APPROXIMATE THROUGHPUT COMPUTATION OF A CLASS OF COOPERATING SEQUENTIAL PROCESSES

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.

Summary

We concentrate on a family of Discrete Event Dynamic Systems (DEDS), modelled with Petri nets and obtained from a simple modular design principle, that include in a controlled way primitives to deal with concurrency, decisions, synchronization, blocking, and bulk movements of jobs. Many assembly systems with complex behaviours at the machine level can be described with the class of DEDS under study. We present a structure based decomposition technique and use a fixed-point search iterative process based on response time preservation of subsystems to approximate the throughput. An extensive battery of numerical experiments has shown that the error is less than 3%, and that the state space is usually reduced by more than one order of magnitude.

Keywords: Discrete event dynamic systems. Petri nets. Manufacturing systems. Performance evaluation.

1 Introduction

Petri nets are a formalism that has been shown to be adequate for the modelling of discrete event dynamic systems [SV90]. Analysis techniques for both the *validation of logical properties* of autonomous models and the *performance evaluation* of timed models have been developed [Bal95].

The goal of this paper is to present an approximation technique to compute the throughput of large models based on a decomposition principle, thus reducing the classical *state explosion problem*. The presented work is a mild generalization of that in [CCJS94].

The decomposition technique that we develop is deeply based on *Structure Theory* of Petri nets. We restrict ourselves to a particular net subclass that, being interesting from a modelling point of view, is restricted enough to permit that several, nice and powerful properties hold. *Deterministic systems of sequential processes* (DSSP's) are obtained by the application of a simple modular design principle: several functional units (in this case, sequential processes modelled with *state machines*, SM's) execute concurrently and cooperate using asynchronous communication by message passing through a set of buffers (places with possibly weighted input and output arcs). Buffers will be considered here origin and destination private (as in [Sou93]). Thus, in particular, competition among functional units is prevented. Moreover, buffers do not represent side conditions at the conflicts of the functional units (i.e., choices are free). In this paper, we deal with approximate throughput computation of DSSP's that include, in a controlled way, primitives to model *concurrency*, *decisions*, *synchronization*, *blocking*, and *bulk movements* of jobs.

In a previous paper [PJCS96], we considered

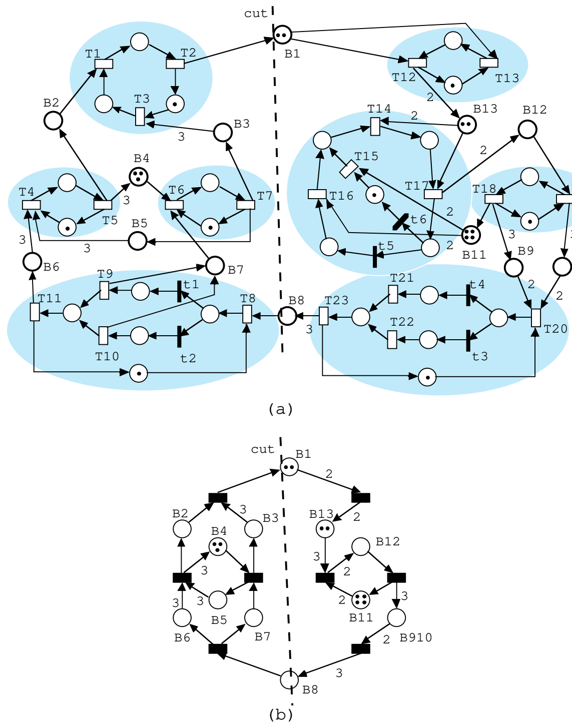

Figure 1: (a) A DSSP reducible to a WTS and (b) its WTS skeleton.

throughput approximation of DSSP's, based on *SM aggregation*. That technique is useful if the model contains large SM's, since in these cases the obtained decrease of state space can be significant. If the system is composed of a large number of sequential processes, additional transformations are needed to significatively decrease the state space size. In this paper, we perform such additional reduction by aggregating certain SM's to transitions which, if successfully done for all SM's, leads to a *weighted T–system* (WTS) skeleton (WTS's are the weighted extension of classical *marked graphs* [TCWCS92]).

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

From a modelling perspective, the class of nets considered here are naturally applied for describing *assembly systems* (see, for example, [Pro93]) working on a cyclic basis. In Fig. 1.a, the operations performed by different machines are modelled with SM's, while buffers (places Bi) represent intermediate stores. Fig. 1.b represents the WTS skeleton view of the system (now machines are represented just by black boxes, i.e., transitions). Assembly operations are performed by machines (or transitions, in the skeleton view) having more than one input buffer. An assembly operation puts parts together to obtain more complex ones. The weights on the input (resp., output) arcs of buffers 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).

The paper is organized as follows. In section 2, we present some basic Petri nets notations and the formal definitions of net subclasses that will be used later. Section 3 is devoted to define a subclass of DSSP's reducible to WTS's together with the reduction procedure itself (each state machine of the system is reduced to a single transition). In section 4, the structural decomposition of WTS's is presented. Section 5 includes the decomposition technique for DSSP's based both on the state machine reduction and on the WTS decomposition. Section 5 also includes a brief explanation of the iterative response time preservation technique that we used to approximate the throughput of transitions as well as some illustrative numerical results. Some concluding remarks are outlined in section 6.

2 Deterministic systems of sequential processes

Petri nets notations. We assume the reader is familiar with concepts of P/T nets. Here we only present notations used in later sections. For further extensions the reader is referred to [Mur89, Sil93].

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 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. M is a *home state* in $(\mathcal{N}, M_0)$ if it is reachable from every reachable marking.

State machines, WTS's, and DSSP's. We recall the definitions of the three net subclasses that are considered in the sequel. *State machines* (SM's) are ordinary Petri nets such that every transition has only one input and only one output place. SM's allow the modelling of sequences, decisions (or conflicts), and re-entrance (when they are marked with more than one token) but not synchronization.

Definition 2.1 (State machine) *A P/T net $\mathcal{N} = (P, T, Pre, Post)$ is a state machine (SM) if it is ordinary and $\forall t \in T$: $|{}^\bullet t| = |t^\bullet| = 1$.*

Weighted T–systems (WTS's) are the weighted generalization of *marked graphs* (see an example in Fig. 1.b). Even if some results for WTS are essentially parallel to those for the ordinary (non-weighted) case [TCWCS92], there are interesting differences that play an important role in the decomposition of WTS models.

Definition 2.2 (Weighted T–system) *A P/T system $(\mathcal{N}, M_0) = (P, T, Pre, Post, M_0)$ is a Weighted T–system (WTS) if $\forall p \in P : |{}^\bullet p| = |p^\bullet| = 1$.*

Deterministic systems of sequential processes (DSSP's) [Sou93, TSCC95] are used for the modelling and analysis of distributed systems composed by sequential processes communicating through input- and output-private buffers. Each sequential process is modelled by a *safe* (1–bounded) SM. The communication among them is described by *buffers* (places) which contain *products/messages* (tokens), which are produced by some processes and consumed by others. Each buffer is *output-private* (*input-private*) in the sense that it is an input (output) place of only one SM (see Fig. 1.a).

Definition 2.3 (Deterministic system of sequential processes) *A P/T system* $(\mathcal{N}, M_0) = (P_1 \cup \ldots \cup P_q \cup B, T_1 \cup \ldots \cup T_q, Pre, Post, M_0)$ *is a deterministic system of sequential processes (DSSP) if:*
1. $P_i \cap P_j = \emptyset$*,* $T_i \cap T_j = \emptyset$*,* $P_i \cap B = \emptyset$*,* $\forall i, j \in \{1, \ldots, q\}, i \neq j$*;*
2. $(\mathcal{SM}_i, M_{0i}) = (P_i, T_i, Pre_i, Post_i, M_{0i})$*,* $\forall i \in \{1, \ldots, q\}$ *is a strongly connected and 1–bounded state machine (where* Pre_i*,*$Post_i$*, and* M_{0i} *are the restrictions of* Pre*,* $Post$*, and* M_0 *to* P_i *and* T_i*);*
3. The set B *of buffers is such that* $\forall b \in B$*: (a)* $|^\bullet b| \geq 1$ *and* $|b^\bullet| \geq 1$*, (b)* $\exists i \in \{1, \ldots, q\}$ *such that* $b^\bullet \subset T_i$*, (c)* $\forall p \in P_1 \cup \ldots \cup P_q$*:* $t, t' \in p^\bullet \Rightarrow Pre(b, t) = Pre(b, t')$*.*

The first two items of the previous definition state that a DSSP is composed by a set of SM's ($\mathcal{SM}_i, i = 1, \ldots, q$) and a set of buffers ($B$). By item *3.a*, buffers are neither source nor sink places. The input- and output-private condition is expressed by *3.b*. Moreover, since the functional units of a DSSP are monomarked SM's, it can be said that buffers with origin and destination to a single functional unit can be ignored *without loosing expressive power*: (a) To preserve consistency and conservativeness, purely private buffers should be structurally implicit [CS91b], and (b) either the buffer is implicit (i.e., it does not constraint at all the behaviour of the global system) or not, and in the last case it kills the global system (and we are looking for live systems only!). Requirement *3.c* justifies the word "deterministic" in the name of the class: the marking of buffers does not disturb the decisions taken by a SM, i.e., choices in the SM's are free. From a queueing network perspective, DSSP's are a mild generalization of *Fork-Join Queuing Networks with Blocking* where servers are complex (safe SM's with a rich connectivity to buffers) and bulk movements of jobs are allowed [TSCC95].

In this paper, we consider a subclass of DSSP's that is *reducible to weighted T–systems*. For these systems, every SM can be reduced to a single transition, preserving the overall functional behaviour (Fig. 1). After such reduction, a WTS skeleton is obtained, and a decomposition technique for this subclass will be applied.

The following properties will be preserved after the above described reduction: boundedness (essential to the finiteness of the underlying CTMC of the stochastically timed models), liveness (for bounded, strongly connected DSSP's equivalent to deadlock-freeness [TSCC95], thus non-null throughput preservation) and the existence of home states (to preserve the CTMC ergodicity [TSCC95]). The decomposition phase will be performed in *polynomial time*.

3 State machine reduction to transition

The idea (see Fig. 2.a) is to reduce a SM of the DSSP to a single transition provided a unique buffer-neighbouring exists in a certain strict sense: depending on the way in which the buffers are connected to the machine and on its initial state, the machine can be reduced or not.

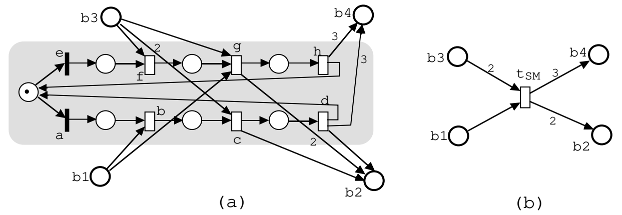

Figure 2: (a) Part of a DSSP. (b) The SM is reducible to a transition preserving boundedness, liveness, and the existence of home states for the full system.

Let $\mathcal{N} = (P, T, Pre, Post)$ be a DSSP net, $\mathcal{N}_{\mathcal{SM}} = (P_{\mathcal{SM}}, T_{\mathcal{SM}}, \Pi^-, \Pi^+)$ a SM subnet of $\mathcal{N}$ (i.e., $\Pi^- = Pre|_{P_{\mathcal{SM}} \times T_{\mathcal{SM}}}, \Pi^+ = Post|_{P_{\mathcal{SM}} \times T_{\mathcal{SM}}}$), $B_{in} = \{b_i | b_i^\bullet \subseteq T_{\mathcal{SM}}\}$ and $B_{out} = \{b_j | ^\bullet b_j \subseteq T_{\mathcal{SM}}\}$ the input and output buffers of $\mathcal{N}_{\mathcal{SM}}$ in $\mathcal{N}$. The reduction of $\mathcal{N}_{\mathcal{SM}}$ leads to a transition $t_{\mathcal{SM}}$ with $^\bullet t_{\mathcal{SM}} = B_{in}$ and $t_{\mathcal{SM}}{}^\bullet = B_{out}$, and some weights $\widetilde{Pre}(b_i, t_{\mathcal{SM}})$ for $b_i \in B_{in}$ and $\widetilde{Post}(b_j, t_{\mathcal{SM}})$ for $b_j \in B_{out}$.

Two conditions should be imposed in the reduction of $\mathcal{N}_{\mathcal{SM}}$ to get some "similarities" in the behaviour of the original and the reduced systems: (1) a token conservation property that allows the calculation of $\widetilde{Pre}(b_i, t_{\mathcal{SM}})$ and $\widetilde{Post}(b_j, t_{\mathcal{SM}})$ (explained in section 3.1) and (2) a token withdrawal (from input buffers)-token throw in (to output buffers) property from the initial marking of the SM (explained in section 3.2), that allows the preservation of liveness and the existence of home states.

3.1 The token conservation property

Firing $t_{\mathcal{SM}}$ in the reduced system, $M[t_{\mathcal{SM}}\rangle M'$, produces a token variation $\mu = M' - M$. It is obvious (assume, for example, $b_i \in B_{in}$) that

$$\mu(b_i) = k_{ij} \mu(b_j) \tag{1}$$

where:

- $k_{ij} = \widetilde{Pre}(b_i, t_{\mathcal{SM}}) / \widetilde{Pre}(b_j, t_{\mathcal{SM}})$, if $b_j \in B_{in}$.
- $k_{ij} = -\widetilde{Pre}(b_i, t_{\mathcal{SM}}) / \widetilde{Post}(b_j, t_{\mathcal{SM}})$, if $b_j \in B_{out}$.

For the computability of $\widetilde{Pre}$ and $\widetilde{Post}$, it is clear that equations similar to (1) should be written in the original system for "arbitrarily long" sequences on $T_{\mathcal{SM}}$:

$$\mu(b_i) = k_{ij} \mu(b_j) + \sum_{p_k \in P_{\mathcal{SM}}} \lambda_k \mu(p_k) \tag{2}$$

where k_{ij} is defined as in (1), while $\sum_{p_k \in P_{\mathcal{SM}}} \lambda_k \mu(p_k)$ represents the contribution of the marking variation induced on the places of the SM. Obviously, if the firing sequence projected on the SM leads to a T-semiflow of the state machine, then $\mu(p_k) = 0$.

Property 3.1 *Let* $C^* = \begin{pmatrix} C_{\mathcal{SM}} \\ C_{B_{in}} \\ C_{B_{out}} \end{pmatrix}$ *be the incidence submatrix of the full net* $\mathcal{N}$ *reduced to* $T_{\mathcal{SM}}$*,* $P_{\mathcal{SM}}$*,* *input buffers (B_{in}), and output buffers (B_{out}). Then the relative weights for the reduced net, are computable iff* rank$(C^*) = n_{\mathcal{SM}} (= |P_{\mathcal{SM}}|)$.

Proof: $\Rightarrow$) rank$(C_{\mathcal{SM}}) = n_{\mathcal{SM}} - 1$, because $\mathcal{N}_{\mathcal{SM}}$ is a state machine. Moreover, if $C_1 = \begin{pmatrix} C_{\mathcal{SM}} \\ C(b_i) \end{pmatrix}$, rank$(C_1) = n_{\mathcal{SM}}$, because b_i (any input, non-output buffer of $\mathcal{N}_{\mathcal{SM}}$) cannot be a linear combination of the places of the SM.

Consider now $C_2 = \begin{pmatrix} C_{\mathcal{SM}} \\ C(b_i) \\ C(b_j) \end{pmatrix}$, where b_j is another input or output buffer of $\mathcal{N}_{\mathcal{SM}}$. If rank$(C_2) = n_{\mathcal{SM}} + 1$, b_j is linearly independent of b_i and $C_{\mathcal{SM}}$ against (2). Therefore rank$(C_2) = n_{\mathcal{SM}}$. By induction on the remaining buffers in vicinity of $\mathcal{N}_{\mathcal{SM}}$ we conclude that rank$(C^*) = n_{\mathcal{SM}}$.

$\Leftarrow$) If rank$(C^*) = n_{\mathcal{SM}}$, rank$(C_{\mathcal{SM}}) = n_{\mathcal{SM}} - 1$ and rank$(C_1) = n_{\mathcal{SM}}$, it is clear that for any buffer in vicinity of $\mathcal{N}_{\mathcal{SM}}$, except b_i we can write: $C(b_j) = \sum_{p_k \in P_{\mathcal{SM}}} \lambda_k C(p_k) + k_{ij} C(b_i)$. Postmultiplying by any firable $\vec{\sigma}_{\mathcal{SM}}$ and taking into account the state equation of a net system, (2) is found. Q.E.D.

A dual perspective of the above rank property is the following. Let $X \geq 0$ be such that $C_{\mathcal{SM}} \cdot X = 0$ (i.e., a T-semiflow of $\mathcal{N}_{\mathcal{SM}}$). Therefore if $C(b_i) \cdot X = \alpha$, then $C(b_j) \cdot X = \alpha k_{ij}$, and so on. In words: Because minimal T-semiflows of SM's are cycles, all cycles of a SM reducible to a transition induce proportional effects (eventually null) on the set of input and output buffers.

From the above reasoning, the value of $\widetilde{Pre}(B_{in})$ or $\widetilde{Post}(B_{out})$ is not fully characterized, but it is known that:

- $\widetilde{Pre}(b_i) = \alpha$
- $\widetilde{Pre}(b_j) = \alpha k_{ij} \quad \forall b_j \in B_{in} \setminus \{b_i\}$
- $\widetilde{Pre}(b_q) = -\alpha k_{iq} \quad \forall b_q \in B_{out}$

The value of α is computed in the next section for the case where the SM is really reducible.

3.2 Token withdrawal and token throw in per cycle

Looking at Fig. 2.b, it is clear that the SM reduction makes sense if, starting in an initially marked place, for all cycles of the SM (i.e., its minimal T-semiflows), all token withdrawal from the input buffers occur before the first token throw in to the output buffers, and additionally the number of withdrawals is independent of the path followed to the first throw in to the output buffers.

Let $\overline{\mathcal{N}}_{\mathcal{SM}}$ be the net obtained from $\mathcal{N}_{\mathcal{SM}}$ by labelling each interface transition t with "$-b_i$" if b_i is an input buffer of t and "$+b_j$" if b_j is an output buffer of t. Moreover, the output places of transitions labelled with "$+b_k$" are blocked.

Definition 3.1 (SM reducible to a transition) *The state machine $\mathcal{N}_{\mathcal{SM}}$, initially marked at p_q, is reducible to $t_{\mathcal{SM}}$ if:*

i) rank $\begin{pmatrix} C_{\mathcal{SM}} \\ B_{in} \\ B_{out} \end{pmatrix} = |P_{\mathcal{SM}}|$ *(pure structural condition from property 3.1).*

ii) All transitions labelled with "$-$" in $\overline{\mathcal{N}}_{\mathcal{SM}}$ are reachable (without going through a blocked place) from p_q, and in all elementary paths from p_q to a blocked place the same number of occurrences of "$-b_k$" appears. The value of α is the number of occurrences of label "$-b_i$" in the paths from p_q to a blocked place.

The condition ii) in the above definition can be computed also in *polynomial time* using, for example, a *single to several* nodes path following algorithm derived from Floyd's algorithm.

For instance, for the net in Fig. 1.a, the SM defined by $T_{\mathcal{SM}} = \{t_5, t_6, T_{14}, T_{15}, T_{16}, T_{17}\}$ should be labelled with $-2B_{11}$ in $\{T_{15}, T_{16}\}$, $-2B_{13}$ in T_{14} and $-B_{13}$ and $+2B_{12}$ in T_{17}. Thus the input place of $t_5 - t_6$ is blocked. From the initial marking T_{16} is not reachable, thus another marking should be considered. Firing backwards t_6 (a persistent firing) does not introduce spurious behaviours and all cycles appear to be labelled with $-2B_{11}$ and $-3B_{13}$ (and $+2B_{12}$). Thus $\widetilde{Pre}(B_{11}, t_{\mathcal{SM}}) = 2$, $\widetilde{Pre}(B_{13}, t_{\mathcal{SM}}) = 3$, $\widetilde{Post}(B_{12}, t_{\mathcal{SM}}) = 2$.

Theorem 3.1 *Let $(\mathcal{N}, M_0)$ be a DSSP and $(\widetilde{\mathcal{N}}, \widetilde{M_0})$ another one obtained through the reduction of a SM to a transition, according with the conditions above. $(\widetilde{\mathcal{N}}, \widetilde{M_0})$ is conservative, live, and it has home states iff $(\mathcal{N}, M_0)$ does. Moreover, the transformation preserves the marking bounds of the buffers.*

Proof: (1) The reduction is a particular case of the elimination of transitions $T_{\mathcal{SM}}$ with the standard P-semiflow calculation algorithm [CS91a]. Therefore conservativeness is preserved.

(2) According with the reduction conditions, a SM is reducible iff for any T-semiflow (cycle) with external effect the same amount of tokens are withdrawn from the input buffers (possibly in different order) before starting the throw in of tokens on the output buffers, and this also for a quantity independent of the cycle being executed. Thus liveness and existence of home states can be equally considered in a two transitions SM: After the initially marked place, a transition resumes all tokens withdrawal from input buffers; its output place, π, is the SM precondition to a transition in which the postcondition resumes all tokens throw in the output buffers. Both reductions naturally preserve liveness and the existence of home states. Moreover, π can be eliminated fusing both transitions into a single one, also preserving liveness and the existence of home states.

(3) The bounds of the buffers are preserved due to the two phases on the aggregated behaviour, because before giving the first token all consumptions controlled from the SM are done and these input buffers have it as the unique destination. Q.E.D.

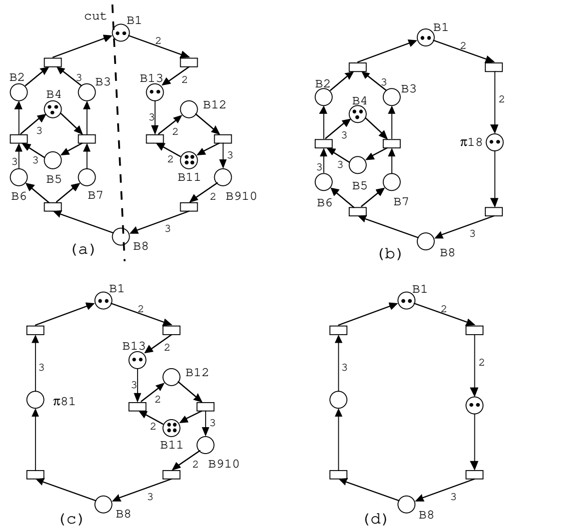

Figure 3: (a) A WTS, its decomposition in aggregated subsystems (b) $\mathcal{AS}_1$ and (c) $\mathcal{AS}_2$, and (d) the basic skeleton.

The rule presented so far reduces a SM, with the characteristics already considered, to a transition. This transformation technique preserves neither the projection of the languages nor the projection of the markings over the preserved nodes (as done for marked graphs [CCJS94]). For instance, in the SM defined by $T_{\mathcal{SM}} = \{t_5, t_6, T_{14}, T_{15}, T_{16}, T_{17}\}$ in Fig. 1.a, if we fire T_{15} and then T_{14} two tokens are removed from buffer B_{11} and then two tokens from buffer B_{13}. When the state machine is reduced to a transition it is impossible to separate these two states or firings. If this reduction is done for all the SM's in the DSSP we obtain a bounded and live WTS, for which it is also known, from other developments, that there exist home states [TCWCS92]. Moreover, the bounds of all places of the WTS are equal to the bounds of the corresponding buffers of the original DSSP.

4 On decomposition of weighted $T-$systems

In order to simplify the presentation, we restrict ourselves to a WTS decomposition obtained from a *single input-single output cut* (SISO-cut) through places. The method is a generalization of that presented in [JSS92, CCJS94] for marked graphs. A SISO-cut is defined by two places whose deletion generates a partition of the system into two unconnected subsystems [JSS92] (delete $B1$ and $B8$ in Fig. 3.a). Assume, to simplify this exposition, that each place of the cut is the only input (output) place of its output (input) transition. The technique presented here can be easily generalized to multiple input-multiple output cuts (as was done in [CCJS94]).

Once a SISO-cut has been decided, the main problem is to derive the *aggregated subsystems* where either the left or the right part of the original system is reduced to a "minimum number" of nodes preserving some interesting functional properties (the *basic skeleton* will be obtained after reduction of both parts). In

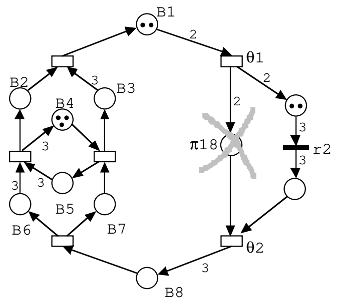

Figure 4: Improvement of the aggregated subsystem in Fig. 3.b with the resistance at transition $r2$.

Fig. 3.a, the right part (from the output transition of $B1$ to the input transition of $B8$) is replaced with a single place, π_{18} (see Fig. 3.b). The input and output weights of π_{18} are computed to preserve the *gain* of the paths in the original system from the output transition of $B1$ to the input transition of $B8$. The gain represents the average number of firings of the output transition of π_{18} per each single firing of the input transition of π_{18}. Note that the gain can be (structurally) computed in polynomial time and its preservation assures consistency and conservativeness preservation. The same reduction would be done for the left hand part.

After this reduction, the projection of the state space of the original system on the non-reduced part is not preserved in the aggregated subsystem (this fact differs from the marked graph case [CCJS94]). In other words, spurious markings can be made reachable after the aggregation. For instance, the aggregated subsystem obtained in Fig. 3.b allows a reachable marking with 12 tokens in $B8$, while in the original system (Fig. 3.a) that marking is non-reachable.

In order to eliminate some of the spurious markings we preserve not only the gain, but also what we call the maximum *resistance* of the aggregated part. The resistance of a path from a transition to another transition is a property of a path related to the minimum number of firings of the first transition that are needed to fire once the second. The resistance between two transitions is the resistance of a path with maximum resistance among all paths joining the two transitions. As the aggregated parts (in the case of a SISO cut) have only one input and one output transition we need to compute the path with maximum resistance joining them. The concept of resistance depends on the structure of the net and also on the initial marking. For the system in Fig. 3.a, to preserve the resistance of the subsystem in the right hand side with respect to Fig. 3.b, we add an immediate transition, $r2$, with its corresponding input and output place to obtain the aggregated subsystem in Fig. 4 (firing θ_1 once does not allow θ_2 to be fired as was the case in Fig. 3.b).

Sometimes the place preserving the gain is *implicit* with respect to the path with the immediate transition implementing the resistance and its input and output place (this is the case for the aggregated subsystem in Fig. 4), thus it can be deleted. In other cases, in order to preserve the resistance, it is not necessary to add the immediate transition with its input and output place because the place added for preserving the gain summarizes also the resistance (as in Fig. 3.c). In general, it can be necessary to include both the place for preserving the gain and the immediate transition for preserving the resistance for summarizing each side of the system.

The formal definition of gain and resistance follows.

Definition 4.1 (Gain, structural resistance, weighted marking and resistance) *Let* $wp = t_0p_1t_1p_2\ldots p_nt_n$ *denote a weighted path in a WTS with* $Post(p_i, t_{i-1}) = x_i$, $Pre(p_i, t_i) = y_i$, $i = 1, \ldots, n$ *and* $M_0[p_1], \ldots, M_0[p_n]$ *tokens in its places.*

i) The gain of wp *is* $g(wp) = \prod_{i=1}^{n} \frac{x_i}{y_i}$.

ii) The structural resistance of wp *is* $SR(wp) = \max_{i=1,\ldots,n} \prod_{j=1}^{i} \frac{y_j}{x_j}$.

iii) The weighted marking of $(wp, M_0[wp])$ *is* $WM(wp, M_0[wp]) = \sum_{i=1}^{n} \frac{\prod_{j=1}^{i-1} y_j}{\prod_{j=1}^{i} x_j} M[p_i]$.

iv) The resistance of $(wp, M_0[wp])$ *is* $R(wp, M_0[wp]) = SR(wp) - WM(wp, M_0[wp])$.

In the above definition, the structural part of the resistance is called *structural resistance*, while the initial marking contribution to the *resistance* is the *weighted marking*.

The problem now is to compute the gain and maximum resistance of a set of paths with the same initial and last transition. Note that all these paths must have the same gain (since we are considering consistent WTS's). To compute a path of maximum resistance we use a heuristic dynamic programming algorithm whose time complexity is *polynomial* on the number of nodes. We have not found yet any example in which the computed path is not the optimal one. Anyhow, if such an example existed, the solution of the algorithm would only give a path with less resistance than the optimal one, thus possibly cutting less spurious markings than eventually can be done (but this is not a problem).

Once we have selected the path with maximum resistance, we substitute the subnet by: (1) the initial and final transition of that path, (2) the transition in which the structural resistance of the path is achieved, and (3) two structurally implicit places (in the original net), with the initial marking needed to make them implicit (in the original system), joining the two pairs of transitions. The marking of the rest of the places of the aggregated subsystems is as follows: The marking of preserved places is the same that in the original system and the marking of the place summarizing the gain is the minimum weighted marking of a path joining the interface transitions. This technique assures the liveness and boundedness of the aggregated subsystems. Now, the marking bounds are not preserved due to buffers aggregation.

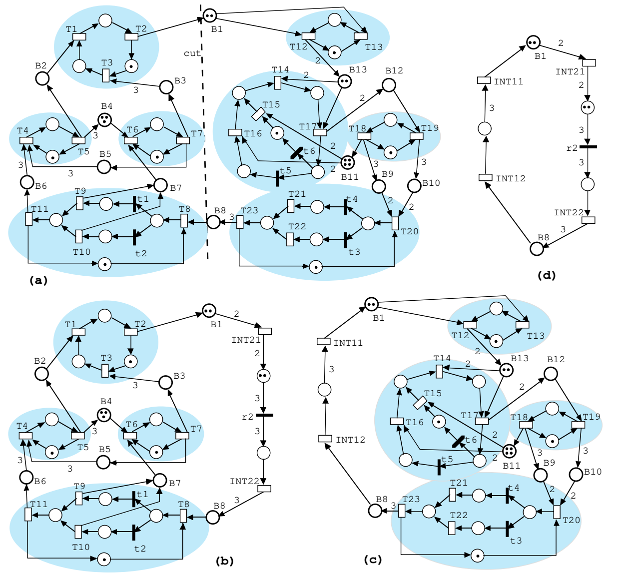

Figure 5: (a) A DSSP; its decomposition in aggregated subsystems (b) $\mathcal{AS}_1$ and (c) $\mathcal{AS}_2$, and (d) the basic skeleton. (Service rates: $Ti = 2.0, i = 5, 8, 12, 17, 23$; $Ti = 3.0, i = 6, 11, 15, 16, 22$; $Ti = 4.0, i = 4, 9, 13, 20$; $Ti = 5.0, i = 10, 14, 18, 19$; $Ti = 8.0, i = 7, 21$; $Ti = 10.0, i = 1, 2, 3$. Routing rates: $ti = 1, i = 1, 2, 3, 4, 5$; $t6 = 2$.)

5 DSSP decomposition and iterative response time approximation

In this section, we apply the state machine reduction (section 3) and the WTS decomposition (section 4) to a DSSP. After that, an iterative method for approximating the throughput of transitions of large stochastic DSSP models is considered (essentially the same used for marked graphs [CCJS94], called *pelota algorithm*).

From the original model and the cut definition (Fig. 5.a) two aggregated subsystems and a basic skeleton are derived. In one aggregated subsystem (Fig. 5.b), all the right hand side state machines are reduced to transitions and the resulting WTS is aggregated as explained in previous section. The other aggregated subsystem (Fig. 5.c) is obtained by reducing the left hand side state machines. In the basic skeleton (Fig. 5.d) the full aggregation is performed. This reduction technique does not preserve, in general, the projection of state spaces of the original system. This fact is (in particular) implied by theorem 3.1 for DSSP's. Even more, the additional WTS reduction leads to non preserving the marking bound of buffers. Nevertheless, boundedness of buffers is preserved.

Theorem 5.1 *Let* $(\mathcal{N}, M_0)$ *be a live and bounded DSSP reducible to a WTS,* $Q \subseteq P$ *a SISO-cut of* $\mathcal{N}$ *and* $\mathcal{AS}_1, \mathcal{AS}_2$ *and* $\mathcal{BS}$ *the aggregated subsystems and the basic skeleton defined by Q, preserving the gain and the resistance. Then,* $\mathcal{AS}_1, \mathcal{AS}_2$*, and* $\mathcal{BS}$ *are live, bounded, and have home state.*

Proof: To obtain $\mathcal{AS}_i$ or $\mathcal{BS}$ we take $\mathcal{N}$ and reduce some SM's to a single transition. By theorem 3.1 this reduction preserves liveness and boundedness. Then we can add the reduced paths. As these reduced paths are implicit we preserve liveness and boundedness. If we eliminate the rest of paths to obtain the aggregated subsystem we only eliminate restrictions to the output transition, so liveness is preserved for this class of nets. As the gain of reduced paths is the same than original ones, structural boundedness is also preserved. With respect to home state, its existence follows from the fact that live and bounded WTS's have home state [TCWCS92]. Q.E.D.

Now, we consider net systems with timed transitions. Marking and time independent exponentially distributed random variables are associated to the firing of transitions. For the modelling of conflicts (inside the SM's) we use *immediate transitions* with the addition of (marking and time independent) *routing rates*. Consequently, routing is completely decoupled from duration of activities.

The technique for an approximate computation of the throughput that we use now is, basically, a *response time approximation* method [ABS84, CCJS94]. The right hand side interface transitions in $\mathcal{AS}_1$ (Fig. 5.b), denoted T_{I_2}, approximate the response time of all the right hand side subsystem of the original model (Fig. 5.a). Symmetrically, the left hand side interface transitions in $\mathcal{AS}_2$ (Fig. 5.c), denoted T_{I_1}, approximate the response time of all the left hand side subsystem of the original model (Fig. 5.a). A direct (non-iterative) method to compute some service rates of interface 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 *pelota* algorithm [CCJS94].

Once a cut has been selected and given some initial values μ_t^0 for service rates of the interface transitions in T_{I_2}, the underlying CTMC of aggregated subsystem $\mathcal{AS}_1$ is solved. 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 T_{I_1} must be derived. To do that, we take the initial values μ_t^0 for service rates of transitions in T_{I_1} and we search in the basic skeleton a *scale factor* for 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 the other aggregated subsystem. The previous steps are repeated until *convergence* is achieved (if there exists).

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. For stochastic WTS's (with time and marking independent rates), the throughput of any transition is a monotonic function of the rate of any other transition.

Service and routing rates as given in Fig. 5.a			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T2)	rate(INT 11)	$\mathcal{X}$(T23)	rate(INT 22)
0.274443	0.343932	0.248674	0.722283
0.253678	0.355383	0.253769	0.723678
0.253753	0.355346	0.253753	0.723678
0.253753	0.355346	0.253753	0.723678
Error: 0.132%			
Service and routing rates equal to 1.0			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T2)	rate(INT 11)	$\mathcal{X}$(T23)	rate(INT 22)
0.090540	0.101778	0.079229	0.208813
0.080421	0.104019	0.080453	0.209198
0.080450	0.104014	0.080450	0.209198
0.080450	0.104014	0.080450	0.209198
Error: 1.199%			
Service rates: $\mathcal{N}_1$ all equal 100.0; $\mathcal{N}_2$ all equal 0.1 Routing rates equal to 1.0			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T2)	rate(INT 11)	$\mathcal{X}$(T23)	rate(INT 22)
0.077193	8.353201	0.023543	0.024763
0.023543	8.353201	0.023543	0.024763
0.023543	8.353201	0.023543	0.024763
Error: 0.004%			

Table 1: Iteration results for the DSSP in Fig. 5.a

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 WTS) converges.

Let us present some numerical values obtained for the system in Fig. 5. The exact value of the throughput of $T2$ in the original system for the service and routing rates given in the figure is 0.253419 (single-server semantics is assumed). That exact value was computed by solving the underlying CTMC with 57288 states, using GreatSPN [Chi87]. In Table 1, the iterative results are shown. Columns 'rate(INT i)' are the rates of transitions i computed with the basic skeleton. These transitions summarize the successive response time approximations of the corresponding aggregated subsystems. Convergence of the method is obtained from the third iteration step. The error for this example was 0.132% and the number of states of the underlying CTMC's of $\mathcal{AS}_1$, $\mathcal{AS}_2$, and $\mathcal{BS}$ are 6854, 2573, and 133, respectively.

The results obtained for alternative service and routing rates are also depicted in Table 1. If all rates are equal to 1.0 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 $T2$ is 0.079497. The error of the approximated value is 1.199%. 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 $T2$ is 0.023542 leading up to an error of 0.004%. Therefore, as we can expect, an exceptionally good approximation is obtained for very asymmetric systems with respect to timing.

Now the convergence and accuracy of the solution of the entire method should be addressed. Although no formal proof gives positive answers so far to the above questions, extensive testing allows us to conjecture that there exists one and only one solution, computable in a finite number of steps, typically between 2 and 5 if the convergence criterion is that the maximum difference between the two last estimations of the throughput of any aggregated subsystem is less than 0.01%. The state spaces are usually reduced in more than one order of magnitude. To initiate the iteration, our experience is that the method seems to be robust with respect to the value of the seed.

6 Conclusions

An aggregation technique has been presented as a basis for system decomposition. In this technique, valid for a subclass of DSSP's, each state machine of a part of the original system is fully reduced to a single transition, leading to a weighted T–system. Moreover, global aggregations of buffers and transitions are performed to increase the state space reduction. The obtained aggregated subsystems do not maintain projected state spaces. Nevertheless, basic functional properties like boundedness, liveness, and the existence of home states are preserved. The aggregation and decomposition can be done in polynomial time on the net size.

After the decomposition phase, a fixed-point search iterative process, that we call *pelota* algorithm, is used to approximate the throughput. Extensive numerical examples have shown that the iterative method converges in three to five steps. The state spaces are usually reduced in more than one order of magnitude and with very little error (less than 3% in all tested cases). Moreover, our experience is that the method seems to be truly robust with respect to the value of the seed needed to initiate the iteration. The results can be applied to evaluate the performance of assembly systems with machines with complex behaviours and bulk services and arrivals, working on a cyclic basis.

References

[ABS84] S. C. Agrawal, J. P. Buzen, and A. W. Shum. Response time preservation: A general technique for developing approximate algorithms for queueing networks. In *Proceedings of the 1984 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 63–77, Cambridge, MA, August 1984.

[Bal95] G. Balbo. Stochastic Petri nets: Accomplishments and open problems. In *Proceedings of the IEEE International Computer Performance and Dependability Symposium*, pages 51–60, Erlangen, Germany, April 1995. IEEE-Computer Society Press.

[CCJS94] 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.

[Chi87] G. Chiola. A graphical Petri net tool for performance analysis. In *Proceedings of the 3^{rd} International Workshop on Modeling Techniques and Performance Evaluation*, Paris, France, March 1987. AFCET.

[CS91a] J. M. Colom and M. Silva. Convex geometry and semiflows in P/T nets. A comparative study of algorithms for computation of minimal p-semiflows. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 79–112. Springer-Verlag, Berlin, 1991.

[CS91b] J. M. Colom and M. Silva. Improving the linearly based characterization of P/T nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 113–145. Springer-Verlag, Berlin, 1991.

[DHP+93] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman & Hall, London, 1993.

[JSS92] H. Jungnitz, B. Sánchez, and M. Silva. Approximate throughput computation of stochastic marked graphs. *Journal of Parallel and Distributed Computing*, 15:282–295, 1992.

[Mur89] T. Murata. Petri nets: Properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[PJCS96] 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.

[Pro93] J. M. Proth. *Principles of System Modeling*, chapter 2. In [DHP+93], 1993.

[Sil93] M. Silva. *Introducing Petri Nets*, chapter 1. In [DHP+93], 1993.

[Sou93] Y. Souissi. Deterministic systems of sequential processes: A class of structured Petri nets. In G. Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 406–426. Springer-Verlag, Berlin, 1993.

[SV90] 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.

[TCWCS92] 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.

[TSCC95] E. Teruel, M. Silva, J. M. Colom, and J. Campos. Functional and performance analysis of cooperating sequential processes. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, Esprit Basic Research Series, pages 52–65. Springer, 1995.