24

Batch Deterministic and Stochastic Petri Nets and Transformation Analysis Methods

Labadi Karim[1], Amodeo Lionel[2], and Chen Haoxun[2]
[1]*Ecole d'Electricité, de Production et de Méthodes Industrielles (Cergy-Pontoise)*
[2]*Université de Technologie de Troyes (Troyes)*
France

1. Introduction

Industrial systems such as production systems and distribution systems are often characterized as batch processes where materials are processed in batches and many operations are usually performed in batch modes to take advantages of the economies of scale or because of the batch nature of customer orders. The Batch Deterministic and Stochastic Petri Nets (BDSPN) is a class of Petri nets recently introduced for the modelling, analysis and performance evaluation of such systems which are discrete event dynamic systems with batch behaviours. The BDSPN model enhances the modelling and analysis power of the existing discrete Petri nets. It is able to describe essential characteristics of logistics systems (batch behaviours, batch operational policies, synchronization of various flows, randomness) and more generally discrete event dynamic systems with batch behaviours. The model is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes) and is capable of describing activities such as customer order processing, stock replenishment, production and delivery in a batch mode. The capability of the model to meet real needs is demonstrated through applications to modelling and performance optimization of inventory systems (Labadi et al., 2007,2005) and a real-life supply chain (Amodeo et al., 2007; Chen et al., 2005,2003).
Graph transformation is a fundamental concept for analysis of the systems described by graphs. The state of the art reporting for languages, tools and applications for graph transformation is given in the "Handbook of Graph Grammars and Computing by Graph Transformation" (Ehrig et al., 1999). In contrast to most applications of the graph transformation approach, where the states of a system are denoted by a graph, and transformation rules describe the state changes and the dynamic behaviour of systems, in the area of Petri nets (Murata, 1989) we apply transformation rules to modify a net in a stepwise way. This kind of transformation for Petri nets is considered to be a vertically structural technique, known as rule-based net transformation. This approach has been applied to various Petri net models such as basic Petri nets (Lee-Kwang et al., 1985, 1987), timed Petri nets (Juan et al., 2001; Wang et al. 2000), stochastic Petri nets (Ma & Zhou, 1992; Li-Yao et al. 1995), and coloured Petri nets (Haddad, 1988). There are different types of reduction/transformation techniques proposed in the literature. As we know, the reduction

is generally applied to resolve the state explosion problem of Petri nets. It aims at reducing the size of a Petri net model while retaining important properties of the model, such as liveness, safety, and boundeness. We can also find the work which transforms a Petri net model into another model such as UML (unified modelling language), diagrams, max + algebra model or vice versa. The objective of such a transformation is to use analysis methods of the resultant model to analyze the original model. Finally, one class of Petri nets may be transformed into another class in order to use theoretical results and analysis methods of the latter class to analyze the former class. A typical example of such a transformation is the transformation of a coloured Petri into an ordinary Petri net.
This chapter is organized into two parts. The first part is dedicated to a general description of the BDSPN model. A formal description of the model and its dynamic execution rules are presented with some illustrative examples. The capability of the model for modelling discrete event dynamic systems with batch behaviours is demonstrated through these examples. The second part of this chapter presents our recent work on structural and behavioural analysis of the BDSPNs by transforming them into other Petri nets. Two transformation analysis methods for the model are developed. The first method transforms a BDSPN into an equivalent classical discrete Petri net under some conditions. In this case, the corresponding transformation procedures are presented. For other cases, especially for BDSPNs with variable arc weights depending on their marking, the transformation is impossible. The second method analyzes a BDSPN based on its associated discrete Petri net which is obtained by converting the batch components (batch places, batch tokens, batch transitions) of the BDSPN into discrete components of the discrete Petri net. We show that although a BDSPN and its associated discrete Petri net behave differently, they have several common qualitative properties. This study establishes a relationship between BDSPNs and classical discrete Petri nets and demonstrates the necessity of the introduction of the BDSPN model.

2. Fundamentals of the BDSPN model

BDSPN extend Deterministic and Stochastic Petri Net (DSPN) (Lindemann, 1998; Ajmone Marsan et al. 1995) by introducing batch components, new transition enabling and firing rules, and specific policies defining the timing concept of the model. A BDSPN consists of places, transitions, and arcs that connect them. As shown in Fig. 1, a BDSPN has two types of places: discrete places and batch places. *Discrete places* can contain *discrete tokens* as in standard Petri nets. *Batch places* can contain *batch tokens* which are represented by Arabic numbers that indicate the sizes of the tokens. The current state of the modelled system (the marking) is given by the number of tokens in each discrete place and a list of positive integers in each batch place. Transitions are active components. They model activities which can occur (by firing transitions), thus changing the state of the system. Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled. When the transition fires, it removes tokens from its input places and adds some at all of its output places. The *enabling* and the *firing* of a transition depends on the cardinality of each arc, and on the current marking of each input places allowing the synchronization of discrete and batch token flows in the model. Since transitions are often used to model activities (production, delivery, order, etc.), transition enabling duration corresponds to activity execution and transition firing corresponds to activity completion.

Hence, a timing concept is naturally included into the formalism of the model. In a BDSPN, three types of transitions can be distinguished depending on their associated delay: *immediate transitions* (no delay), *exponential transitions* (delay is an exponential distribution), and *deterministic transitions* (delay is fixed). In this section, we recall the basic definition and the dynamical behaviour of the model.

2.1 Definition of the model

A BDSPN is a nine tuple ($P, T, I, O, V, W, \Pi, D, \mu_0$) where :

- $P = \{p_1, p_2, \ldots, p_m\} = P_d \cup P_b$ is a finite set of places consisting of the discrete places in set P_d and the batch places in set P_b. Discrete places and batch places are represented by single circles and squares with an embedded circle, respectively. Each token in a discrete place is represented by a dot, whereas each batch token in a batch place is represented by an Arabic number that indicates its size.
- $T = \{t_1, t_2, \ldots, t_n\} = T_i \cup T_d \cup T_e$ is a set of transitions consisting of immediate transitions in set T_i, the deterministic timed transitions in set T_d, and exponentially distributed transitions in set T_e. T can also be partitioned into $T_d \cup T_b$: a set of discrete transitions T_d and a set of batch transitions T_b. For simplicity, here we abuse the notation T_d which is used for both the set of deterministic timed transitions and the set of discrete transitions in case of non confusion. A transition is said to be a batch transition (respectively a discrete transition) if it has at least an input batch place (respectively if it has no input batch place).
- $I \subseteq (P \times T)$, $O \subseteq (T \times P)$, and $V \subseteq (P \times T)$ define the input arcs, the output arcs and the inhibitor arcs of all transitions, respectively. It is assumed that only immediate transitions are associated with inhibitor arcs and that the inhibitor arcs and the input arcs are two disjoint sets.
- W is the arc weight function that defines the weights of the input, output, and inhibitor arcs. For any arc (i, j), its weight $w(i, j)$ is a linear function of the M-marking with integer coefficients α, β, i.e., $w(i, j) = \alpha_{ij} + \sum_{p \in P} \beta_{(i, j)p} \times M(p)$. The weight $w(i, j)$ is assumed to take a positive value.
- Π: $T \rightarrow IN$ is a priority function assigning a priority to each transition. Timed transitions are assumed to have the lowest priority, i.e.; $\Pi(t) = 0$ if $t \in T_d \cup T_e$. For each immediate transition $t \in T_i$, $\Pi(t) \geq 1$.
- D: $T \rightarrow [0, \infty)$ defines the firing times of all transitions. It specifies the mean firing delay for each exponential transition, a constant firing delay for each deterministic transition, and a zero firing delay for each immediate transition
- μ_0 is the initial μ-marking, a row vector that specifies a multiset of batch tokens for each batch place and a number of discrete tokens for each discrete place.

The state of the net is represented by its *μ-marking*. We use two different ways to represent the μ-marking of a discrete place and the μ-marking of a batch place. The first marking is represented by a nonnegative integer as in standard Petri nets, whereas the second marking is represented by a multiset of nonnegative positive integers. The multiset may contain identical elements and each integer in the multiset represents a batch token with a given size. Moreover, for defining the net, another type of marking, called *M-marking*, is also introduced. For each discrete place, its M-marking is the same as its μ-marking, whereas for each batch place its M-marking is defined as the total size of the batch tokens in the place.
The state or μ-marking of the net is changed with two types of transition firing called "*batch*

firing" and "*discrete firing*". Whether the firing of a transition is batch firing or discrete firing depends on whether the transition has batch input places. To introduce batch firing, we need some notations. A place connected with a transition by an arc is referred to as input, output, and inhibitor place, depending on the type of the arc. The set of input places, the set of output places and the set of inhibitor places of transition t are denoted by ${}^{\bullet}t$, $t^{\bullet}$, and ${}^{\circ}t$, respectively, where ${}^{\bullet}t = \{ p \mid (p, t) \in I \}$, $t^{\bullet} = \{ p \mid (t, p) \in O \}$, and ${}^{\circ}t = \{ p \mid (p, t) \in V \}$. The weights of the input arc from a place p to a transition t, of the output arc from t to p are denoted by $w(p, t)$, $w(t, p)$ respectively.

2.2 Batch enabling and firing rules

A batch transition t is said to be enabled at μ-marking μ if and only if there is a *batch firing index* (positive integer) $q \in IN$ ($q > 0$) such that:

$$\forall p \in {}^{\bullet}t \cap P_b, \exists b \in \mu(p) : q = b / w(p,t) \tag{1}$$

$$\forall p \in {}^{\bullet}t \cap P_d, M(p) \geq q \times w(p,t) \tag{2}$$

$$\forall p \in {}^{\circ}t, M(p) < w(p,t) \tag{3}$$

The batch firing of t leads to a new μ-marking μ':

$$\forall p \in {}^{\bullet}t \cap P_d, \mu'(p) = \mu(p) - q \times w(p,t) \tag{4}$$

$$\forall p \in {}^{\bullet}t \cap P_b, \mu'(p) = \mu(p) - \{q \times w(p,t)\} \tag{5}$$

$$\forall p \in {}^{\bullet}t \cap P_d, \mu'(p) = \mu(p) + q \times w(t,p) \tag{6}$$

$$\forall p \in {}^{\bullet}t \cap P_b, \mu'(p) = \mu(p) + \{q \times w(t,p)\} \tag{7}$$

A batch transition t is said to be *enabled* if : (*i*) Each batch input place p of the transition has a batch token with size b such that all these batch tokens have the same batch firing index q defined as $b/w(p, t)$ for the transition, (*ii*) Each discrete input place of the transition has enough tokens to simultaneously fire the transition for a number of times given by the index, (*iii*) The number of tokens in each inhibitor place of the transition is less than the weight of the inhibitor arc connecting the place to the transition. For any batch output place, the firing of an enabled batch transition generates a batch token with the size given by the multiplication of the batch firing index and the weight of the arc connecting the transition to the batch place. For any discrete output place, the firing of the transition generates a number of discrete tokens with the number given by the multiplication of the discrete firing index and the weight of the arc connecting the transition to the discrete place.

2.3 Batch firing example

To well understand the mathematical and intuitive meaning of the batch transition firing of the BDSPN model, consider the net in Fig.1 describing an assembly-to-order system that requires two components. In the model, discrete places p_1 and p_2 are used to represent the stock of component A and the stock of component B respectively. Batch place p_3 is used to represent batch customer orders with different and variable sizes. To fill a customer order of size b, we need $b \times w(p_1, t_1) = 2b$ units of component A from the stock represented by p_1 and $b \times w(p_2, t_1) = b$ units of component B from the stock represented by p_2. These components will be assembled to b units of final product to fill the order. For instance, at the current μ-marking $\mu_0 = (4, 3, \{4, 2, 3\}, \varnothing, 0)^T$, it is possible to fill the batch customer order $b = 2$ in batch place p_3 since the batch transition t_1 is enabled with $q = b / w(p_3, t_1) = 2$. After the batch firing of transition t_1 (start assembly), the corresponding batch token $b = 2$ will be removed from batch place p_3, $q \times w(p_1, t_1) = 4$ discrete tokens will be removed from discrete place p_1, and $q \times w(p_2, t_1) = 2$ discrete tokens will be removed from discrete place p_2. A batch token with the size equal to $q \times w(t_1, p_4) = 2$ will be created in batch place p_4 and two discrete tokens will be created in discrete place p_5. Therefore, the new μ-marking of the net after the batch firing is: $\mu_1 = (0, 1, \{4, 3\}, \{2\}, 2)^T$ and its corresponding M-marking is $M_1 = (0, 1, 7, 2, 2)^T$.

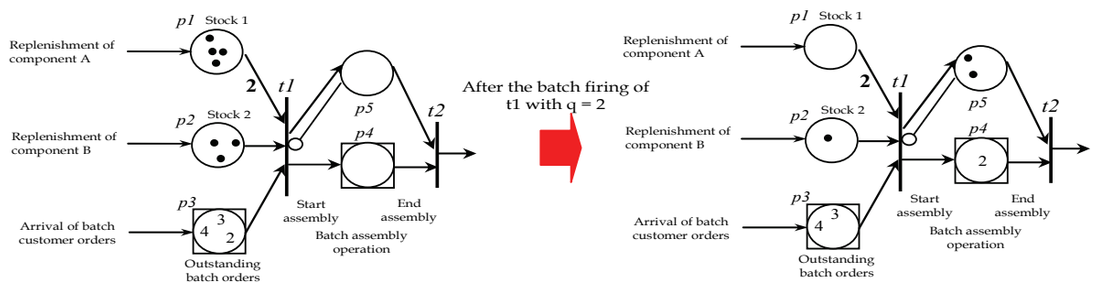

Fig. 1. An assembly-to-order system: synchronization and coordination of material and information flows (Batch firing example)

2.4 Discrete enabling and firing rules

A discrete transition t is said to be enabled at μ-marking μ (its corresponding M-marking M) if and only if:

$$\forall p \in {}^{\bullet}t, M(p) \geq w(p,t) \tag{8}$$

$$\forall p \in {}^{\circ}t, M(p) < w(p,t) \tag{9}$$

The discrete firing of t leads to a new μ-marking μ':

$$\forall p \in {}^{\bullet}t, \mu'(p) = \mu(p) - w(p,t) \tag{10}$$

$$\forall p \in t^{\bullet} \cap P_d, \mu'(p) = \mu(p) + w(t,p) \tag{11}$$

$$\forall p \in t^{\bullet} \cap P_b, \mu'(p) = \mu(p) + \{w(t,p)\} \tag{12}$$

The firing rules in this case are the same as those for a transition in a classical Petri net. For each output batch place p, after the firing of transition t, a batch token with the size equal to the weight $w(p, t)$ will be created. The discrete enabling and firing rules of a discrete transition t can be regarded as a special case of the batch enabling and firing rules of a batch transition. For $q = 1$ and ${}^{\bullet}t \cap P_b = \varnothing$ the batch firing rules are reduced to the discrete firing rules. A close look of the enabling conditions (2) and (3) finds that they are actually the *q-enabling* conditions for a standard Petri net. In other words, in a standard Petri net, a transition t is said to be q-enabled at a marking M if and only if (2) and (3) are satisfied (i.e., there is at least $q \times w(p, t)$ tokens in each input place of t and the number of tokens in each inhibitor place p does not exceed $w(p, t)$). The q-firing of a q-enabled transition t consists of firing the transition q times simultaneously.

2.5 Discrete firing example

As an example, consider an inventory control system represented in Fig. 2. The inventory control policy used in the system is a continuous review (s, S) policy specified by the immediate transition t_3. The order-up-to-level of the policy are taken as $s = 3$ and $S = 10$ respectively, and the initial μ-marking of the net is $\mu_0 = (2, 2, \varnothing)$. The model is explained in more detail in section 3 (see Fig. 8). At the current μ-marking $\mu_0 = (2, 2, \varnothing)^T$, the discrete transition t_3 is enabled since $M(p_2) = 2 < 3$. After the firing of transition t_3, a batch token with the size equal to $w(t_3, p_3) = 10 - 2 = 8$ will be created in batch place p_3 and $w(t_3, p_2) = 8$ discrete tokens will be created in discrete place p_2. Therefore, the new μ-marking of the net after the firing is: $\mu_1 = (2, 10, \{8\})^T$.

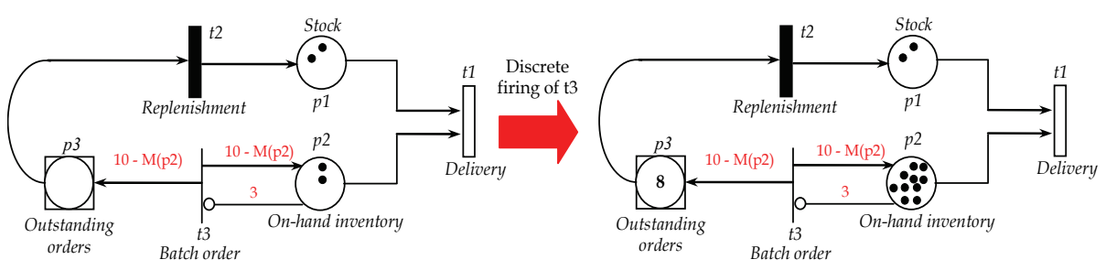

Fig. 2. An inventory control system (discrete firing example)

As shown in the two examples given in Fig. 1 and Fig 2, with their powerful graphical and mathematical formalism, BDSPNs are able to adequately describe the batch behaviour occurring at various stages of discrete event dynamic systems. Batch tokens are used to model batch quantities of material and/or information entities (batch customer orders, batch products, etc.). The sizes of the batch tokens have a quantitative meaning in defining the dynamic behaviour of the model. This is different from the concept of colors used in colored Petri nets (Jensen, 1997). The colors are rather qualitative attribute; programming languages are usually needed to define their data types and values. The BDSPN model keeps the

simplicity and the pertinence of standard discrete Petri nets while being able to model batch behaviours.

2.6 Analysis methods

As a mathematical tool, the BDSPN model has a number of properties. These properties, when interpreted in the context of the system modelled, allow the identification of the presence or absence of functional properties of the system. Besides the graphical representation, a fundamental advantage of the BDSPN is its capacity to systematically investigate many properties and characteristics of the system modelled. Specific analysis methods for BDSPN developed include: (1) the coverability (reachability) tree method, (2) the matrix algebraic approach, (3) reduction techniques, and (4) transformation techniques.
One ultimate goal for the introduction of the BDSPN model is to evaluate the performance of discrete event dynamic systems with batch behaviours, which requires a stochastic BDSPN model. It is clear that when the timing concept is considered in the model, particular policies should be specified to choose a batch token in each batch input place to fire its output transition and to resolve the conflict when multiple transitions are enabled. The temporal and the stochastic behaviour of the model are defined in our previous papers (Chen et al. 2005; Labadi et al. 2007). Similar to the existing stochastic Petri nets, the main performance analysis approach for BDSPN is based on the analysis of the stochastic marking process of the net. The approach is feasible particularly when the underlying stochastic process has a finite number of states (Labadi et al., 2007). For complex systems, simulation methods may be required (Chen et al., 2005, Amodeo et al. 2007).

2.7 Applications of the model

The BDSPN model increases the modelling and analysis power of the existing discrete Petri nets. It is able to describe essential characteristics of logistics systems (batch behaviours, randomness, operational policies, synchronization of various flows) and more generally discrete event dynamic systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. The model is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes) and makes it possible to describe more specific activities such as customer order processing, replenishment of stocks, production and delivery in a batch mode. The capability of the model to meet real needs is shown through applications dedicated to modelling and performance optimization of inventory control systems (Labadi et al., 2007) and a real-life supply chain (Chen et al., 2005; Amodeo et al., 2007).

3. Transformation into an equivalent classical Petri net

The objective of this section is to study the transformation of a BDSPN model into an equivalent classical Petri net model. Such a transformation is possible in some cases for which the corresponding transformation methods are developed. We will also show that for the model with variable arc weights depending on its marking, the transformation is impossible. This study establishes a relationship between BDSPNs and classical discrete Petri nets and demonstrates the necessity of the introduction of the BDSPN model.

3.1 Reachability graph

For the introduction of the transformation methods for the BDSPN model, we need to define two types of reachability graphs of the model. An illustration example of the reachability concept of the model is given in Fig. 3. A μ-marking reachability graph of a given BDSPN is a directed graph (V_μ, E_μ), where the set of vertices V_μ is given by the reachability set (μ_0*: all μ-markings reachable from the initial marking μ_0 by firing a sequence of transitions and the initial marking), while the set of directed arcs E_μ is given by the feasible μ-marking changes in the BDSPN due to transition firing in all reachable μ-markings. Similarly, we define M-marking reachability graph (V_M, E_M) which can be obtained from (V_μ, E_μ) by transforming each μ-marking in V_μ into its corresponding M-marking and by merging duplicated M-markings (and duplicated arcs).

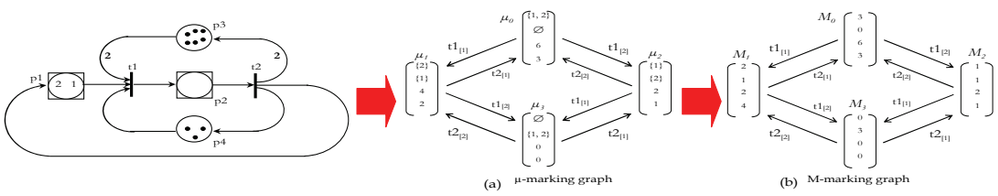

Fig. 3. An illustration of the μ-reachability and the M-reachability graphs

3.2 Special case with simple batch places

Firstly, we consider the case where all batch tokens in each batch place of the BDSPN are always identical (have the same size). A batch place p_i is said to be simple if the sizes of its all batch tokens are the same for any μ-marking reachable from μ_0.

$$p_i \in P_b \text{ is a simple batch place} \Rightarrow \forall \mu \in \mu_0^*, \forall b \in \mu(p) : b = \text{constant} \tag{13}$$

In order to facilitate the description of the transformation method for this case, we consider an illustrative example given in Fig. 4. As shown in the figure, the net (a) whose all batch places are simple can be easily transformed into an equivalent classical discrete Petri net (b). We observe that the two nets have the same M-marking reachability graph (the same dynamical behaviour). Indeed, the two properties, (*i*) all batch places of the net are simple and (*ii*) the net has no variable arc weight, lead to a constant batch firing index q_j for each batch transition $t_j \in T_b$ of the net. As formulated in the following procedure, the transformation method consists of (*i*) transforming each batch place into a discrete place and (*ii*) integrating the constant batch firing index of each batch transition in the weights of its input and output arcs in the resulting classical net in order to respect the dynamic behaviour of the original batch net.

Transformation procedure (special case)

Given a batch deterministic and stochastic Petri net BDSPN = (P, T, I, O, V, W, Π, μ_0) whose all batch places are simple and whose all arcs have a constant weight. This net can be transformed into an equivalent classical discrete Petri net, denoted by DPN = (P*, T, I, O, V, W*, Π, M_0), by the following procedure:

Step 1. The set of discrete places P_d of the BDSPN and their markings remain unchanged for the DPN.

$$\forall p_i \in P_d, M_0(p_i) = \mu_0(p_i). \tag{14}$$

Step 2. Each batch place of the BDSPN is transformed into a discrete place M-marked in the DPN.

$$\forall p_i \in P_b, M_0(p_i) = \sum_{b \in \mu(p_i)} b. \tag{15}$$

Step 3. The set of transitions T of the BDSPN remains unchanged for the *DPN*.
Step 4. The weight of each output arc of each batch place $p_i \in P_b$ of the BDSPN is set to the size of its batch tokens b_i.

$$\forall p_i \in P_b, \forall t_j \in p_i^{\bullet}, w^*(p_i, t_j) = w(p_i, t_j) \times \frac{b_i}{w(p_i, t_j)} = b_i. \tag{16}$$

Step 5. The weight of each output arc of each batch transition $t_j \in T_b$ of the BDSPN is set to its original weight multiplied by its batch firing index q_j.

$$\forall p_i \in P_b, \forall t_j \in p_i^{\bullet}, w^*(t_j, p_i) = w(t_j, p_i) \times q_j = w(t_j, p_i) \times \frac{b_i}{w(p_i, t_j)}. \tag{17}$$

Step 6. The weight of each output arc of each discrete transition $t_j \in T_d$ of the BDSPN remains unchanged for the *DPN*.

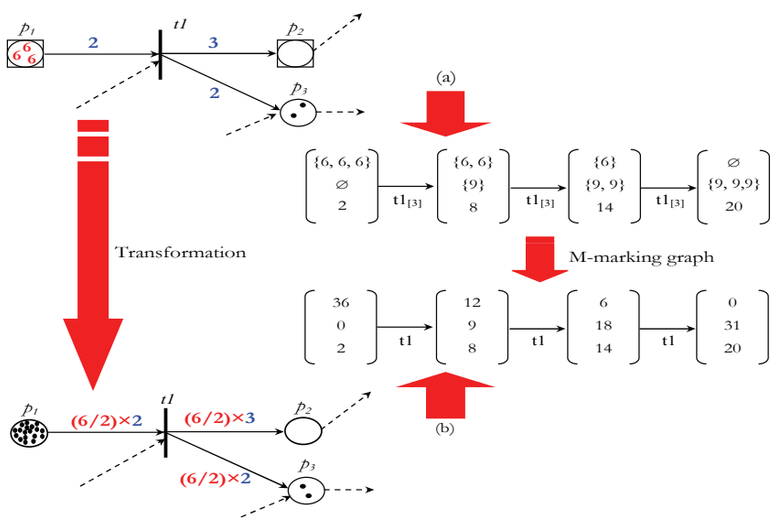

Fig. 4. Transformation of a BDSPN into an equivalent classical Petri net (special case)

3.3 General case

The proposed transformation procedure can be generalized to allow the transformation of a BDSPN containing batch places which are not simple into an equivalent classical Petri net.

The transformation is feasible if we know in advance all possible batch firings of all batch transitions and all possible batch tokens which can appear in each batch place of the net during its evolution. In other words, the transformation can be performed when we well know the dynamic behaviour of the BDSPN for its given initial μ-markings μ_0.

- Let $D(t_j)$ denote the set of all q-indexed transitions $t_{j[q]}$ generated by the firings of the batch transition tj with all possible batch firing indexes q during the evolution of the BDSPN starting from μ_0. Formally:

$$D(t_j) = \left\{ t_{j[q]} \middle| \exists \mu \in \mu_0^*, \mu\left[t_{j[q]} \rightarrow \right\}. \tag{18}$$

where μ_0 denotes the set of reachable μ-markings of the net from μ_0 and $\mu[t_{j[q]} \rightarrow$ denote that the batch transition t_j can be fired from μ with a batch firing index q. For example, the batch transition t_1 of the net in Fig. 5(a) can be fired during the evolution of the net (see the reachability graph) by two possible batch firings indexes $q = 1$ and $q = 2$, then $D(t_1) = \{t_{1[1]}, t_{1[2]}\}$.

- Let $D(p_i)$ denote the set of all possible batch tokens which can appear in the batch place p_i during the evolution the BDSPN starting from its initial μ-marking μ_0. Formally:

$$D(p_i) = \left\{ b \middle| \exists \mu \in \mu_0^*, b \in \mu(p_i) \right\}. \tag{19}$$

For example, the set of all possible batch tokens which can appear in the batch place p_1 of the BDSPN in Fig. 5(a) during its evolution starting from $\mu_0 = (\{1,2\}, \varnothing, 6, 3)^T$ is $B(p_1) = \{1, 2\}$ as shown in the reachability graph.

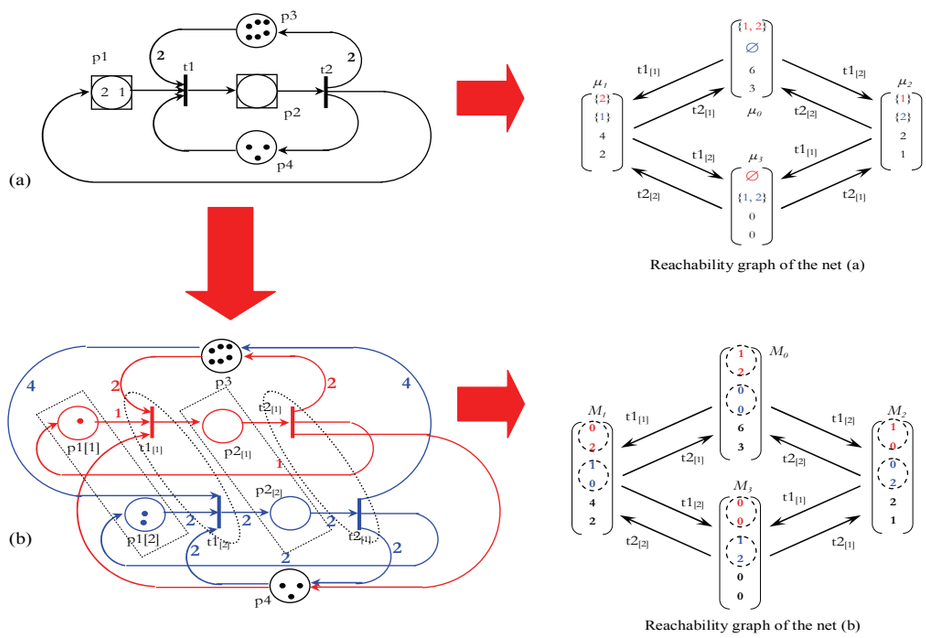

Fig. 5. Transformation of a BDSPN into an equivalent classical Petri net (general case)

By analogy with the transformation procedure for the special case, the transformation for the general case consists of the transformation of its each batch place p_i into a set of discrete places corresponding to $D(p_i)$ and the transformation of its each batch transition t_j into a set of discrete transitions corresponding to $D(t_j)$. For example, the transformation of the BDSPN into a classical Petri net given in Fig. 5 is realized by transforming the batch transition t_1 (resp. t_2) into a set of discrete transitions $\{t_{1[1]}, t_{1[2]}\}$ (resp $\{t_{2[1]}, t_{2[2]}\}$) and by transforming the batch place p_1 (resp. p_2) into a set of discrete places $\{p_{1[1]}, p_{1[2]}\}$ (resp. $\{p_{2[1]}, p_{2[2]}\}$ as shown in Fig. 5(b). Similar to the special case, to respect the dynamical behaviour of the BDSPN, each possible batch firing index of each batch transition is integrated in the weights of the input and output arcs of the corresponding transition in the resulting classical net. After a close look of the reachability graphs of the two nets, we find that the two nets have the same behaviour. As illustrated in the figure, each μ-marking μ_i of the BDSPN corresponds to the marking M_i of the resulting classical Petri net. The M-marking of each batch place p_i is expressed by its corresponding set of discrete places $D(p_i)$. The transformation procedure for the general case is outlined in the following.

Transformation procedure (general case)

Step 1. The set of discrete places P_d of the BDSPN and their markings remain unchanged for the *DPN*.

$$p_i \in P_d, M_0(p_i) = \mu_0(p_i). \tag{20}$$

Step 2. Each batch place p_i of the BDSPN is converted into a set of discrete places $D(p_i)$ in the DPN such as:

$$D(p_i) = \left\{ p_{i[b]} \middle| b \in D(p_i) \right\} \text{and } \forall p_{i[b]} \in D(p_i), M_0(p_{i[b]}) = \sum_{l \in \mu(p_i) \text{ and } l=b} l. \tag{21}$$

Step 3. Each batch transition t_j of the BDSPN is converted into a set of discrete transitions $D(t_j)$ in the DPN such that:

$$D(t_j) = \left\{ t_{j[q]} \middle| t_{j[q]} \in D(t_j) \right\}. \tag{22}$$

The set of discrete transitions T_b of the BDSPN remains unchanged for the *DPN*.

Step 4. Each place $p_{i[b]} \in D(p_i)$ is connected to the output transitions $(p_{i[b]})^\bullet$ such that:

$$\forall p_{i[b]} \in D(p_i), (p_{i[b]})^\bullet = \left\{ t_{j[q]} \middle| t_j \in p_i^\bullet \text{ and } q = b / w(p_i, t_j) \right\}. \tag{23}$$

$$\forall p_{i[b]} \in D(p_i), \forall t_{j[q]} \in (p_{i[b]})^\bullet, w(p_{i[b]}, t_{j[q]}) = w(p_i, t_j) \times b. \tag{24}$$

Step 5. Each transition $t_{j[q]} \in D(t_j)$ is connected to the output places $(t_{j[q]})^\bullet$ such that:

$$\forall t_{j[q]} \in D(t_j), (t_{j[q]})^{\bullet} = \left\{ p_{i[b]} \middle| (p_{i[b]} \in D(p_i)), (p_i \in t_j^{\bullet} \cap P_d) \text{ and } (q = b / w(p_i, t_j)) \right\} \cup \left\{ p_i \middle| p_i \in t_j^{\bullet} \cap P_d \right\}. \tag{25}$$

Clearly for each $t_{j[q]}$, its output places will be all discrete output places of transition t_j and all places generated by the batch output places of transition t_j. The weights of the corresponding arcs are given by:

$$\forall t_{j[q]} \in D(t_j), \forall (p_i \vee p_{i[b]}) \in (t_{j[q]})^{\bullet}, w(t_{j[q]}, p_{i[b]}) = q \times w(t_j, p_i). \tag{26}$$

Step 6. Each place $p_{i[b]} \in D(p_i)$ is connected to the input transitions ${}^{\bullet}(p_{i[b]})$ such that:

$$\forall p_{i[b]} \in D(p_i), {}^{\bullet}(p_{i[b]}) = \left\{ t_{j[q]} \middle| t_j \in {}^{\bullet} p_i \text{ and } q = b / w(t_j, p_i) \right\} \cup \left\{ t_j \in ({}^{\bullet} p_i \cap P_d) \right\}. \tag{27}$$

Clearly for each $p_{i[b]}$, its input transitions will be all discrete input transitions of transition t_j and all transitions generated by the output batch transitions of transition t_j. The weights of the corresponding arcs are given by:

$$\forall p_{i[b]} \in D(p_i), \forall (t_j, t_{j[q]}) \in {}^{\bullet}(p_{i[b]}), w(t_{j[q]}, p_{i[b]}) = q \times w(t_j, p_i). \tag{28}$$

Step 7. Each transition $t_{j[q]} \in D(t_j)$ will be connected to the set ${}^{\bullet}(t_{j[q]})$ of input places such that:

$$\forall t_{j[q]} \in D(t_j), {}^{\bullet}(t_{j[q]}) = \left\{ p_{i[b]} \middle| (p_i \in {}^{\bullet} t_j \cap P_d) \text{ and } (q = b / w(p_i, t_j)) \right\} \cup \left\{ p_i \middle| p_i \in {}^{\bullet} t_j \cap P_d \right\}. \tag{29}$$

Clearly for transition $t_{j[q]}$, its input places will be all discrete input places of transition t_j and all places generated by the output batch places of transition t_j. The weights of the corresponding arcs are given by:

$$\forall t_{j[q]} \in D(t_j), \forall (p_i \vee p_{i[b]}) \in {}^{\bullet}(t_{j[q]}), w(p_{i[b]}, t_{j[q]}) = q \times w(p_i, t_{j[q]}). \tag{30}$$

Step 8. The arcs which connect discrete places with discrete transitions in the BDSPN and their weights remain unchanged in the *DPN*.

3.3 Case with inhibitor arcs

The transformation is also possible for BDSPNs with inhibitor arcs whose weights are constant. We will illustrate it by using some examples.

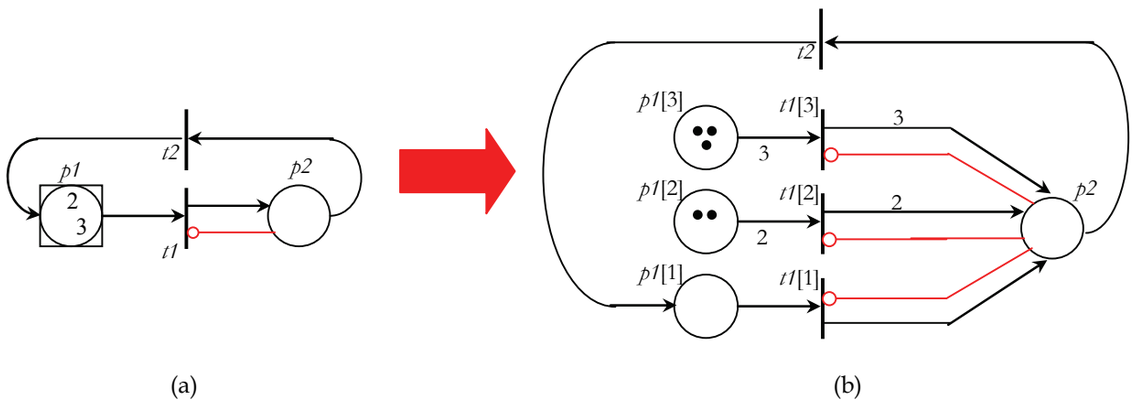

Fig. 6. Transformation of a BDSPN with inhibitor arc connecting a discrete place to a batch transition

Sub-case 1: Inhibitor arc connecting a discrete place to a batch transition.
As shown in the net depicted in Fig. 6(a), in the case where there is an inhibitor arc connecting a discrete place p_i to a batch transition t_j, the corresponding inhibitor condition must be reproduced in the resulting classical Petri net for all q-indexed transitions $t_{j[q]}$ generated by the batch transition t_j. Clearly, in this example, the batch transition t_1 can be fired with three possible batch firing indexes during the evolution of the net. In other words, the transition t_1 generates three possible q-indexed transitions $t_{1[1]}$, $t_{1[2]}$, $t_{1[3]}$. Thus, in the corresponding classical Petri net there are three inhibitor arcs which connect the discrete place p_2 to the three q-indexed transitions, respectively. The reason for the duplication of the inhibitor arc is obvious: Firing the transition t_1 with one of the three possible batch firing indexes must respect the enabling condition $M(p_2) < w(t_1, p_2)$ in the net (a). In the classical net, the condition is expressed as $M(p_2) < w(t_{1[q]}, p_2)$ for each q-indexed transition generated by t_1. It is easily to observe that the two nets are identical in terms of their dynamical behaviours.

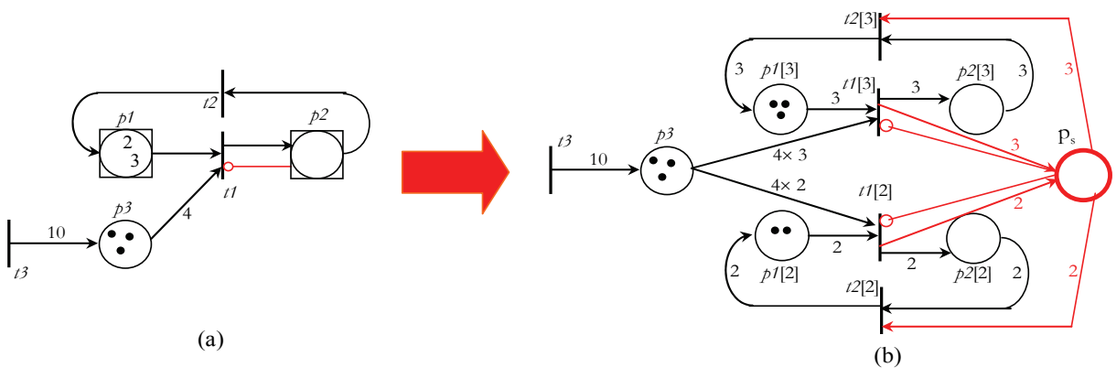

Fig. 7. Transformation of a BDSPN with inhibitor arc connecting a batch place to a transition

Sub-case 2: Inhibitor arc connecting a batch place to a transition
We now consider the case as shown in Fig. 7(a) where there is an inhibitor arc connecting a

batch place to a transition. The enabling of the transition t_1 for a given batch firing index q in the net (a) must satisfy the condition $M(p_2) < w(t_1, p_2)$ imposed by the inhibitor arc. After the transformation of each batch place (resp. batch transition) into a set of discrete places (resp. a set of transitions), we observe that to respect the enabling condition imposed by the inhibitor arc in the net (a), it is necessary to capture the total marking of the discrete places generated by the batch place p_2 by using a supplementary place p_s in the classical Petri net.

3.4 Case of the timed model

The transformation techniques discussed so far do not consider temporal and/or stochastic elements in a BDSPN, but they can be adapted for timed and stochastic BDSPN models. The basic idea is as follows: Each discrete transition in the BDSPN model keeps its nature (immediate, deterministic, stochastic) in the resulting classical Petri net. The q-indexed transition $t_{j[q]}$ which may be generated by each batch transition t_j has the same nature as the transition t_j. Other elements of the BDSPN model may also be taken into account in the resulting classical model such as the execution policies; the firing priorities of some transitions; etc.

3.5 Necessity of the BDSPN model

In this section, the necessity of the introduction of the BDSPN model is demonstrated through an analysis of the transformation procedures presented in the previous section. The advantages of the model are discussed in two cases: the case where a BDSPN can be transformed into a classical Petri net and the case where the transformation is impossible.

Case 1 - the BDSPN model is transformable

In the case where the transformation is possible, the advantages of the BDSPN model are outlined in the following:

- As shown in the transformation procedures developed in this section, we note that the resulting classical Petri net depends on the initial μ-marking of the BDSPN. Obviously, if we change the initial μ-marking of the BDSPN given in Fig. 5(a), we will obtain another classical Petri net. For example, if there is another batch token of different size in the batch place p_1, all the structure of the corresponding classical Petri net must be changed. In fact, the batch places of the BDSPN may not generate the same set of q-indexed transitions $D(t_j)$ for each batch transition t_j and may not generate the same set of discrete places $D(p_i)$ for each batch place p_i during the evolution of the net.
- The transformation of a given BDSPN model into an equivalent classical Petri net may lead to a very large and complex structure. According to the transformation procedure developed in subsection 3.2, the number of places $|P^*|$ and the number of transitions $|T^*|$ in the equivalent classical Petri net are given by:

$$\left|P^*\right| = \left|P_d\right| + \sum_{i=1}^{|P_b|} \left|D(p_i)\right| \tag{31}$$

$$\left|T^*\right| = \left|T_d\right| + \sum_{j=1}^{|T_b|} \left|D(t_j)\right| \tag{32}$$

where $|P_b|$ is the number of the batch places; $|P_d|$ is the number of the discrete places; $|T_b|$ is the number of the batch transitions; $|T_d|$ is the number of the discrete transitions of the given BDSPN. $D(t_j)$ is the set of q-indexed transitions generated by each batch transition $t_j \in T_b$ and $D(p_i)$ is the set of all possible batch tokens which appear in each batch place $p_i \in P_b$ during the evolution of the BDSPN.

As an example, if a BDSPN contains 10 batch places, 10 batch transitions, and its evolution leads to 10 different possible batch tokens in each batch place, and 10 different batch firing indexes for each batch transition, then we need at least 100 places and 100 transitions to construct an equivalent classical Petri net. Note that the number of arcs to be introduced in the classical Petri net is also very large. We see a much higher complexity of the resulting Petri net compared to the original BDSPN.

Case 2 - the BDSPN is not transformable

The modelling of some discrete event systems such as inventory control systems and logistic systems, as shown in (Labadi et al. 2007,2005), require the use of the BDSPN model with variables arc weights depending on its M-marking and possibly on some decision parameters of the systems. It is the case of the BDSPN model of an inventory control system whose inventory replenishment decision is based on the inventory position of the stock considered and the reorder and order-up-to-level parameters (see Fig. 8). The modelling of such a system is possible by using a BDSPN model with variables arc weights depending on its M-marking. This kind of BDSPN model is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes).

The BDSPN model shown in Fig. 8 represents an inventory control system where its operations are modelled by using a set of transitions: generation of replenishment orders (t_3); inventory replenishment (t_2); and order delivery (t_1) that are performed in a batch way because of the batch nature of customer orders represented by batch tokens in batch place p_4 and the batch nature of the outstanding orders represented by batch tokens in batch place p_3. In the model, the weights of the arcs (t_3, p_2), (t_3, p_3) are variable and depend on the parameters s and S of the system and on the M-marking of the model ($S - M(p_2) + M(p_4)$; $s + M(p_4)$). The model may be built for the optimization of the parameters s and S. In this case, the techniques for the transformation of the BDSPN model into an equivalent classical Petri net model proposed in the previous section is not applicable.

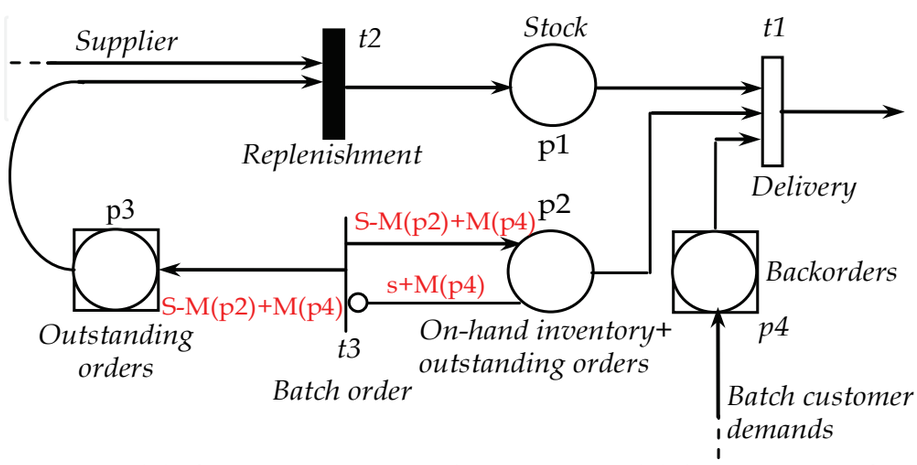

Fig. 8. BDSPN model of an inventory control system: a model that is not transformable

In fact, contrary to the example given in Fig. 5, in this model, the sizes of the batch tokens that may be generated depend on both the initial μ-marking of the model and the parameters *s* and *S*. In other words, a change of the decision parameters *s* and *S* of the system or the initial μ-marking of the model will lead to another way of the evolution of the discrete quantities (variable batches of different sizes). Moreover, the appearance of stochastic transitions in the model makes more difficult to characterize all possible sizes of the batch tokens that are necessary to be known for the application of the transformation methods.

4. Transformation into an associated discrete Petri net

In this section, the associated discrete Petri net of a BDSPN is defined and the relationships between the two models are then explored. Because the objective is to develop structural (qualitative) analysis methods for the BDSPN model, we confine ourselves hereafter to the untimed version of BDSPN model obtained by removing the firing times of all transitions from the model. Before giving a formal definition of the associated discrete Petri net, we recall the state equation and the q-firing of a transition of the model.

4.1 Incidence matrix and state equation

Suppose that the current μ-marking of a BDSPN is μ_k and transition t_j is enabled at the μ-marking. The firing of the transition will result in a new marking μ_{k+1} written as:

$$\forall p \in {}^{\bullet}t_j, \mu_{k+1}(p) = \mu_k(p) - q_j^k \times w(p, t_j) \tag{33}$$

$$\forall p \in t_j^{\bullet}, \mu_{k+1}(p) = \mu_k(p) + q_j^k \times w(t_j, p) \tag{34}$$

$$\forall p \notin \left({}^{\bullet}t_j \cup t_j^{\bullet}\right), \mu_{k+1}(p) = \mu_k(p) \tag{35}$$

where q_j^k is the batch firing index of the transition t_j at the μ-marking μ_k such that:

$$q_j^k = \begin{cases} 1 & \text{if } {}^{\bullet}t_j \cap P_b \neq \varnothing \\ b / w(p,t) \Big| \quad b \in \mu_k(p) & \text{if } {}^{\bullet}t_j \cap P_b \neq \varnothing \end{cases} \tag{36}$$

Similar to classical Petri nets, the incidence matrix *W* of a BDSPN with *m* places and *n* transitions is an $n \times m$ matrix $W = [w_{t,p}]$ whose entries are defined as:

$$w_{t,p} = w(t,p) - w(p,t) \tag{37}$$

where *w(t, p)* is the weight of the arc from transition *t* to its output place *p* and *w(p, t)* is the

weight of the arc to transition t from its input place p. The equations (33), (34), and (35) can then be written in matrix form (38), called state equation, as:

$$\mu_{k+1} = \mu_k + W \times Q \tag{38}$$

where μ_{k+1} is the μ-marking obtained (reached) after the firing of transition t_j from μ-marking μ_k. μ_{k+1} and μ_k are both $m \times 1$ column vector. The entry $\mu_{k+1}(p)$ is the number of discrete tokens if p is a discrete place or the multiset of batch tokens (positive integers) is p is batch place. Q is an $n \times 1$ column vector with $(n - 1)$ zero entries and one nonzero entry. A nonzero entry q_j^k in the j^{th} position indicates that transition t_j is fired at the k^{th} firing. Q is called firing count vector defined as:
$Q = [q_j^k]_{n\times 1}$ such as :

$$q_j^k = \begin{cases} 1 & \text{if } \left(^\bullet t_j \cap P_b\right) \neq \varnothing \\ b / w(p,t) & b \in \mu_k(p) \text{ if } \left(^\bullet t_j \cap P_b\right) \neq \varnothing \end{cases} \tag{39}$$

4.2 The q-firing of a transition

The *q-firing* of a q-enabled transition t consists of firing the transition q times simultaneously. In a classical Petri net, a transition t_j is said to be *q-enabled* at a marking M if and only if:

$$\forall p_i \in {}^\bullet t_j, M(p_i) \geq q \times w(p_i, t_j) \tag{40}$$

$$\forall p_i \in {}^\circ t_j, M(p_i) < w(p_i, t_j) \tag{41}$$

From the above definition, a transition t_j is q-enabled if and only if there is at least $q \times w(p_i, t_j)$ tokens in each input place of t_j and the number of tokens in each inhibitor place p_i does not exceed $w(p_i, t_j)$. The q-firing of a q-enabled transition t_j consists of a sequential firing of t_j where t_j is fired q times.
The matrix equation which characterizes a q-firing of transition t_j is as follows:

$$M_{k+1} = M_k + W \times U \times q \tag{42}$$

where $U[i] = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$

We can also write the equation (42) as:

$$M_{k+1} = M_k + W \times Q \tag{43}$$

where $Q[i] = \begin{cases} 0 & \text{if } i \neq j \\ q & \text{if } i = j \end{cases}$

Note that for a BDSPN the equation (43) is similar to the state equation (38) if we consider the M-marking. This leads us to a certain dynamic analogy between a BDSPN and its associated discrete Petri net.

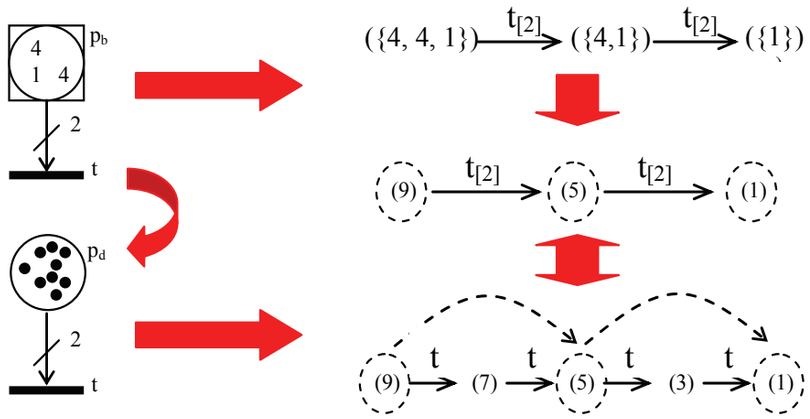

Fig. 9. An illustration of the q-firing concept in a BDSPN and a classical Petri net

The basic idea of the analogy is illustrated by an example given in Fig. 9, where the M-marking graph of a BDSPN is presented in an unfolded form in the marking graph of its associated discrete Petri net. As shown in the previous section, the discrete enabling and firing rules of a discrete transition *t* can be regarded as a special case of the batch enabling and firing rules of a batch transition. For *q = 1*, the batch rules are reduced to the discrete rules. A close look of the enabling conditions (2) and (3) finds that they are actually the q-enabling conditions for a standard Petri net. In other words, in a standard Petri net, a transition *t* is said to be q-enabled at a marking M if and only if (2) and (3) are satisfied (i.e., there is at least $q \times w(p, t)$ tokens in each input place of t and the number of tokens in each inhibitor place p does not exceed $w(p, t)$).

4.3 Associated discrete Petri net

Any BDSPN can be associated with a discrete Petri net, denoted by *DPN*, which is obtained by:

- Transforming each batch place into a discrete place with the initial number of tokens in the discrete place equal to the initial total size of the batch tokens in the batch place, i.e.,

$$\forall p_i \in P_b, M_0(p_i) = \sum_{\mu_p \in \mu_0(p_i)} \mu_p$$

- Keeping all its transitions and arcs unchanged in the associated discrete Petri net.

An illustrative example of the transformation of a BDSPN into its associated discrete Petri net is given in Fig. 10, where an order treatment system is modelled by both the BDSPN model (a) and its associated classical discrete Petri net (b). Each transition t_j (t_1 and t_2), in the two Petri nets, represents the same operation. Contrary to those in the classical model (b), the places p_2 and p_3 in the BDSPN are batch places with batch tokens.

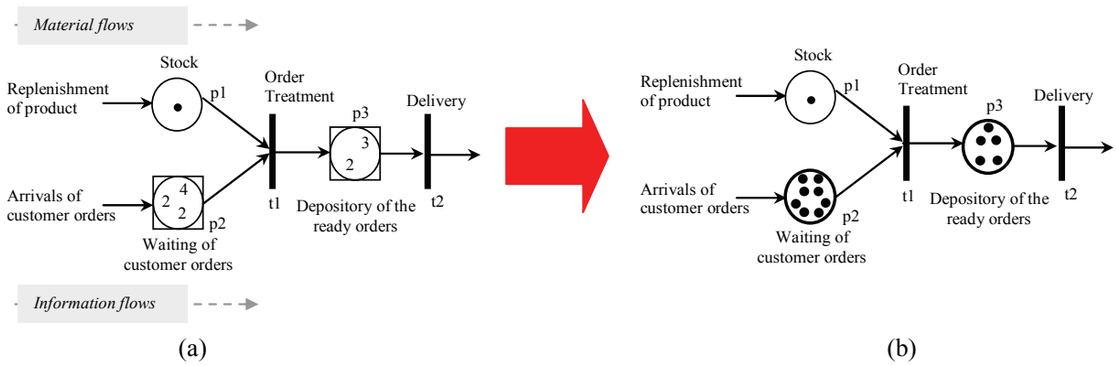

Fig. 10. Synchronization and coordination of material and information flows

In the BDSPN model (a), customer orders with different sizes arrive and are recorded in the batch place p_2 where they wait for treatment according to their dates of arrival. In the example, three customer orders currently arrived in the batch place p_2: two customer orders with size 2 and one customer order with size 4 (i.e., $\mu(p_2) = \{2, 4, 2\}$). The customer orders will be filled, according to their orders of arrival in batch place p_2, from the stock of finished products, represented by the discrete place p_1, if the stock is sufficient. At the current μ-marking of the BDSPN, there is only one unit of finished product available in the stock (i.e., $\mu(p_1) = 1$). The transition t_2 is not enabled and the customer orders must keep waiting until stock is sufficient to fill any of the orders. Contrary to the behaviour of the BDSPN, in the classical model (b), the orders are recorded in the discrete place p_2. In this representation, there are no informations about the various sizes of the customer orders, except for their total size $M(p_2) = 8$ which corresponds to the M-marking of the place p_2 in the BDSPN model (sum of the sizes of its batch tokens). In this case the transition t_2 is enabled at the current marking of the net. The place p_3 in the two models corresponds to the customer orders ready for delivery. The delivery is carried out in batch mode in the BDSPN by firing the transition t_2 and each customer will receive his/her order as it is ordered. This is not the case in the discrete Petri net model where the delivery is carried out by multiple firings of the discrete transition t_2.

This illustration clearly shows that the BDSPN and its associated discrete Petri nets behave differently. However, despite this difference, the BDSPN and its associated discrete Petri net have several common qualitative properties. The rest of this section is thus dedicated to establish formal relationships between the two models.

4.4 Relationships between a BDSPN and its associated discrete Petri net

Before conducting a formal analysis of a BDSPN and its associated discrete Petri net to explore the relationships, we first examine an illustrative example given in Fig. 11.

Important observations can be obtained by a careful investigation of the two marking graphs in the figure. The M-marking graph of the BDSPN is reproduced in an unfolded form in the marking graph of its associated discrete Petri net. In fact, each one-step state transition $M_k[t_{j[q]} \rightarrow M_{k+1}$ appeared in the M-marking graph of the BDSPN has its corresponding multi-step state transition presented as $M_k[t_j\ t_j \ldots.\ t_j \rightarrow M_{k+1}$ in the marking graph of its associated discrete Petri net. For instance, the batch firing of the transition t_1 with batch firing index $q = 2$ (i.e, firing of $t_{1[2]}$) at the M-marking M_0 which leads to a new M-marking M_2 (i.e., $M_0[t_{1[2]} \rightarrow M_1$) in the M-marking graph of the BDSPN has its corresponding multi-step transition firing presented as $M_0[t_1t_1 \rightarrow M_1$ in the marking graph of the discrete Petri net. In other words, each batch firing of a batch transition t_j with a batch firing index q at a given M-marking M_k can be interpreted as q discrete firings occurred concurrently in the associated discrete Petri net. Consequently, the set of reachable M-marking of the BDSPN is included in the set of reachable marking of its associated discrete Petri net.

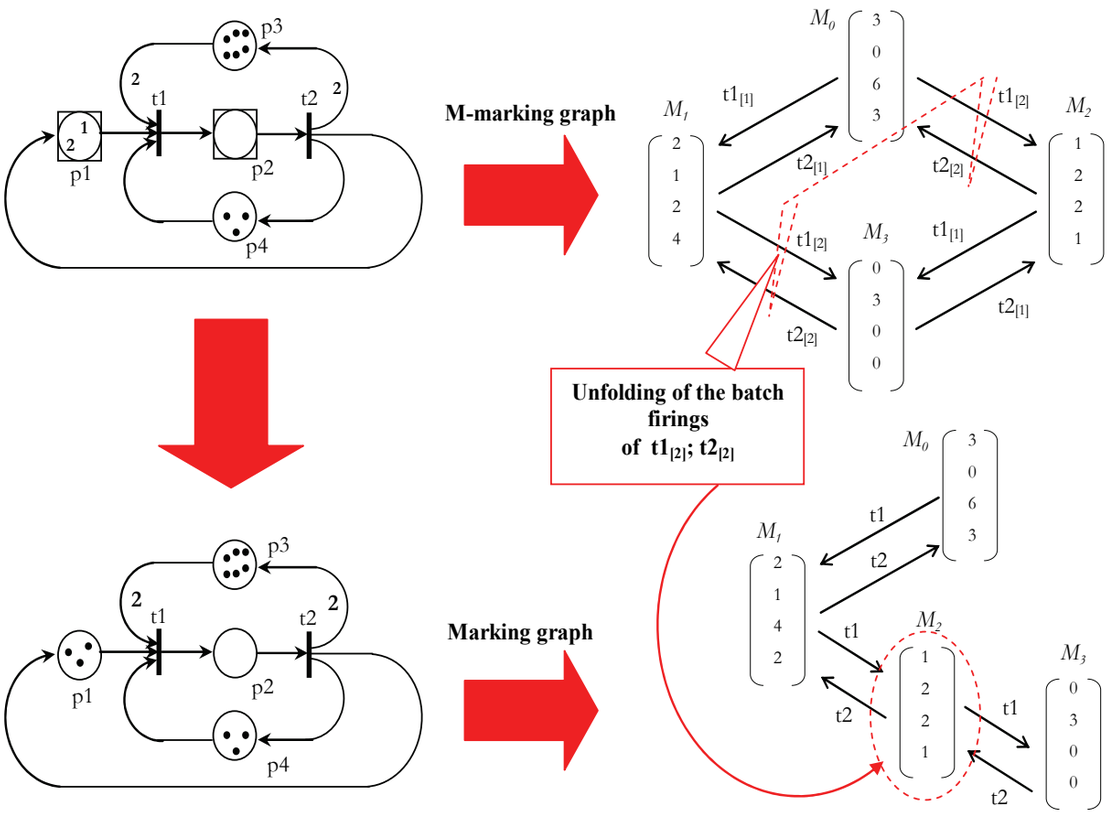

Fig. 11. An illustration of the behaviours of a BDSPN and its associated discrete Petri net

As shown in Fig. 12, due to the unfolding operation of batch firings, the associated discrete Petri net may generate some intermediate states (markings) between two states (markings) in the BDSPN model. In general case, the intermediate states may be evaluated to other states which do not appear in the BDSPN model.

As shown by an example given in Fig. 13, when the BDSPN contains an inhibitor arc, some of its reachable M-markings are not reproduced by its associated discrete Petri net. It is

simply due to the different effect of the inhibitor arc on the behaviours of the two nets. Contrary to the BDSPN, in the associated discrete Petri net, the inhibitor arc prevents the occurrence of the sequence of firings of the transition t_1 which corresponds to the batch firings of $t_{1[3]}$ and $t_{1[2]}$ in the BDSPN.

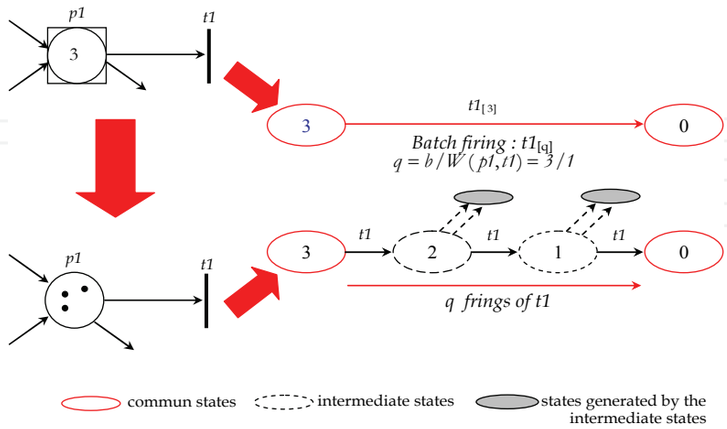

Fig. 12. Common and intermediate states of a BDSPN and its associated discrete Petri net

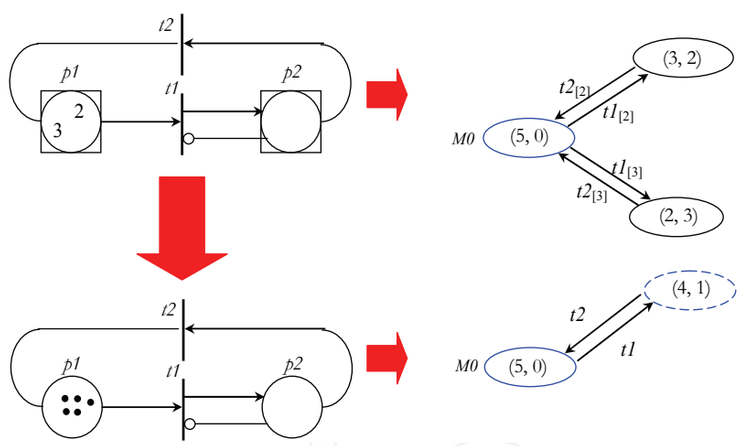

Fig. 13. Illustration of the case of a BDSPN with inhibitor arcs.

The above discussion leads to some important properties about the behaviours of the BDSPN and its associated discrete Petri. In the following, it is assumed that the BDSPN model has no inhibitor arc. We use G_μ(BDSPN, μ_0), G_M(BDSPN, M_0) and G_M(DPN, M_0) to denote the μ-marking reachability graph of the BDSPN, the M-marking reachability graph of the BDSPN, and the reachability graph of the associated discrete Petri net *DPN*, respectively.

- ***Property 1.*** A BDSPN and its associated discrete Petri net have the same initial M-marking M_0 and the same incidence matrix W.

Proof: This property is a direct consequence of the definition of the associated discrete Petri net.

- ***Property 2.*** The set of the reachable M-markings of a BDSPN, M^*_{BDSPN}, is included in the set of the reachable markings, M^*_{DPN}, of its associated discrete Petri net, i.e.,

$$M^*_{BDSPN} \subseteq M^*_{DPN} \Leftrightarrow \forall M \in M^*_{BDSPN} \Rightarrow M \in M^*_{DPN}$$

Proof: The property 1 implies that: $\forall M_0 \in M^*_{BDSPN} \Rightarrow M_0 \in M^*_{DPN}$

Let t_j be a transition enabled at the initial μ-marking μ_0 (its corresponding M-marking M_0) in the BDSPN. The transition is also enabled at the marking M_0 in the associated discrete Petri net.

i) If t_j is a discrete transition, after the firing of the transition t_j in the two nets, their M-marking (marking) will be changed to the same marking M_1. Thus, the marking M_1 appears in the two reachability graphs $G_M(BDSPN, M_0)$ and $G_M(DPN, M_0)$. We then have:

$$M_1 \in M^*_{BDSPN} \Rightarrow M_1 \in M^*_{DPN}$$

ii) If t_j is a batch transition in the BDSPN, then it can be fired at M_0 with a batch firing index q. This batch firing corresponds to q consecutive firings of the transition in the associated discrete Petri net as represented by:

$$M_0 \lfloor t_{j[q]} \rightarrow M_k \Leftrightarrow M_0 \lfloor \underbrace{t_j t_j t_j ... t_j}_{q \text{ times}} \rightarrow M_k$$

We then have: $M_k \in M^*_{BDSPN} \Rightarrow M_k \in M^*_{DPN}$

Continuing this process, for each new reachable marking M_k in the BDSPN, it is reachable in the associated discrete Petri net.

- ***Property 3.*** All feasible sequences of firings (multiple transitions which can be fired consecutively) in a BDSPN are also feasible in its associated discrete Petri net, i.e.,

$$\forall S \in G_\mu(BDSPN, \mu_0) \Rightarrow S \in G_M(DPN, \mu_0)$$

Proof: This property is an implication of the property 2 which says that all reachable M-markings of the BDSPN are included in the set of reachable markings of its associated discrete Petri net. For example, consider $S = t_{3[2]}\ t1\ t_{2[4]}$ as a feasible sequence by a given BDSPN from M to M'.

$$M[t_{3[2]}t_1t_{2[4]} \rightarrow M' \in G_\mu(BDSPN, \mu_0) \Rightarrow M[t_3t_3t_1t_2t_2t_2t_2 \rightarrow M' \in G_M(DPN, M_0)$$

- ***Property 4.*** If the associated discrete Petri net of a BDSPN is bounded then the BDSPN is also bounded (*DPN is bounded* $\Rightarrow$ *BDPN is bounded*).

Proof: By definition, a DPN is bounded if all its places are bounded. Formally:

$$\forall p \in P, \forall M \in G_M(DPN, M_0) \Rightarrow M(p) \le k,\ (k \in \text{IN})$$

However, all the M-markings of the BDSPN are included in the set of reachable markings of its associated discrete Petri net, then: $\forall p \in P, \forall M \in G_M(BDSPN, M_0) \Rightarrow M(p) \le k$. Hence the BDSPN is bounded.

- ***Property 5.*** The boundness of a BDSPN does not imply the boundness of its associated discrete Petri net.

Proof. As illustrated in Fig. 13, the associated discrete Petri net of a BDSPN may generate some intermediate states (markings) which do not appear in the BDSPN. These intermediate markings, which are not included in the states of the BDSPN, may be unbounded.

- ***Property 6.*** All P-invariants (resp. T-invariants) of the associated discrete Petri net of a BDSPN are also P-invariants (resp. T-invariants) of the BDSPN.

Proof. By definition, every P-invariant (resp. T-invariant) of a net is a solution of the equation $Y^T \times W = 0$ (resp. $W \times X = 0$), where W is the incidence matrix of the net. Since the BDSPN and its associated discrete Petri net have the same incidence matrix W, they have the same P-invariants and T-invariants.

- ***Property 7.*** The liveness (resp. the reversibility) of a BDSPN can not be concluded from the liveness (resp. the reversibility) of its associated discrete Petri net.

Proof. As we know, intermediate markings may appear in the reachability set of the associated Petri net. At these intermediate markings, transitions may be fired, leading to other markings (see Fig. 13). For the reversibility of the associated discrete Petri net, it is possible that the return to the initial marking starting from a given marking is made through a sequence of transitions which is not feasible in the BDSPN. Moreover, because of possible transition firings at the intermediate states, the liveness of the associated discrete Petri net does not imply the liveness of the BDSPN due to the existence of some sequences of transitions which are not feasible in the BDSPN.

- ***Property 8.*** A BDSPN is reversible if its associated discrete Petri net is reversible. In this case, all reachable markings of the two nets are identical.

Proof. By definition, the associated discrete Petri net is reversible if for every reachable marking M there exists a sequence of transitions whose firing reproduces M_0. That is,

$$\forall M \in M^*_{DPN}, M_0 = M \times W$$

Since the BDSPN and its associated discrete Petri net have the same incidence matrix, they have the same set of reachable markings. Hence, the BDSPN is reversible.

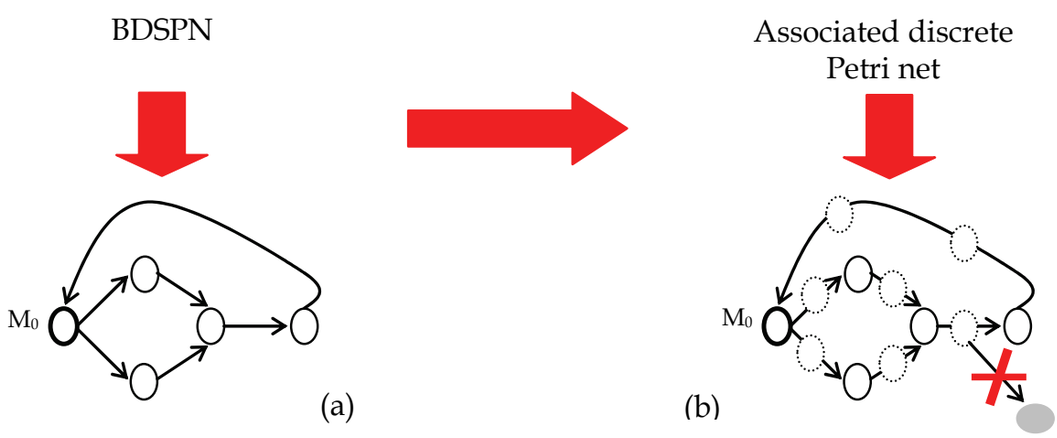

Fig. 14. Illustration of the ideal configuration between a BDSPN and its associated discrete Petri net

From the above discussion, we know that although a BDSPN and its associated discrete Petri net behave differently, formal relationships between the two models and their common qualitative properties can be explored. More results can be derived from the properties presented above. Intuitively, a case with common qualitative properties (liveness, reversibility, boundness, …) of the BDSPN and its associated discrete Petri net is illustrated in Fig. 14.

5. Conclusion

In the first part of this chapter, a new class of Petri nets, called batch deterministic and stochastic Petri nets (BDSPN), is presented as a powerful modelling and performance

evaluation tool for discrete event dynamic systems with batch behaviours. The model enhances the modelling and analysis power of the existing discrete Petri nets. It is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes) and is capable of describing activities such as customer order processing, stock replenishment, production and delivery in a batch mode. The model and its associated analysis methods are particularly suitable for the modelling and analysis of industrial and manufacturing systems where materials are processed in batches and operations are performed in batch modes to take advantages of the economies of scale or because of the batch nature of customer orders. As demonstrated in our previous applications (Amodeo et al., 2007; Labadi et al., 2007, Chen et al., 2005), the model is able to describe essential characteristics of logistics and inventory control systems (batch behaviours, batch operational policies, synchronization of various flows, randomness) and more generally discrete event dynamic systems with batch behaviours. The second part of this chapter contributes to the structural and behavioral analysis of the model by using the transformation approach. The first method proposed transforms a BDSPN into an equivalent classical discrete Petri net under some conditions. In this case, the corresponding transformation procedures are presented. For other cases, especially for BDSPNs with variable arc weights depending on their marking, the transformation is impossible. The second method analyzes a BDSPN based on its associated discrete Petri net which is obtained by converting the batch components (batch places, batch tokens, batch transitions) of the BDSPN into discrete components of the discrete Petri net. We show that although a BDSPN and its associated discrete Petri net behave differently, they have several common qualitative properties. This study establishes a relationship between BDSPNs and classical discrete Petri nets and demonstrates the necessity of the introduction of the BDSPN model.
In the future, we intend to develop new efficient methods for analysis of the BDSPN model based on the obtained formal relationships between the BDSPNs and classical discrete Petri nets and by exploring existing results for classical Petri nets. With the formal relations obtained in this work, we think that some results of classical Petri nets can be adapted and extended for the BDSPN model. On the other hand, we want to develop reduction rules for the BDSPNs following the methodology of the transformation approach. In this case, the BDSPN model will not be converted into a classical Petri net as shown in this chapter, but the size of the model will be reduced. The reduced model can be used for deriving qualitative and/or quantitative proprieties of the original BDSPN model and the analysis of the original model is thus simplified. This expected research aims at the development of an automated toolset which can help us to efficiently model and analyze untimed, timed and stochastic discrete systems with batch behaviours.

6. References

Ajmone Marsan, M, Balbo, G., Conte, G., Donatelli, S. & Franceschinis, G. (1995). Modelling with Generalized Stochastic Petri Nets, *John Wiley and Sons, ISBN: 0-471-93059-8, 1995.*

Amodeo, L., Chen, H. & El Hadji, A. (2007). Multiobjective Supply Chain Optimization: An Industrial Case Study, *Lecture Notes in Computer Science, Applications of Evolutinary Computing, ISBN: 978-3-540-71804-8, Vol. 4448, pp. 732–741, Springer Berlin Heidelberg, 2007.*

Chen, H., Amodeo, L. & Boudjeloud, L. (2003). Supply chain optimization with Petri Nets and genetic algorithms, *Proceedings of IEEE International Conference on Industrial Engineering and Production Management, ISBN: 2-930294-13-02, vol. 2, pp. 49-58, Proceedings FUCAM Editors, Porto, Portugal, May 2003.*

Chen, H., Amodeo, L., Chu, F., and Labadi, K. (2005). Modelling and performance evaluation of supply using batch deterministic and stochastic Petri nets, *IEEE transactions on Automation Science and Engineering, ISSN: 1545-5955,Vol.2, N°2, pp. 132-144, April 2005*.

Ehrig, H., Engels, G., Kreowski, H.-J. & Rozenberg G., (editors). Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools. *World Scientific, ISBN 981-02-4020-1, 1999.*

Haddad, S., "A reduction theory for coulored Nets", European Workshop on applications and theory in Petri Nets", *Lecture Notes in Computer Science, Advances in Petri Nets 1989, ISBN 978-3-540-52494-6, pp. 209-235, Springer Berlin / Heidelberg, 1989.*

Jensen, K. (1997). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Vol. I, Basic Concepts. Monographs in Theoretical Computer Science, *Springer-Verlag, 2nd corrected printing, ISBN: 3-540-60943-1, London, UK, 1997.*

Juan, E.Y.T., Tsai, Jeffrey J.P., Murata, T. & Zhou, Yi., (2001). Reduction Methods for real-Time Systems Using Delay Time Petri nets, *IEEE Transactions on Software Engineering, ISSN: 0098-5589, Vol. 27, N°5, pp. 422-448, May 2001.*

Labadi, K., Chen, H. & Amodeo, L. (2005). Application des BDSPNs à la Modélisation et à l'Evaluation de Performance des Chaînes Logistiques", *Journal Européen des Systèmes Automatisés, DOI:10.3166/jesa.39.863-886, Vol.39/7, pp. 863-886, 2005.*

Labadi, K., Chen, H. & Amodeo, L. (2007). Modeling and Performance Evaluation of Inventory Systems Using Batch Deterministic and Stochastic Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, ISSN: 1094-6977, Vol. 37, N°6, pp. 1287-1302, 2007.*

Lee-Kwang, H. & Favrel, J., (1985). Hierarchical Reduction Methods for analysis and Decomposition of Petri Nets, *IEEE Transactions on Systems, Man, and Cybernetics, ISSN 0018-9472, Vol. 15, pp. 272-280, March 1985.*

Lee-Kwang, H. & Favrel, J., and Paptiste, P. (1987). Generalized Petri Nets Reduction Method", *IEEE Transactions on Systems, Man, and Cybernetics, ISSN: 0018-9472, Vol. 17, N°2, pp. 297-303, April 1987.*

Li, Yao. & Woodside, C. Murray., (1995). Complete Decomposition of Stochastic Petri Nets Representing Generalized Service Networks, *IEEE Transactions on Computers, ISSN: 0018-9340, Vol. 44, N° 4, pp. 577–592, April 1995.*

Lindemann, C., (1998). Performance Modelling with Deterministic and Stochastic Petri Nets, *John Wiley and Sons, ISBN: 0471976466, New York, USA, 1998.*

Ma, J. & Zhou, M.C., (1992). Performance Evaluation of Discrete Event Systems via Stepwise Reduction and Approximation of Stochastic Petri Nets, *Proceedings of the 31st IEEE Conference on decision and Control, ISBN: 0-7803-0872-7, Vol.1, pp. 1210-1215, Tucson, Arizona, USA, 1992.*

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE, ISSN: 0018-9219, Vol. 77, N° 4, pp. 541-580, April, 1989.*

Thapa, D., Dangol, S., & Wang, Gi-Nam. (2005). Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping technique.

Computational Intelligence for Modelling, Control and Automation, ISBN: 0-7695-2504-0, Vol. 2, pp. 228-233, Nov. 2005.

Wang, J., Deng, Yi. & Zhou M.C., (2000). Compositional Time Petri Nets and Reduction Rules, *IEEE Transactions on Systems, Man, and Cybernetics, Part B, ISSN: 1083-4419, Vol. 30, N° 4, pp.562-572, August 2000.*

New Developments in Robotics Automation and Control

Edited by Aleksandar Lazinica

ISBN 978-953-7619-20-6
Hard cover, 450 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

This book represents the contributions of the top researchers in the field of robotics, automation and control and will serve as a valuable tool for professionals in these interdisciplinary fields. It consists of 25 chapter that introduce both basic research and advanced developments covering the topics such as kinematics, dynamic analysis, accuracy, optimization design, modelling , simulation and control. Without a doubt, the book covers a great deal of recent research, and as such it works as a valuable source for researchers interested in the involved subjects.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Labadi Karim, Amodeo Lionel and Chen Haoxun (2008). Batch Deterministic and Stochastic Petri Nets and Transformation Analysis Methods, New Developments in Robotics Automation and Control, Aleksandar Lazinica (Ed.), ISBN: 978-953-7619-20-6, InTech, Available from: http://www.intechopen.com/books/new_developments_in_robotics_automation_and_control/batch_deterministic_and_stochastic_petri_nets_and_transformation_analysis_methods

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen