, the first estimation $\mathcal{X}_1^{(1)}$ of the throughput of $\mathcal{AS}_1$ can be computed. Then, the initial estimated values of service rates of interface transitions IT_1 must be derived. To do that, we take the initial values $\mu_t^{(0)}$ for service rates of transitions in IT_1 and we search in the basic skeleton a *scale factor* for all these rates such that the throughput of the basic skeleton and the throughput of $\mathcal{AS}_1$, computed before, are equal. The same procedure is executed for each aggregated subsystem in a cyclic way. Each time we solve the aggregated subsystem $\mathcal{AS}_i$ we obtain in the basic skeleton a new estimation of the interface transitions rates of IT_i. The previous steps are repeated until *convergence* is achieved (if there exists).
With regard to the computation of the ratios among the interface transitions rates in the aggregated subsystems, we compute in the solution of the $\mathcal{AS}_i$, the probability p_t that the transition $t \in IT_i$ is enabled (similar to [2]). Then we put $\mu_t^{(j)} = \mathcal{X}_t / p_t$.
The computation of the scale factor in the basic skeleton can be implemented with a linear search. Now the net system (the basic skeleton) has considerably fewer states than the original one. In each iteration of this linear search, the underlying CTMC of the basic skeleton is solved. Note that only in the first iteration the CTMC is completely derived. For later iterations only some values must be changed. From [1] follows that in stochastic SWTS's (with time and marking independent rates) the throughput of any transition is a monotonic function of the rate of any other transition. In other words, if we increase (decrease) the rate of any transition, the throughput of any other transition will not be decreased (increased). This fact assures that the linear search in the basic skeleton (being a SWTS) converges.
We conjecture that the convergence of the entire method can be proved (probably using similar arguments to those in [6]). With regard to the accuracy of the results, no formal proof gives positive answers to the question, but an extensive battery of numerical experiments has shown us that the error is less than 5% in all tested cases.

4 NUMERICAL RESULTS

The class of SWTS's is naturally applied for describing assembly systems. The operations performed by different machines can be modelled with transitions, while places represents intermediate stores. Assembly operations are

performed by machines (transitions) having more than one input place. An assembly operation puts parts together to obtain more complex ones. The weights on the input (resp., output) arcs of places represent the number of parts of each type required to obtain one unit of the next type (resp., the number of complex parts that are obtained after the operation).
We have choosen the example depicted in Fig. 4.a because any splitting of the net will generate strongly coupled aggregated subsystems. This fact makes the system difficult to study and puts our method through a test.
Let us present some numerical values obtained for the system modelled in Fig. 4.a. We select the following cut: $Q = \{P21, P22, P23, P24, P25, P26, P27, P28\}$. The corresponding aggregate subsystems are depicted in Figs. 4.b and 4.c. The basic skeleton is that in Fig. 4.d. The underlying CTMC of the original SWTS has 90171 states, while the aggregated subsystems and the basic skeleton have 9168, 5591 and 735 states, respectively.
We consider three different situations from different transition service rates (we assume single server semantics in all cases). In the first case, we suppose that the service transitions rates are those in the Fig. 4.a. In this case, the exact throughput of transition $T5$ is 0.778787.
In Table 1 we present the iteration steps of the method for this case. Columns $\mathcal{X}(Ti)$ are the estimated values for throughput of transition Ti at each iteration step. Columns 'scale f.' are the scale factors modifying the previous estimated service rates, computed with the basic skeleton. Convergence of the method is usually obtained from the fourth iteration step. The error for this example was -0.806%.
In the second case, we suppose all transitions rates equal to one in the original system (representing a symmetric timing case, in the sense that both sides of the system have response times of the same order of magnitude). The exact throughput of $T5$ is 0.217662. The error of the approximated value is -0.681%. On the other hand, if a very asymmetric timing is considered (service rates differ in three orders of magnitude for both subsystems, as given in the third case of Table 1), the exact throughput of $T5$ is 0.027686 leading up to an error of -0.282%.
With regard to the accuracy of the method in other cases, no formal proof gives positive answers, but extensive testing allows us to assure that the error of the method is usually below 5%. To initiate the iteration, our experience is that the method seems to be robust with respect to the initial values of the interface transitions.

5 CONCLUSIONS

The approximation technique presented here is mainly based on decomposition and aggregation of the obtained submodels. In the particular case of marked graphs (nets without weights), such aggregation can be achieved substituting sets of paths between transitions by places. In the case of weighted models, considered in this paper, it is necessary to preserve some complex structural characteristics in order to achieve good approximation results. The set of considered characteristics to be preserved arises from a trade-off between: (1) good representation of the submodel being aggregated, (2) small size of the number of nodes that must be added to the submodels for

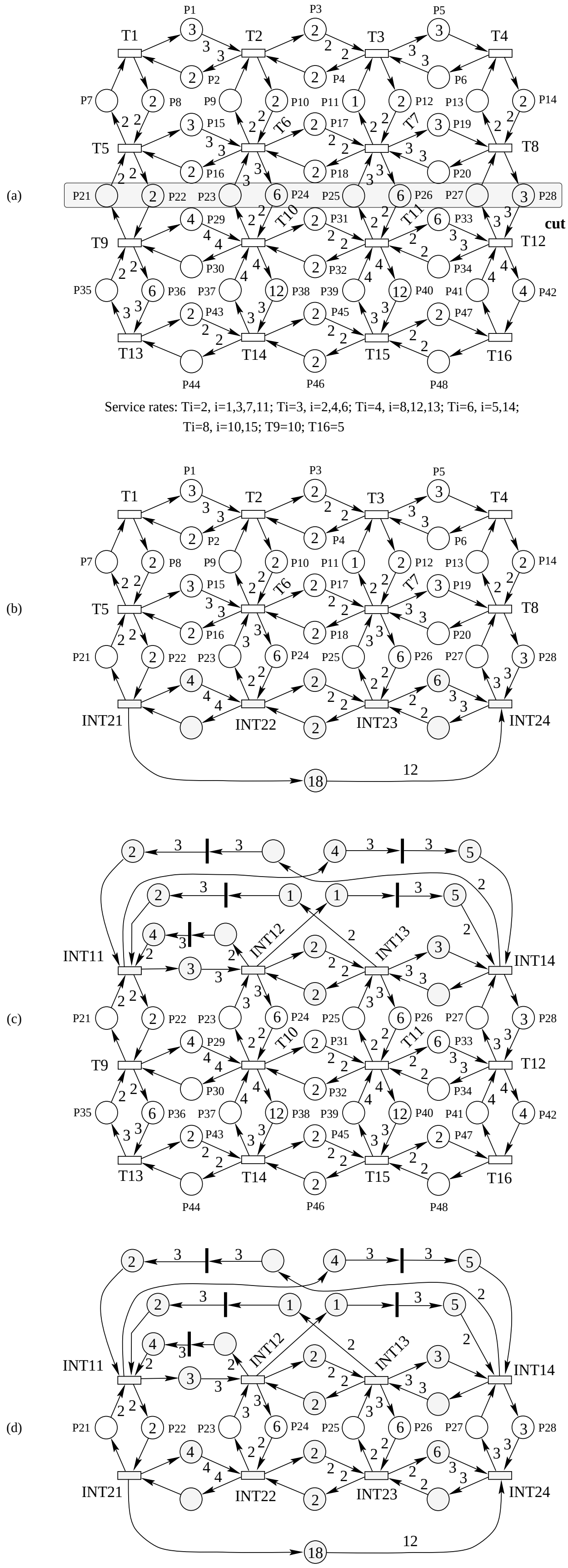

Figure 4: (a) A SWTS, its decomposition in aggregated subsystems (b) $\mathcal{AS}_1$ and (c) $\mathcal{AS}_2$, and (d) the basic skeleton.

Service rates as given in Fig. 4.a			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T5)	scale f.	$\mathcal{X}$(T9)	scale f.
0.808645	1.040916	1.496626	1.182425
0.773665	1.064321	1.542132	1.174937
0.772511	1.064686	1.544977	1.174788
0.772508	1.064674	1.545019	1.174793
0.772509	1.064674	1.545017	1.174793
Error: -0.806%			
Service rates equal to 1.0			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T5)	scale f.	$\mathcal{X}$(T9)	scale f.
0.244795	1.015023	0.425064	1.085438
0.216842	1.045162	0.432083	1.082689
0.216183	1.045339	0.432353	1.082672
0.216177	1.045314	0.432354	1.082683
0.216178	1.045339	0.432355	1.082678
Error: -0.681%			
Rates: $T1$ to $T8 = 10.0$; $T9$ to $T16 = 0.1$			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T5)	scale f.	$\mathcal{X}$(T9)	scale f.
0.039888	1.000000	0.055215	1.061891
0.027608	1.000000	0.055215	1.061891
0.027608	1.000000	0.055215	1.061891
Error: -0.282%			

Table 1: Iteration results for the SWTS in Fig. 4.a

preserving the characteristics, and (3) existence of an efficient algorithm to execute the aggregation. Our election was to preserve the gain, the maximum resistance and the minimum weighted marking between each pair of interface transitions. On the one hand, the preservation (after the aggregation) of the gain and of the minimum weighted marking assures liveness and boundedness of the obtained submodels. On the other hand, the preservation of the maximum resistance reflects part of the response time of the aggregated subsystem that is due to synchronization of tokens that must be consumed in batches by some transitions (in case of weighted input arcs to those transitions). A *dynamic programming* algorithm was presented to compute in polynomial time on the net size the three selected characteristics (gain, maximum resistance and minimum weighted marking between each pair of interface transitions).

With respect to the throughput approximation algorithm, we used an iterative *response time approximation* method, called *pelota* algorithm, that iteratively solves the CTMC underlying each of the aggregated subsystems and a basic skeleton. Each time that an aggregated subsystem is solved, new estimations for the interface transition rates of the other aggregated subsystems are computed. Extensive numerical experiments using this method showed very good results with respect to efficiency and accuracy. Convergency is generally observed after four iteration steps and the approximate computation of throughput can be achieved with a considerable saving of time and memory (more than one order of magnitude in many cases) and with a little error (less than 5%).

Even if the approximation technique was presented here for the particular case of weighted T–systems, larger net subclases can take advantage of this decomposition method. For instance, it can be applied to the performance evaluation of a class of cooperating sequential processes, called Deterministic Systems of Sequential Processes (see, e.g., [9]). These systems are more interesting from a modelling point of view since they include, in a controlled way, primitives to deal with concurrency, decisions, synchronization, blocking, and bulk movementes of jobs, therefore they add the possibility of modelling choices to the nets considered here.

References

[1] F. Baccelli, G. Cohen, and B. Gaujal. Recursive equations and basic properties of timed Petri nets. *Journal of Discrete Event Systems*, 1:415–439, 1991.

[2] J. Campos, J. M. Colom, H. Jungnitz, and M. Silva. Approximate throughput computation of stochastic marked graphs. *IEEE Transactions on Software Engineering*, 20(7):526–535, July 1994.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill, 1990.

[4] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman & Hall, London, 1993.

[5] G. Florin and S. Natkin. Les réseaux de Petri stochastiques. *Technique et Science Informatiques*, 4(1):143–160, February 1985.

[6] V. Mainkar and K. S. Trivedi. Fixed point iteration using stochastic reward nets. In *Proceedings of the Sixth International Workshop on Petri Nets and Performance Models*, pages 21–30, Durham, North Carolina, USA, October 1995. IEEE-Computer Society Press.

[7] A. Munier. Régime asymptotique optimal d'un graphe d'événements temporisé généralisé: application à un problème d'assemblage. *APII*, 27(5):487–513, 1993.

[8] T. Murata. Petri nets: Properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[9] C. J. Pérez-Jiménez, J. Campos, and M. Silva. Approximate throughput computation of a class of cooperating sequential processes. In *Proceedings of the Rensselaer's Fifth International Conference on Computer Integrated Manufacturing and Automation Technology (CIMAT'96)*, Grenoble, France, May 1996.

[10] C. J. Pérez-Jiménez, J. Campos, and M. Silva. State machine reduction for the approximate performance evaluation of manufacturing systems modelled with cooperating sequential processes. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, USA, April 1996.

[11] M. Silva and R. Valette. Petri nets and flexible manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 374–417. Springer-Verlag, Berlin, 1990.

[12] E. Teruel, P. Chrząstowski-Wachtel, J. M. Colom, and M. Silva. On weighted T-systems. In K. Jensen, editor, *Application and Theory of Petri Nets 1992*, volume 616 of *Lecture Notes in Computer Science*, pages 348–367. Springer-Verlag, Berlin, 1992.