A new algorithm to compute synchronizing sequences for synchronized Petri nets

Marco Pocci, Isabel Demongodin, Norbert Giambiasi, Alessandro Giua

A new algorithm to compute synchronizing sequences for synchronized Petri nets

Marco Pocci*†, Isabel Demongodin*, Norbert Giambiasi*, Alessandro Giua*†
*Aix Marseille University, Laboratoire des Sciences de l'Information et des Systèmes, Marseille, France
† University of Cagliari, Department of Electrical and Electronic Engineering, Cagliari, Italy
(Email: {marco.pocci,isabel.demongodin,norbert.giambiasi, alessandro.giua}@lsis.org, giua@diee.unica.it)

***Abstract*—In basic testing problems, a topic of great interest is that of state identification. State identification has been solved by the way of synchronizing sequences, i.e., sequences that drive the system to a unique final state regardless of the initial one and do not require the observation of the system's outputs. In this paper we provide a novel approach for the computation of synchronizing sequences on discrete event systems modeled by synchronized Petri nets. This approach is compared with our previous results for synchronizing sequence computation by means of two different benchmarks. First, we generate random nets, second we provide a parametric manufacturing system.**

I. Introduction

Basic testing problems have been introduced in the pioneering work of Moore [1]. In this paper we focus on the synchronization problem. Synchronization concerns how to drive a system to a known state when its current state is unknown and the outputs are unobservable.

The classic approach to solve this problem considers systems modeled by finite automata [2]. In particular a standard technique requires the computation of a synchronizing sequence (SS), i.e., a sequence of inputs that drives the system to a unique final state independently of the initial state and does not require the observation of the system's outputs.

In our previous works [3], [4], we have dealt with this problem in the Petri net (PN) framework. We have shown how the automata approach [2] can be applied with minor changes to the class of bounded synchronized PNs. In this setting, one needs to construct the *reachability graph* (RG) $\mathcal{G}$ of the net — which describes the state-space of the net and depends on the initial marking — and its corresponding *auxiliary graph* of cardinality $\frac{1}{2}n(n+1)$, with $n = |\mathcal{G}|$[1]. We will refer to this approach as the RG approach.

We have considered a special class of synchronized PNs, called *state machine* (SM) PNs [5], where each transition has a single input and a single output place. For this class, we have proposed a novel approach which allows us to determine a SS for nets containing 1 token (*cf.* 1-SS), that can be used as a building block to construct a SS for the same net in the k-token case (*cf.* k-SS). Such a sequence is called *synchronizing transition sequence* (STS). The STS approach [3] constructs SSs with a depth-first search on the net structure and verifies certain conditions over the labeling function. This avoids the state-space explosion problem encountered in the RG-approach. However, not all SSs can be obtained in this way, because of the more restrictive required conditions.

The first contribution of this paper is a new approach that builds on results of the RG approach to construct all 1-SS that satisfy the conditions required by the STS approach. This allows one to determine a k-SS — for an arbitrary larger k — with no further computation, thus avoiding any reachability analysis.

The second contribution consists of a set of experiments results, which aim to compare our new approach with the two previously presented ones. Here, randomly generated SM PNs are taken into account. Furthermore they are applied to a family of manufacturing plants, which are not SMs.

The paper is organized as follows. In Section II a background on synchronized PNs is provided. In addition, the section provides the comparative scenarios where to finally show how to obtain a 1-SS for synchronized SM PNs. Section III presents a novel approach for SS computation. Section IV and Section V are devoted to compare the existing techniques for SS computation on PNs. Numerical results for SS computation are presented first by the aid of randomly constructed nets, then via a manufacturing example. Finally, in Section VI, conclusions are drawn and open areas of research are outlined.

II. Background

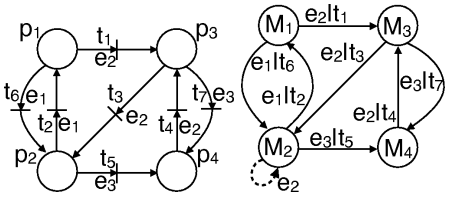
A. Synchronized Petri net formalisms

A Petri net (PN) is a structure $N = (P, T, Pre, Post)$, where P is the set of m places, T is the set of q transitions, $Pre: P \times T \to \mathbb{N}$ and $Post: P \times T \to \mathbb{N}$ are the pre and post incidence functions that specify the weighted arcs. An ordinary PN is a PN where $\forall p \in P, \forall t \in T,\ Pre(p,t),\ Post(p,t) \leq 1$. A *marking* is a vector $M: P \to \mathbb{N}$ that assigns to each place a nonnegative integer number of tokens; $M(p)$ is the marking of a place p. A marked PN is denoted $\langle N, M_0 \rangle$.

A transition t is enabled at M iff $M \geq Pre(\cdot, t)$. An enabled transition may be fired yielding the marking $M' = M + Post(\cdot,t) - Pre(\cdot,t)$. The set of enabled transitions at M is denoted $\mathcal{E}(M)$. $M[\sigma\rangle$ denotes that the sequence of transitions $\sigma = t_1 \dots t_k$ is enabled at M and $M[\sigma\rangle M'$ denotes that the firing of σ from M yields M'. A marking M is said to be *reachable* in $\langle N, M_0 \rangle$ iff there exists a firing sequence σ such that $M_0[\sigma\rangle M$. The whole markings reachable from M_0 defines the *reachability set* of $\langle N, M_0 \rangle$ and is denoted with $R(N, M_0)$.

The $preset$ and $postset$ of a place p (resp. transition t) are respectively denoted ${}^\bullet p$ and $p^\bullet$ (resp. ${}^\bullet t$ and $t^\bullet$). These

[1]Here by $|\mathcal{G}|$ we denote the cardinality of the RG nodes, which coincides with the number of reachable markings of the net.



(a) (b)

M_1	$[1\,0\,0\,0]^T$
M_2	$[0\,1\,0\,0]^T$
M_3	$[0\,0\,1\,0]^T$
M_4	$[0\,0\,0\,1]^T$

(c)

Fig. 1: A synchronized SM PN (a), its completely specified RG with one token (b) and the reachable markings (c).

notions can be easily extended to a set of places and a set of transitions. A *state machine* (SM) PN is an ordinary PN such that each transition t has exactly one input place and exactly one output place, i.e., $\forall t \in T$ it holds that $|{}^\bullet t| = |t^\bullet| = 1$.

Definition 1: A synchronized PN *[6] is a structure $\langle N, E, f\rangle$ such that: i) N is a PN; ii) E is an alphabet of input events; iii) $f : T \to E$ is a labeling function that associates with each transition t an input event $f(t)$.* ■

The labeling function is extended to sequences of transitions as follows: if $\sigma = t_1 t_2 \dots t_k$ then $f^*(\sigma) = f(t_1)f(t_2)\dots f(t_k)$. The set T_e of transitions associated with input event e is defined as follows: $T_e = \{t \mid t \in T, f(t) = e\}$. All transitions in T_e are said to be receptive to input event e.

A synchronized PN is driven by input sequences as follows. At marking M, transition $t \in T$ is fired iff:

1) it is enabled, i.e., $t \in \mathcal{E}(M)$;
2) the event $e = f(t)$ occurs.

On the contrary, the occurrence of an event associated with a transition $t \notin \mathcal{E}(M)$ does not produce any firing. Note that a single server semantic is here adopted, i.e., when input event e occurs, the enabled transitions in T_e fire only once regardless of their enabling degree. One writes $M \xrightarrow{w} M'$ to denote the fact that the application of input event sequence $w = e_1 \dots e_k$ from M drives the net to M'.

The following structural restriction is common in the literature to ensure the synchronized PN to be *deterministic*: $\nexists p \; s.t. \quad t, t' \in p^\bullet, \; t \neq t' \; and \; f(t) = f(t')$. When an event occurs in a deterministic net, all enabled transitions receptive to that event can simultaneously fire. Thus an input sequence $w = e_1 e_2 \cdots e_k \in E^*$ ($*$ is the kleen star) drives a deterministic net through the sequence of markings M_0, M_1, M_2, $\cdots$, M_k where M_0 is the initial marking and

$$M_{i+1} = M_i + \sum_{t \in T_{e_{i+1}} \cap \mathcal{E}(M_i)} (Post(\cdot, t) - Pre(\cdot, t)) .$$

A synchronized marked PN $\langle N, M_0, E, f\rangle$ is said to be bounded if there exists a positive constant k such that for all $M \in R(N, M_0)$, $M(p) \leq k$, $\forall p \in P$. Such a net has a finite reachability set. In this case, the behavior of the net can be represented by the *reachability graph* (RG), a directed graph whose vertices correspond to reachable markings and whose edges correspond to the transitions and the associated event causing a change of marking.In Fig. 1a an example of synchronized SM PN is shown. Note that labels next to each transition denote its name and the associated input event. Its RG for the one-token case — disregarding the dashed arc — it is shown in Fig. 1b.

In the rest of the paper, the reader will only deal with the class of synchronized deterministic PNs.

B. Previous approaches for synchronzing sequences

A synchronizing sequence (SS) it is a sequence that drives the model to a known state when its current state it is unknown and the outputs are unobservable. We recall the formal definition of SS in the framework of SM PNs.

Definition 2: [4] *Given a synchronized SM PN $\langle N, E, f\rangle$, assume that the initial marking M_0 is not given but is known to belong to a set $\mathcal{M}_0 = \{M \in \mathbb{N}^m \mid \sum_i M(p_i) = k\}$. $\bar{w}$ is called a k-SS if for all $M \in \mathcal{M}_0$ it holds $M \xrightarrow{\bar{w}} \bar{M}$.* ■

In [3] we have provided a first approach, namely the RG approach, for SS computation. This is a straightforward approach that consists in adapting the existing approach for automata and applying it to the RG $\mathcal{G}$ of the net, of n markings. That requires first to turn the RG $\mathcal{G}$ into a completely specified graph $\tilde{\mathcal{G}}$ — condition that in a RG of a PN is not always true — and to construct its auxiliary graph (AG) $\mathcal{A}(\tilde{\mathcal{G}})$. In Fig. 1b, the self-loop on marking M_2 is added to to turn the graph into a completely specified one.

We remind that $\mathcal{A}(\tilde{\mathcal{G}})$ has a node for every unordered pair (M_i, M_j) of markings of $\tilde{\mathcal{G}}$, including pairs (M_i, M_i) of identical markings. There is an edge from (M_i, M_j) to (M_p, M_q) labeled with an input event $e \in E$ iff in $\tilde{\mathcal{G}}$ there exists an arc from M_i to M_p and an arc from M_j to M_q, both labeled e. Note that, while constructing the AG, self-loops are generally omitted because non significant for the synchronization scope.

The cardinalities of those graphs can significantly increase with an increasing number of tokens and constructing a k-SS can be a very tricky problem. In fact, it can be shown that for a strongly connected SM PN with m places and k tokens the following equations hold:

$$|\mathcal{G}| = \begin{pmatrix} m+k-1 \\ m-1 \end{pmatrix} \leq \frac{1}{(m-1)!} k^{m-1}, \; |\mathcal{A}(\tilde{\mathcal{G}})| = \frac{|\tilde{\mathcal{G}}|(|\tilde{\mathcal{G}}|+1)}{2}$$

A more efficient technique, namely the *synchronizing transition sequence* (STS) approach [3], consists in determining first a 1-SS, that, under certain conditions, can be used as a building block to construct a k-SS. The STS approach allows to do so, while in the contrary an arbitrary SS constructed by the RG approach does not. This approach constructs any SS with a depth-first search on the net structure avoiding the state space enumeration. This avoids the state space explosion problem but not all SSs can be obtained in this way, because the required conditions are more restrictive.

III. A NEW APPROACH FOR SS COMPUTATION

In this section, a novel approach for SS computation on PNs is proposed. In a first step, we consider the net with one token and construct a modified and less complex reachability graph. The objective is to determine on this graph a 1-SS, which satisfies all conditions required by the STS approach. This allows one to determine a k-SS — for an arbitrary larger

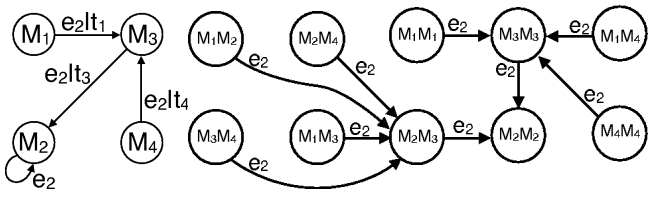


Fig. 2: The completely specified MRG $\tilde{\mathcal{G}}$ of the PN in Fig. 1a having $\bar{p} = p_2$ (a) and its AG $\mathcal{A}(\tilde{\mathcal{G}})$ (b).

k — with no further computation, thus avoiding any additional reachability analysis.

The 1-SS computation is based on the RG approach and ensures that the sole sequences satisfying conditions required by the STS approach, if any exists, are found. The proposed approach is a modified implementation of the RG approach, based on an arc-pruning. Given a target place $\bar{p}$, events labeling arcs outputting $\bar{p}$ — disregarding self-loops — are not considered. These events belong to the set of *forbidden events*, which is denoted as $\mathcal{F}(\bar{p})$.

For any given target place, we construct the so-called *modified reachability graph* (MRG) $\mathcal{G}_M$, obtained by removing all forbidden events from the RG of the net with only 1 token. Note that there is a different MRG for any given target place.

Example 3: Consider the synchronized PN of Fig. 1a with one token. Let the target place $\bar{p}$ be p_2. It holds that $\mathcal{F}(p_2) = \{e_1, e_3\}$. Its completely specified MRG $\tilde{\mathcal{G}}_M$ is shown in Fig. 2a, where the self-loop on marking M_2 is added to make it completely specified. The corresponding AG $\mathcal{A}(\tilde{\mathcal{G}}_M)$ is shown in Fig. 2b. ■

The MRG computation of 1-SS consists of three steps. First, we determine the set of forbidden events, which is used to construct the MRG and its AG; second, we apply the algorithm for the RG computation [3]. Finally we prove that the obtained sequence is a 1-SS for the complete model.

The so-constructed SS is a 1-SS, which leads the net to a target marking $\bar{M} \in R(N, M_0)$, where place $\bar{p}$ is the only marked place.

Algorithm 1: (MRG computation of a 1-SS on SM PNs)
Input: a synchronized SM PN $\langle N, E, f\rangle$ and a target place $\bar{p}$.
Ouput: a 1-SS $\bar{w}$ for a marking $\bar{M}$ such that $\bar{M}(p) = 1$ if $p = \bar{p}$ and 0 otherwise.

1. Let $\mathcal{F}(\bar{p}) \subseteq E$ be the set of forbidden events such that $\mathcal{F}(\bar{p}) = \{e \in E : \exists t \in \bar{p}^\bullet \backslash {}^\bullet\bar{p} \wedge f(t) = e\}$.
2. Construct the RG $\mathcal{G}$ for any $M_0 : M_0^T \cdot \vec{1} = 1$.
3. Construct the MRG $\mathcal{G}_M$, by removing all events $e \in \mathcal{F}(\bar{p})$ from $\mathcal{G}$.
4. Let $\tilde{\mathcal{G}}_M$ be the completely specified MRG.
5. Construct the corresponding AG $\mathcal{A}(\tilde{\mathcal{G}}_M)$.
6. Let $i = 0$.
7. Let $w = \varepsilon$, the empty initial input sequence.
8. Let $\phi(w) = \bigcup_i M_i \quad \forall M_i \in \tilde{\mathcal{G}}_M$, the initial current marking uncertainty.
9. Let $\bar{M}$ be the target marking, such that $\bar{M}(p) = 1$ if $p = \bar{p}$ and 0 otherwise.
10. While $\phi(w_i) \neq \{\bar{M}\}$, do
    - 10.1. $i = i + 1$.
    - 10.2. Pick two markings $M, M' \in \phi(w_{i-1})$, such that $M \neq M'$.
    - 10.3. **If** there exists no path in $\mathcal{A}(\tilde{\mathcal{G}}_M)$ from node (M, M') to $(\bar{M}, \bar{M})$, stop the computation. **Else** find the shortest path from node (M, M') to $(\bar{M}, \bar{M})$ and let w be the input sequence along this path, do
        - 10.3.1. $w_i = w_{i-1}w$.
        - 10.3.2. $\phi(w_i) = \{M : \forall M' \in \phi(w_{i-1}), M' \xrightarrow{w} M\}$;
11. $\bar{w} = w_i$ ■

The correctness of its results is proven by the following theorem.

Theorem 4: Sequences determined by the way of Algorithm 1 are 1*-SS for the considered synchronized SM PN.*

Proof: Let $\bar{w} = e_1 e_2 \cdots e_k$ be the sequence constructed by Algorithm 1, such that it drives the MRG through the sequence of markings M'_0, M'_1, ···, M'_k. We have to prove that: i)$\bar{w}$ drives the given synchronized PN through exactly the same sequence of markings; ii) point i) holds for every marking $M'_0 \in \mathcal{M}_0$.

First, we prove point i). Consider some marking M'_i. For any event $e_i \in \bar{w}$ occurrence, every transition $t \in T_{e_i}$ is fired, driving the net to marking M'_{i+1}. Since graph $\mathcal{G}_M$ takes into account all but the forbidden events, the same marking will be reached. Second, we prove point ii), by showing that at step 8. the current state uncertainty corresponds to the set of reachable markings of the net, which is necessary by Definition 2 of 1-SS. This proof is easily provided, since $\mathcal{G}_M$ has the same cardinality of $\mathcal{G}$, whose behavior is equivalent to the synchronized net itself. □

We show now that determining a 1-SS using Algorithm 1 allows us to readily determine a k-SS for an arbitrary large number of k tokens. That is because for such sequences the following sufficient condition holds.

Proposition 5: Consider a strongly connected synchronized SM PN $\langle N, E, f\rangle$ containing k tokens. Let $\bar{w}$ be a 1*-SS constructed via Algorithm 1 for a target marking $\bar{M}$, s.t. $\bar{M}(p) = 1$ if $p = \bar{p}$, 0 otherwise. Input sequence $\bar{w}^k$ — where $\bar{w}^k$ is the concatenation of k times $\bar{w}$ — is a k-SS for a target marking $\bar{M}_k$, s.t. $\bar{M}_k(p) = k$ if $p = \bar{p}$, 0 otherwise.*

Proof: Every sequence $\bar{w}$ constructed by Algorithm 1 satisfies the following condition: $(\forall t \in \bar{p}^\bullet \backslash {}^\bullet\bar{p})\ f(t) \notin w$. In other words sequence $\bar{w}$ does not contain any symbol labeling a transition that outputs place $\bar{p}$ and inputs any place $p \neq \bar{p}$. A first application of sequence w drives at least one token to $\bar{p}$. Any further application of w moves to $\bar{p}$ at least one of the tokens which is not contained by this place, and does not move the tokens already in $\bar{p}$, as none of its output transitions is receptive to any event in w. Here, we disregard self-loop transitions, as they do not change the marking of the net. □

IV. Numerical results for randomly constructed SM PNs

In this section we prove the efficiency of our novel algorithm by comparing it with our previous PN techniques for

m \ q	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1.000	1.000	1.000	0.899	0.944	0.925	0.977	0.944	0.987	0.987	1.000	0.987	1.000
3		1.000	0.867	0.716	0.892	0.899	0.793	0.815	0.915	0.899	0.973	0.915	0.951
4			0.797	0.797	0.758	0.746	0.867	0.778	0.797	0.833	0.850	0.883	0.867
5				0.737	0.757	0.716	0.815	0.758	0.915	0.778	0.815	0.716	0.902
6					0.786	0.852	0.738	0.725	0.852	0.858	0.887	0.917	0.887

TABLE I: $\mathcal{N}_{MRG\%}$

m \ q	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0.900	0.900	0.900	0.809	0.850	0.823	0.876	0.850	0.888	0.888	0.900	0.888	0.900
3		0.900	0.780	0.644	0.809	0.809	0.717	0.734	0.823	0.809	0.876	0.823	0.850
4			0.717	0.717	0.682	0.644	0.780	0.700	0.717	0.750	0.765	0.795	0.780
5				0.664	0.664	0.644	0.734	0.682	0.823	0.700	0.734	0.644	0.809
6					0.707	0.767	0.682	0.652	0.767	0.767	0.799	0.826	0.799

TABLE II: $\mathcal{N}_{STS\%}$

m \ q	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0.788	0.835	0.818	0.834	0.834	0.826	0.835	0.848	0.838	0.837	0.845	0.843	0.833
3		0.829	0.860	0.892	0.882	0.872	0.910	0.905	0.898	0.941	0.920	0.930	0.980
4			0.878	0.873	0.890	0.926	0.906	0.934	1.003	0.984	0.979	0.977	0.986
5				0.855	0.860	0.879	0.915	0.932	0.961	0.944	1.024	0.945	1.064
6					0.814	0.832	0.900	0.821	0.862	0.876	0.872	1.030	1.030

TABLE III: $\hat{\mathcal{T}}_{STS}/\hat{\mathcal{T}}_{MRG}$

m \ q	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
3		1.000	0.963	0.889	1.023	1.200	1.167	1.875	1.600	1.500	1.786	1.222	1.185
4			0.781	0.971	1.032	1.333	0.830	1.193	1.127	1.137	1.429	1.167	1.215
5				0.868	0.769	0.926	1.160	0.952	1.125	1.260	0.976	1.079	1.429
6					1.625	0.653	0.556	0.400	0.224	0.300	0.274	0.544	0.431

TABLE IV: $\hat{L}_{STS}/\hat{L}_{MRG}$

SS computation, i.e., the RG and the STS approach. Experimental results are carried out by applying these approaches to randomly generated nets and analyzing their performance. The model data and MATLAB programs can be downloaded from [7]. Simulations have been run on a mini Mac intel core Duo 2, 2.53 GHz processor, with 4 GB 1067 MhZ DDR3 RAM.

For selected values of m places and q transitions, we randomly generate 100 deterministic synchronized SM PNs having $2 \leq m \leq 7$ places, $m \leq q \leq 15$ transitions and $k = 1$ token. In all cases the input alphabet has a randomly chosen cardinality $\frac{q}{m} \leq f \leq q$. Note that $\frac{q}{m}$ is the minimal alphabet cardinality to ensure a SM PN with m places and q transitions to be deterministic. Note that all generated PNs are synchonizable, i.e., the RG-approach would succeed, computing a SS.

It has been proved [3] that the reachability condition of Algorithm 1 (see step 10.3.) is verified only when there exists only one ergodic component and there may exist a SS only for those states belonging to this ergodic component. That is why these simulations deal only with strongly connected PNs.

For each net a place is randomly selected and we use both Algorithm 1 and the STS approach to determine a SS to this place. The algorithms are compared by means of three performance indexes:

- $\mathcal{N}_{MRG\%}$, $\mathcal{N}_{STS\%}$: number of times the algorithm successfully terminates returning a SS over the total number of generated nets;
- $\hat{\mathcal{T}}_{MRG}$, $\hat{\mathcal{T}}_{STS}$: average time required to compute the SS;
- $\hat{L}_{MRG}$, $\hat{L}_{STS}$: average length of the SS.

Table I and Table II show the number of times a 1-SS has been found, using respectively the MRG and the STS approaches, over the whole number of constructed nets. In the previous sections we have mentioned that while the RG approach always determines a SS if any exists, the STS and MRG approaches may fail to do so, due to structural constraints. Hence the value in the table should be contained in the interval $[0, 1]$. We can observe, however, that all the Table 2 entries show a value greater than 65% and that results for the MRG approach are even better. In fact, this comparison shows that the MRG approach finds a solution in 10% of cases where the STS does not. This confirms that these results are not too restrictive.

Table III shows the ratio $\hat{\mathcal{T}}_{STS}/\hat{\mathcal{T}}_{MRG}$ between the average execution time to compute a SS using the STS and the MRG approach. Here we expect the STS approach to be more efficient — since the MRG approach enumerates the whole space-set reducing just the branching factor of the constructed

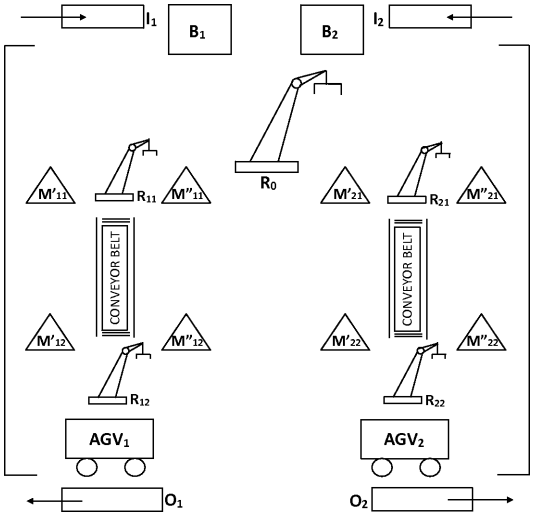


Fig. 3: Layout of the manufacturing system.

graphs — and this is confirmed from the fact that in almost all cases the table entries are smaller than one.

Table IV shows the ratio $\hat{L}_{STS}/\hat{L}_{MRG}$. Here we can see that for larger nets the STS approach produces shorter SS. This is due to the fact that the STS approach computes SS via a depth first search, while Algorithm 1 finds a solution that may be not the best one. In fact, at step 10.3. it finds a subsequence w that synchronizes at least two markings but does not pick those markings with any criterium.

V. Synchronizing sequences computation for PNs composed state machine subnets

In the following, we present some results on synchronized PNs which do not belong to the class of SM PNs but are composed by SM subnets. Since differences between the STS and MRG approach have already been highlighted, in order to evaluate the quality of the new proposed approach, a second experiment has been performed: a family of manufacturing systems — represented by a parametric PN — is analyzed by applying our techniques MRG and RG to find a SS.

A. Theoretical results

In this section we show how our approach can still be applied in this more general setting.

This family includes several classes of nets used to model *resource allocation systems* (RAS) [8], [9] including S^3PR nets, S^4PR nets, S^*PR nets, $NS-RAP$, $ERCN$–merged nets or PR nets. A broad and deep survey of the field can be found in [9].

We first give the theoretical results for nets containing several state machine subnets.

Proposition 6: [10] *Consider a synchronized PN* $\langle N, E, f\rangle$. *Let* $P_s \cup P_z = P$ *and* $T_s \cup T_z = T$, *where* $P_s = \bigcup_{i=1}^{n} P_{s,i}$ *and* $T_s = \bigcup_{i=1}^{n} T_{s,i}$ *(here* $\cup$ *denotes the union of disjoint subsets). These sets are such that for* $i = 1, 2, \dots n$ $N_{s,i} = (P_{s,i}, T_{s,i}, Pre_{s,i}, Post_{s,i})$ *is a strongly connected SM PN subnet.* $Pre_{s,i}$ *and* $Post_{s,i}$ *are the restrictions to* Pre *and* $Post$ *to* $P_{s,i} \times T_{s,i}$.

For every subnet $N_{s,i}$, *let* $\bar{w}_i$ *be a SS that drives the subnet* $N_{s,i}$ *to a target marking* $\bar{M}_{s,i}$. *The sequence* $\bar{w} = \bar{w}_1\bar{w}_2 \dots \bar{w}_n$ *drives* N *to a target marking* $\bar{M}$ *such that:*

$$\bar{M}(p) = \bar{M}'_{s,i} \; with \; \bar{M}_{s,i} \xrightarrow{\bar{w}_{i+1}\bar{w}_{i+2}\cdots\bar{w}_n} \bar{M}'_{s,i} \quad if \;\; p \in P_{s,i},$$

if the two following conditions hold:

i) $\{{}^\bullet T_z \cup T_z^\bullet\} \cap P_s = \varnothing$;

ii) $(\forall e \in \bar{w}_i)\; T_e \cap P_z^\bullet \bigcap_{j=1}^{i} T_{s,j} = \varnothing$. □

Any sequence $\bar{w}$ determined by the way of the previous proposition is a SS for the subnet N_s. It also drives the complete model N to a state where the marking of places in P_s is known, while in general nothing can be said about the marking of places in P_z.

The next section provides an example of application of such a technique.

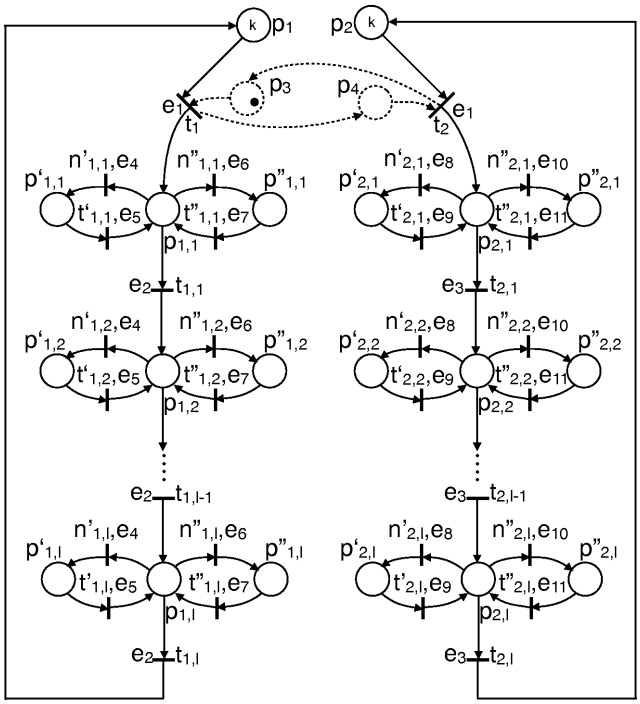


Fig. 4: Petri net model of the manufacturing system in Fig. 3

B. A manufacturing example

We consider a manufacturing plant consisting of two parallel and symmetric production lines, that produce two different kinds of final product. Each line i has a fixed number of k pallets, that limit the number of parts that can be under processing at a given time. A raw piece entering the system waits in the buffer (not modeled) until a pallet becomes available. Robot R_0 feeds the two lines alternatively, taking one part from the buffer and mounting it on an empty pallet.

In each line there are two workstations, which are composed by two machines and one robot. A workstation can process several pallets at a time.

For instance consider line 1. The pallet entering the system is deposed by R_0 on a conveyor belt and is moved to the first worstation where machines $M'_{1,1}$ and $M"_{1,1}$ perform their operation as requested. Robot R_{11} moves the pallet from the conveyor belt to the machines and viz.

Once the operations required on the first workstation have been completed, the pallet is put again onto the conveyor belt and moved to the second workstation where the processing is repeated. After processing, parts are unloaded from the pallets by robot $R_{1,2}$ and put on an AGV that moves them to the output buffer O_1 (not represented in Fig. 4).

The system's layout is shown in Fig. 3: we say that such a plant has production length $l = 2$ because each line is composed by a series of two workstations. We can consider a parameterized family of plants of this type, assuming that the number of pallets k and length l of the production lines may vary. For this family of plants, we obtain the synchronized Petri net depicted in Fig. 4.

This Petri net has $4 + 6l$ places and $2 + 10l$ transitions. The marking of places p_1 and p_2 represents number of the empty pallets in each line, the marking of place p_3 (resp. p_4) denotes that robot R_0 is ready to move a pallet to the left (resp. right) production line.

Consider machines M'_{ij} and $M"_{ij}$, which compose workstation j in line i. Transition $n'_{i,j}$ (resp. $n"_{i,j}$) represents the loading of a pallet from the conveyor belt queue to machine M'_{ij} (resp. $M"_{ij}$) while transition $t'_{i,j}$ (resp., $t"_{i,j}$) represents the unloading. Tokens in place p'_{ij} (resp. $p"_{ij}$) denote pallets loaded on machine M'_{ij} (resp. $M"_{ij}$).

The firing of transition $t_{i,j}$ denotes the transfer of a pallet to the next workstation.

The PN in Fig. 4, without taking into account dashed places and arcs, is composed by two SM PNs. Hence, according to Proposition 6, $P_s = P_{s,1} \cup P_{s,2} = P \backslash \{p_3, p_4\}$, $P_{s,1} = \{p_1, p_{1,1}, p'_{1,1}, p''_{1,1}, \ldots p_{1,l}, p'_{1,l}, p''_{1,l}\}$, $P_{s,2} = \{p_2, p_{2,1}, p'_{2,1}, p''_{2,1}, \ldots p_{2,l}, p'_{2,l}, p''_{2,l}\}$ and $T_s = T$.

We look for a SS that from an arbitrary state can empty the system, thus moving all empty pallets to the input buffers.

Results obtained using the MRG and the RG approach are shown in Table V, where required time of the SS are summarized for different values of m and l. Note that here only one simulation per setting has been performed. Also, the table shows the cardinality of the RG ($|\mathcal{G}|$), of the MRG ($|\mathcal{G}_M|$) and of the corresponding AGs. These are important parameters to understand the limits of the RG approach, while exhaustively enumerating the set space of the net. Note that these values do not correspond, because Proposition 6 applies so that SM subnets can be analyzed.

Note that the table shows also non-numerical values where the corresponding result cannot be provided: i) *out of time* (o.t.), when the corresponding value has not been computed within 6 hours; ii) *not computable* (n.c.), if the corresponding value cannot be computed: e.g., the RG is o.t. and the corresponding AG cannot be evaluated.

		RG			MRG		
k	l	$\|\mathcal{G}\|$	$\|\mathcal{A}(\mathcal{G})\|$	time $[s]$	$\|\mathcal{G}_M\|$	$\|\mathcal{A}(\mathcal{G}_M)\|$	time $[s]$
1	1	32	528	3.43	4	10	0.42
1	2	98	4851	*o.t.*	7	28	0.46
1	3	200	20100	*o.t.*	10	55	0.92
1	4	338	57291	*o.t.*	13	91	1.57
2	1	200	20100	*o.t.*	4	10	0.42
2	2	1568	*o.t.*	*n.c.*	7	28	0.46
2	3	*o.t.*	*n.c.*	*n.c.*	10	55	0.92
2	4	*o.t.*	*n.c.*	*n.c.*	13	91	1.57
3	1	800	320400	*o.t.*	4	10	0.42
3	2	*o.t.*	*n.c.*	*n.c.*	7	28	0.46
3	3	*o.t.*	*n.c.*	*n.c.*	10	55	0.92
3	4	*o.t.*	*n.c.*	*n.c.*	13	91	1.57

TABLE V: Time results for a manufacturing system.

The table denotes, for an increasing number of tokens, that the RG approach goes almost always o.t., due to a significant larger space state. On the contrary, the required time does not change with the MRG approach, that always finds a solution.

VI. Conclusion

This paper has two main contributions. First, we provide a new method that allows to determine a k-SS for the class of synchronized state machine PNs. Second, we provide a set of experimental results. Models data and MATLAB programs can be downloaded from [7]. The results show the advantages of our approaches, rather than the adaptation of the automata based approach. These approaches significantly reduce the computation time for nets with a large number of tokens, hence with a large state space. There is an open line for interesting future works. We plan to extend our MRG approach to any synchronized PN which do not belong to the class of SM PNs.

References

[1] E. F. Moore, "Gedanken-experiments on sequential machines," *Automata Studies, Annals of Mathematical Studies*, vol. 34, pp. 129 – 153, 1956.

[2] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machine – a survey," *Proc. of the IEEE*, vol. 84, no. 8, August 1996.

[3] M. Pocci, I. Demongodin, N. Giambiasi, and A. Giua, "Testing discrete event system: Synchronizing sequences using Petri nets," in *22nd European Modelling and Simulation Symposium*, FES, Morocco, Oct. 2010.

[4] ——, "Synchronizing sequences on not strongly connected Petri nets," in *Symposium On Theory of Modelling and Simulation (DEVS/TMS'11)*, Boston, MA, USA, 2011.

[5] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541 –580, apr 1989.

[6] R. David and H. Alla, *Discrete, Continuous and Hybrid Petri Nets*. Springer-Verlag, 2004.

[7] M. Pocci, The MATLAB toolbox is available at the following web address: http://www.lsis.org/poccim/PN_SYNCHRO.zip. September, 2012.

[8] J. Colom, "The resource allocation problem in flexible manufacturing systems," in *Applications and Theory of Petri Nets 2003*, ser. Lecture Notes in Computer Science, W. Aalst and E. Best, Eds. Springer Berlin Heidelberg, 2003, vol. 2679, pp. 23–35.

[9] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[10] M. Pocci, I. Demongodin, N. Giambiasi, and A. Giua, "Testing experiments on synchronized Petri nets," in *IEEE Transactions on Automation Science and Engineering*, 2013.