Generation of Mathematical Programming Representation for Discrete Event Simulation Models of Timed Petri Nets

Mengyi Zhang[a], Arianna Alfieri[b], Andrea Matta[a]

[a] Department of Mechanical Engineering, Politecnico di Milano, via La Masa 1, 20156 Milano, Italy
[b] Department of Management and Industrial Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy

ABSTRACT
This work proposes a mathematical programming (MP) representation of discrete event simulation of timed Petri nets (TPN). Currently, mathematical programming techniques are not widely applied to optimize discrete event systems due to the difficulty of formulating models capable to correctly represent the system dynamics. This work connects the two fruitful research fields, mathematical programming and timed Petri nets. In the MP formalism, the decision variables of the model correspond to the transition firing times and the markings of the TPN, whereas the constraints represent the state transition logic and temporal sequences among events. The MP model and a simulation run of the TPN are then totally equivalent, which is validated through several applications. Using a TPN model as input, the MP model can be routinely generated and used as white box for further tasks such as sensitivity analysis, cut generation in optimization procedures, and proof of formal properties.

KEYWORDS
discrete event simulation, mathematical programming, timed Petri net

1. Introduction

Discrete event dynamic systems (DEVS) have wide application in manufacturing and service applications. Even though many theoretical and analytical studies on discrete event dynamic systems have been developed, performance of real systems with high degree of complexity can be, sometimes, only evaluated with discrete event simulation (DES). Hence, simulation–optimization algorithms are widely used when numerical performance evaluation must be coupled with optimization, i.e., when the best system configuration, according to some criteria, has to be found meanwhile guaranteeing a given value of some performance measures (Fu, 2015).

Most of the simulation–optimization studies consider DES as a *black–box* function. Under the black–box setting, simulation and optimization modules are decoupled. The simulation module provides the performance measure of a given solution generated by the optimization module, which can help a further optimization round. However, the information given by simulation in a black–box setting is quite limited. Hence, the

CONTACT Mengyi Zhang. Email: mengyi.zhang@polimi.it

optimization module is almost blind and cannot be very effective, which leads to possibly many trials. This is the main cause for large computational inefficiencies of the overall black–box procedure.

A promising direction to improve the efficiency of black–box procedure is to merge the system dynamics into optimization, i.e., considering DES as *white–box* and modeling system dynamics as part of mathematical programming (MP) models. Several works studying such issue can be found in the literature. To cite a few examples, an MP model was proposed to minimize the makespan of single–server manufacturing systems in (Di Marino, Su, & Basile, 2020), the buffer allocation problem in multi–stage production flow lines with blocking is addressed in (Matta, 2008), (Weiss & Stolletz, 2015) and (Alfieri, Matta, & Pastore, 2020), the optimization of multi stage systems with complex blocking mechanisms are studied in (Pedrielli, Alfieri, & Matta, 2015). Decomposition or linear approximation approaches are applied to solve the MP models, which improve the efficiency of black–box algorithms, thus enhancing also optimality (Weiss & Stolletz, 2015)(Alfieri et al., 2020).

However, white–box simulation–optimization approaches are not as widely–applied as black–box ones. The reason is that developing a white–box algorithm requires to master discrete event dynamic systems, simulation and mathematical programming, and these disciplines are not tightly connected. The above mentioned works are all tailored to specific cases, and there is a lack of generalization to extend them to other problems. To fill this gap, this work proposes an approach to translate the simulation of a timed Petri net (TPN) into MP models.

In the literature, a few works address similar problems, i.e., translating the behavior of general discrete event dynamic systems into mathematical programming representation. Bemporad and Morari (1999) proposed a mixed integer programming modeling framework for hybrid systems, whose states are mixed integer, in discrete time. Differently, this work deals with continuous–time discrete–state systems. Chan and Schruben (2008) proposed a modeling framework to translate a DES model into an MP model starting from an Event Relationship Graph (ERG) representation of the system dynamics. However, despite its generality the ERG representation is not as commonly–used as TPNs. Indeed, developing an ERG model is time consuming and as a consequence their approach is not used often. On the other hand, TPNs are well–known representations taught in most engineering study courses, and software tools for easy creation of TPN models also exist.

The benefits of using MP models to represent discrete event systems are many. As already mentioned, when coupled with optimization, the MP-based algorithms can be faster and reach better solution than black–box optimization algorithms. Furthermore, black–box approaches have limited capability in solving constrained optimization problems, while MP–based approaches can easily deal with them (Zhang & Matta, 2020). The vast theoretical and methodological results developed in the MP field can be introduced into the study of DEVS through simulation. For instance, using sensitivity analysis of MP models, gradient estimates can be easily derived as suggested in (Chan & Schruben, 2008). In fact, this paper does not propose to totally replace simulation with MP, but to enrich the toolbox for analyzing and optimizing DEVS. Finally, the application of MP models of DEVS is not limited to optimization but also to show formal properties of the system under study. For instance, Basile, Chiacchio, and De Tommasi (2012) proposed the sufficient and necessary condition of K-diagnosability of TPN based on MP analysis.

The major concern about the application of MP is the computational complexity, as the solution procedure can be time–consuming or even unbearable. However, many approaches from optimization community are available to improve the efficiency. Linear programming approximation (Alfieri & Matta, 2012), Benders decomposition (Weiss & Stolletz, 2015), row–column generation (Alfieri et al., 2020), have been studied to solve optimization problems in real contexts based on mathematical programming representation of DEVS.

The contribution of this work is to propose the first framework translating TPNs into MP models. The translation procedure can be conveniently implemented in general–purpose programming languages and turned into an automated procedure. This work lays a foundation for developing MP–based approaches that can be used to analyze and optimize TPNs, based on the vast literature available in the mathematical programming field.

The rest of the paper is organized as follows. Section 2 briefly introduces the TPN and its simulation algorithm. Section 3 describes the generation of the MP model from an existing TPN model. Section 4 shows examples of application and validation of the equivalence between simulation and the MP. Discussion and conclusion are reported in Section 5.

2. Timed Petri Nets

2.1. Basic definitions and notations

A TPN is a directed bipartite graph. The set of nodes in that graph is partitioned into a set of *places* P and a set of *transitions* T. Places and transitions are connected with weighted and directed edges. In the example shown in the right side of Figure 1, places (i.e., p_1, p_2, p_3), transitions (i.e., t_1, t_2) and tokens are represented by white circles, rectangles and small black circles, respectively. Places hold *tokens*, and tokens are transferred through transition *firings*, i.e., one transition firing absorbs tokens from some places and releases them to some other places. Each transition $t \in T$ can absorb tokens from all of its precedents, and the number of tokens absorbed from place $p \in P$ is equal to the weight of the edge connecting p and t. The number of tokens in each place must be non–negative at any time; hence, a firing of transition t is enabled if and only if the number of tokens in all its precedents p is not smaller than the weight on edge (p, t). Each transition $t \in T$ can release tokens to all of its successors, and the number of tokens released to place $p \in P$ is equal to the weight of the edge connecting t and p. It is not necessary that the set of precedents and the set of successors are mutually exclusive. In a TPN, the duration between the moment of absorption and the

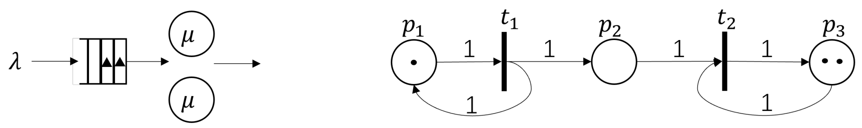

Figure 1. G/G/2 queue and its TPN model.

moment of release is not always negligible, and the duration is called *firing time*. This work considers firing time that are non–negative and can follow arbitrary stochastic distribution.

The distribution of tokens in the places is called *markings*. The initial state of the system, i.e., the state at time zero, is represented by *the initial marking*.

Formally, a TPN can be defined as a 6–tuple $N = (P, T, F, W, M_0, \tau)$, where P denotes the set of places, T denotes the set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ denotes the set of edges, $W : F \to \mathbb{Z}^+$ denotes the weights on edges, $M_0 : P \to \mathbb{N}$ denotes the initial marking, $\tau : T \to$ *random distribution* denotes the firing time of transitions.

The illustrative example showed in Figure 1 is a G/G/2 queue, in which customers arrive at the queue following an arbitrary arrival process. One of the two identical servers (or the idle one if only one is idle) processes customers in the queue one by one, and the processing time is arbitrarily distributed. After processing, the customer will be immediately released from the system, and the server becomes idle. On the right side of Figure 1, the TPN graph of the G/G/2 system is reported. A token in place p_1 shows that it is possible to schedule the arrival of the next customer, and the firing time of transition t_1 is equal to the inter-arrival time. At the finish of firing t_1, a customer arrives, and it is possible to schedule the next arrival, thus, it releases one token to p_1. Furthermore, an arrival will increase the length of the queue by 1, which is represented by the number of tokens in place p_2. Two tokens in place p_3 shows that two servers are idle. The firing of transition t_2 shows the processing of a customer, and it is enabled if and only if at least one server is idle and there is a customer in the queue. Once the process of the customer finishes, i.e., the finish of firing t_2, a server is released, and a token is sent to place p_3.

2.2. Simulation of Timed Petri Nets

The event-scheduling approach is the logic behind all the major DES software and used by practitioners when developing simulation codes with general purpose languages (Law, 2014). For sake of completeness, the logic for simulating TPN is briefly described in this section. This description will also be useful in the next section to better point out the consistency of the generated MP models with the system dynamics.

Within the event scheduling worldview of discrete event simulation (Zeigler, Muzy, & Kofman, 2018), an event is first scheduled and occurs afterwards. An *event list* stores all the scheduled events that will occur in the future. The event with the earliest occurring time is polled to occur from the event list, and the clock is advanced to its occurring time. The system state is changed upon the event occurrence, and the new state enables the schedule of new events. A new cycle starts from polling the earliest event. In TPN, events, whose occurrence changes the system state instantaneously, are either start or finish of transition firings in a TPN, since they are the only moments when the marking is changed. Thus, an event in a simulation realization of TPN can be represented by a triple $(t, type, time)$, where t is a transition, *type* is either *start* or *finish*, and *time* is the occurring time.

Given a TPN $N = (P, T, F, W, M_0, \tau)$, its simulation can be implemented as in Algorithm 1. In Algorithm 1, event list, clock time and system state (i.e., the markings) are initialized as empty, zero and initial markings, respectively. Each iteration is composed of two steps, i.e., to start all the enabled transitions and to finish the firing in the event list with the earliest finishing time. In the first step, if firing of a certain transition t_1 is enabled by the current markings, the firing can be started and tokens are immediately removed from precedent places of t. In the mean time, the finish of the firing can be scheduled with a delay equal to the firing time, which is a sample τ_0 of random distribution $\tau(t_1)$, and the finish of firing is added to the event list. It can be noticed that the event list contains only finish–of–firing events, since

Algorithm 1 Simulating a TPN.

```
Input:
   Initial markings: $M_0$.
1: Initialization:
2: $EventList = \{\}$; simulation clock: $clock \leftarrow 0$; current markings: $M \leftarrow M_0$;
3: iteration counter: $k \leftarrow 0$.
4: while stopping condition is not true do
5:    for each transition $t_1 \in T$ do
6:       while current marking enables the firing of $t_1$, i.e., $M(p_1) \geq W(p_1, t_1)$, $\forall (p_1, t_1) \in F$ do
7:          Sample the firing time $\tau_0$ of random variable $\tau(t_1)$
8:          for each precedent place $p_1$ of $t_1$ do
9:             Remove $W(p_1, t_1)$ tokens from $p_1$: $M(p_1) \leftarrow M(p_1) - W(p_1, t_1)$
10:         end for
11:         Add event $(t_1, finish, clock + \tau_0)$ to event list.
12:      end while
13:   end for
14:   From $EventList$, take the event $(t_2, finish, time)$ with the earliest time.
15:   Advance the clock: $clock \leftarrow time$.
16:   for each successive place $p_2$ of $t_2$ do
17:      Add $W(t_2, p_2)$ tokens to $p_2$: $M(p_2) \leftarrow M(p_2) + W(t_2, p_2)$.
18:   end for
19:   Update iteration counter: $k = k + 1$
20: end while
```

all the start–of–firing events are executed immediately without going into the list. In the second step, the firing with the earliest finish time will be taken from the event list, and the tokens are released to the successive places. The iterative procedure stops when a given condition is reached, which is usually a given number of iterations or a value of the clock time.

3. Generation of Mathematical Programming Representations

This section introduces the MP models translating the dynamics of the TPN described in Algorithm 1. Such MP models are mathematical programming representations (MPR) of the TPNs under study. Specifically, the time when transition firings start and finish and markings can all be seen (in the MP model) as decision variables. The time when transition t starts to fire for the i-th time is denoted by $e_i^{t,s}$, and $e_i^{t,f}$ denotes the time when the fire finishes. Variables $\mathcal{E}_k$ denotes the simulation clock at the beginning of iteration k in Algorithm 1. Variables $e_i^{t,s}$, $e_i^{t,f}$ and $\mathcal{E}_k$ are all real–valued and non–negative. Variable m_k^p is used to denote the number of tokens in place p at the beginning of iteration k in Algorithm 1. Some binary variables are also used in the MPR, and they will be introduced in the following, during the explanation of the model.

3.1. Completion of transitions

The first group of mathematical relationships, denoted by group–A, are the constraints for firing finishing as in lines 14 and 15 of Algorithm 1.

First of all, sets $\mathbb{K}$, $\mathbb{T}$ and $\mathbb{I}^t$ are used to indicate the set of iterations, the set of transitions and the set of firings of transition t, respectively. Binary variables $w_{i,k}^t$ are introduced to represent that the i-th start firing of transition t finishes in iteration k. If $w_{i,k}^t$ is equal to one, variable $e_i^{t,f}$ is bounded to $\mathcal{E}_k$, as shown in constraints (A1) and (A2). Constraints (A3) state that each firing-finish event executes at most once. Constraints (A4) state that, in each iteration, exactly one firing-finish event is executed. Constraints (A5) implies that the delay between the start and finish of firing transition t is equal to a sample from the random variable T^t. Constraints (A6) imply that the clock cannot be reversed from iteration $k-1$ to iteration k. Constraint (A7) implies that the simulation clock is initialized to zero. The value of M in constraints (A1) and (A2) can be set to a value that is larger than the simulation time span, as it is needed to bound the finishing time of transitions.

$$\begin{aligned}
e_i^{t,f} - \mathcal{E}_k &\geq M(w_{i,k}^t - 1) & \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t, k \in \mathbb{K} \quad & (A1)\\
\mathcal{E}_k - e_i^{t,f} &\geq M(w_{i,k}^t - 1) & \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t, k \in \mathbb{K} \quad & (A2)\\
\sum_{k\in\mathbb{K}} w_{i,k}^t &\leq 1 & \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t \quad & (A3)\\
\sum_{t\in\mathbb{T}}\sum_{i\in\mathbb{I}^t} w_{i,k}^t &= 1 & \forall\, k \in \mathbb{K} \quad & (A4)\\
e_i^{t,f} - e_i^{t,s} &= \tau_i^t & \forall t \in \mathbb{T}, i \in \mathbb{I}^t \quad & (A5)\\
\mathcal{E}_k - \mathcal{E}_{k-1} &\geq 0 & \forall\, k \in \mathbb{K} \quad & (A6)\\
\mathcal{E}_0 &= 0 & & (A7)
\end{aligned}$$

3.2. Start of transitions

The second group of constraints, denoted by group–B, depicts the start of transition firing, as in lines 5 to 13 in Algorithm 1. Binary variables $x_{i,k}^t$ represent that a fire of transition t starts in iteration k if it is equal to one. Constraints (B1) and (B2) shows that the start firing time is equal to the simulation clock. Also in this case, the value of M can be set to a value that is larger than the simulation time span as it is used to bound the transition firing time.

$$\begin{aligned}
e_i^{t,s} - \mathcal{E}_k &\geq M(x_{i,k}^t - 1) & \forall\, t \in \mathbb{T}, k \in \mathbb{K}, i \in \mathbb{I}^t \quad & (B1)\\
\mathcal{E}_k - e_i^{t,s} &\geq M(x_{i,k}^t - 1) & \forall\, t \in \mathbb{T}, k \in \mathbb{K}, i \in \mathbb{I}^t \quad & (B2)
\end{aligned}$$

The condition to start firing transition t is that the tokens in all the precedent places p are above the required level W^p, t. Binary variables z_k^t equal to one represents the condition for transition t is true, as constraints (B3). Moreover, a set of binary variables $v_k^{t,p}$ is used to verify if the number of tokens in precedent place p is smaller than $W^{p,t}$, as constraints (B4). The value of M can be chosen as the upper bound of marking of place p minus $(W^{p,t} - 1)$. Variable $v_k^{t,p}$ is equal to 1, if marking in place p

is smaller than $W^{p,t}$. Constraints (B5) assures that z_k^t is equal to zero only if at least one precedent place does not contain enough number of tokens. Constraints (B6) show that if z_k^t is equal to one, transition t must start to fire in iteration k. Constraints (B7) state that at most one firing of any transition starts for each transition.

$$
\begin{aligned}
m_k^p - W^{p,t} &\geq W^{p,t}(z_k^t - 1) && \forall\, t \in \mathbb{T}, k \in \mathbb{K}, p \in \mathbb{P} && (B3)\\
(W^{p,t} - 1) - m_k^p &\geq M(v_k^{t,p} - 1) && \forall\, t \in \mathbb{T}, k \in \mathbb{K}, p \in \mathbb{P} && (B4)\\
1 - z_k^t &\leq \sum_{p \in \mathbb{P}} v_k^{p,t} && \forall\, t \in \mathbb{T}, k \in \mathbb{K} && (B5)\\
\sum_{i \in \mathbb{I}^t} x_{i,k}^t &= z_k^t && \forall\, t \in \mathbb{T}, k \in \mathbb{K} && (B6)\\
\sum_{k \in \mathbb{K}} x_{i,k}^t &\leq 1 && \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t && (B7)
\end{aligned}
$$

Constraints (B7) avoid that certain markings can enable firings of the same transition to start multiple times. Thus, modifications are made to the TPN model before the proposed MPR model is applied. The modification is to expand the non–zero–firing-time transition t into two transitions, denoted by $\tilde{t}$ and $\bar{t}$, respectively. Transition $\tilde{t}$ has zero firing time and maintains all the precedent places and their weights of transition t, while transition $\bar{t}$ maintains the firing time, successive place and their weights of transition t. Transitions $\tilde{t}$ and $\bar{t}$ are both connected to an intermediate place p_t, with one arc directed from $\tilde{t}$ to place p_t and one arc directed from p_t to $\bar{t}$. An example is shown in Figure 2. This expansion can freeze the simulation clock before all the possible firings of transition t start.

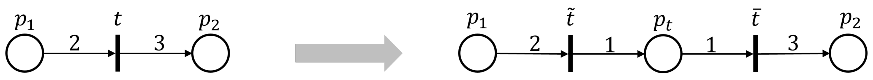

Figure 2. TPN expansion.

3.3. Event sequencing

The index i of event $e_i^{t,s}$ represents the sequence of firing starting, i.e., if a firing starts in an earlier iteration, then its index will be smaller. Moreover, a firing must finish after starting. Group–C constraints force such sequences, which are relevant to all the events. Constraints (C1) show that if the i–th firing of transition t does not start before simulation termination, then the $(i+1)$–th firing will not start. Constraints (C2) state that if the i–th firing of transition t does not start before simulation termination, then it will not finish. Constraints (C3) state that a firing cannot finish before starting, unless it remains in the future event list at the end of the simulation run. Constraints (C4) depict that the $(i+1)$–th firing of transition t must be scheduled after the i–th execution, unless the $(i+1)$–th execution is not scheduled before simulation termination. The value of M in constraints (C3) and (C4) can be set to K, i.e., the total number of iterations.

$$\sum_{k\in\mathbb{K}} x^t_{i+1,k} - \sum_{k\in\mathbb{K}} x^t_{i,k} \leq 0 \quad \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t \quad (C1)$$

$$\sum_{k\in\mathbb{K}} w^t_{i,k} - \sum_{k\in\mathbb{K}} x^t_{i,k} \leq 0 \quad \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t \quad (C2)$$

$$\sum_{k\in\mathbb{K}} k w^t_{i,k} - \sum_{k\in\mathbb{K}} k x^t_{i,k} \geq M(\sum_{k\in\mathbb{K}} w^t_{i,k} - 1) \quad \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t \quad (C3)$$

$$\sum_{k\in\mathbb{K}} k x^t_{i+1,k} - \sum_{k\in\mathbb{K}} k x^t_{i,k} \geq 1 + M(\sum_{k\in\mathbb{K}} x^t_{i+1,k} - 1) \quad \forall\, t \in \mathbb{T}, i \in \mathbb{I}^t \quad (C4)$$

3.4. State transitions

The tokens in place p are changed from m^p_k to m^p_{k+1} in iteration k, as in constraints (D1) and in lines 16 to 18 in Algorithm 1. The transitions t that start in firing in iteration k absorbs $A^{p,t}$ tokens from the precedent places p, and the firing that finishes in iteration k releases tokens to the successive places.

$$m^p_{k+1} = m^p_k - \sum_{t\in\mathbb{T}} W^{p,t} \sum_{i\in\mathbb{I}^t} x^t_{i,k} + \sum_{t\in\mathbb{T}} W^{t,p} \sum_{i\in\mathbb{I}^t} w^t_{i,k} \quad \forall p \in \mathbb{P}, k \in \mathbb{K} \quad (D1)$$

Equation (D1) can also be reformulated as (D2).

$$m^p_{k+1} = m^p_0 - \sum_{t\in\mathbb{T}} W^{p,t} \sum_{k'=0}^{k} \sum_{i\in\mathbb{I}^t} x^t_{i,k'} + \sum_{t\in\mathbb{T}} W^{t,p} \sum_{k'=0}^{k} \sum_{i\in\mathbb{I}^t} w^t_{i,k'} \quad \forall p \in \mathbb{P}, k \in \mathbb{K} \quad (D2)$$

3.5. Objective function

With the constraints defined in the previous sections, there is a unique feasible solution in terms of transition firing times and the sequence of simulation clock values, i.e., $e^{t,s}_i$, $e^{t,f}_i$ and $\mathcal{E}_k$. Thus, the objective function can be any function of these variables, such as, average system time or average waiting time in queueing systems. Notice that multiple feasible solutions may appear in terms of binary variables, since the sequence of events with identical execution time is not uniquely defined.

The flexibility of the objective function definition is a main difference between the formulation proposed by Chan and Schruben (2008) and the approach proposed in this work, since the objective function of MPR in Chan and Schruben (2008) can be only the sum of all the execution times. The uniqueness of the optimal solution and the flexibility of the objective function formulation is particularly beneficial if one wants to make use of the resulting MPR to solve an optimization problem related to the design or the operation of the discrete event system (e.g., the capacity of the queue or the control policy of a manufacturing system), since changing the objective function or adding new constraints to calculate the system performance will not influence the equivalence between the MPR and a simulation run.

4. Examples of MPR generation from TPN

In this section, the $G/G/m$ is used as a example to generate MPR from TPN. Both the proposed and the state-of-the-art formulation are presented. Finally, also an extended model to optimize the $G/G/m$ queue is stated.

4.1. MPR of G/G/m queue

A $G/G/m$ queue is composed by m parallel identical servers. Customers arrive at the queue following an general arrival process. One of the identical idle servers processes customers in the queue one by one, and the processing time of each customer is generally distributed. After being processed, the customer will be released from the system immediately, and the server becomes idle again. In Figure 3, the TPN of $G/G/2$ is shown. A token in place p_{arr} shows that it is possible to schedule the arrival of the next customer, and the firing time of transition t_{arr} is equal to the inter-arrival time. At the end of firing of t_{arr}, a customer arrives, and it is then possible to schedule the next arrival, i.e., it releases one token to p_{arr}. Furthermore, each arrival will increase the length of the queue by 1, which is represented by the number of tokens in place p_{queue}. Two tokens in place p_{idle} shows that two servers are idle. The firing of transition $t_{process}$ represents the processing of a customer, and it is enabled if and only if at least one server is idle and there is at least one customer in the queue. Once the process of the customer finishes, i.e., the end of firing of $t_{process}$, the server is released, and a token is sent to place p_{idle}.

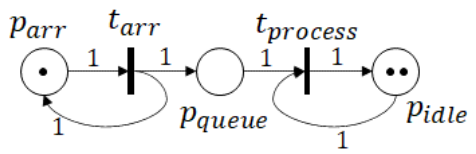

Figure 3. TPN of G/G/2.

As mentioned in Section 3.2, the transitions with non-zero firing times are expanded before implementing the MPR generation. In the case of G/G/2 queue, transition $t_{process}$ needs to be expanded to $\tilde{t}_{process}$, $\bar{t}_{process}$ and $p_{process}$. Transition t_{arr} does not need expansion since it is self-limiting. The resulting expanded TPN is shown in Figure 4.

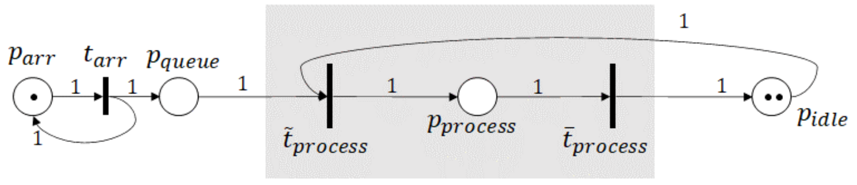

Figure 4. TPN expansion of G/G/2.

The sets $\mathbb{T}$, $\mathbb{K}$, $\mathbb{I}^t$ are specified as follows. Set $\mathbb{T}$ is composed of three transitions $\{t_{arr}, \tilde{t}_{process}, \bar{t}_{process}\}$ as shown in Figure 4. If n job arrival, processing and departure are simulated, each of the three transitions is fired for n times, and the set $\mathbb{I}^t$ is equal to $\{1, 2, ..., n\}$. There are $3n$ transition firings in the simulation execution and the set

$\mathbb{K}$ is equal to $\{0, 1, ..., 3n-1\}$. The initial markings $m_0^{p_{arr}}$, $m_0^{p_{queue}}$, $m_0^{p_{process}}$, $m^{p_{idle}}$ are specified as 1, 0, 0, 2. Once the TPN with the initial markings and the simulation length are provided, the MPR corresponding to constraints (A1) to (A7), (B1) to (B7), (C1) to (C4) and (D1) can be generated. The implementation pseudo code is presented in Algorithm 2.

$$\begin{aligned}\mathbb{T} &= \{t_{arr}, \tilde{t}_{proc}, \bar{t}_{proc}\}\\ \mathbb{K} &= \{0, 1, ..., 3n-1\}\\ \mathbb{I}^t &= \{1, 2, ..., n\}\end{aligned}$$

Algorithm 2 Implementation pseudo code.

```
Input:
   Timed Petri net: $N = (\mathbb{P}, \mathbb{T}, F, W, M_0, \tau)$.
   Number of firing of each transition $t$: $I^t$ and index set $\mathbb{I}^t$
   Total number of firings: $K$ and index set $\mathbb{K} = \{0, 1, ..., K-1\}$
 1: Expand TPN:
 2: for transition $t \in \mathbb{T}$: do
 3:    if $t$ has non-zero firing time then
 4:       Expand $t$ into $\tilde{t}, p_t, \bar{t}$ and update $N$.
 5:    end if
 6: end for
 7: for $t \in \mathbb{T}$, $i \in \mathbb{I}^t$, $k \in \mathbb{K}$ do
 8:    (A1), (A2), (B1), (B2)
 9: end for
10: for $t \in \mathbb{T}$, $i \in \mathbb{I}^t$ do
11:    (A3),(A5),(B7),(C1),(C2),(C3),(C4)
12: end for
13: for $k \in \mathbb{K}$ do
14:    (A4),(A6)
15: end for
16: for $t \in \mathbb{T}$, $k \in \mathbb{K}$, $p \in \mathbb{P}$ do
17:    (B3),(B4)
18: end for
19: for $t \in \mathbb{T}$, $k \in \mathbb{K}$ do
20:    (B5),(B6)
21: end for
22: for $p \in \mathbb{P}$, $k \in \mathbb{K}$ do
23:    (D1),(D2)
24: end for
```

Table 1 shows the first ten iteration of a simulation run of a G/G/m queue.

In iteration 0, the system is in its initial state with markings (1,0,0,2), and transition t_{arr} can be fired; equivalently, binary variables z_0^{arr} and $x_{1,0}^{arr}$ are equal to 1 in the solution of the MPR. The time to start firing transition t_{arr} is equal to 0, and the variable $e_1^{arr,s}$ takes value 0 in the MPR solution. The fire will finish at time 2.3, and variable $e_1^{arr,f}$ takes value 2.3 equivalently. As that firing finished in the current iteration, $w_{1,0}^{arr}$ is equal to 1. After that, the markings are changed to (1,1,0,2) and the current iteration ends.

At the beginning of iteration 1, the simulation clock is advanced to the time of the firing has finished in iteration 0, i.e., 2.3. In the MPR solution, the variable $\mathcal{E}_1$ is then equal to 2.3. With markings (1,1,0,2), it is possible to fire t^{arr} and $\tilde{t}^{proc}$, and variables z_1^{arr} and $z_1^{\tilde{proc}}$ are equal to 1. These firings are the second and first firing of t^{arr} and $\tilde{t}^{proc}$, respectively, and variables $x_{2,1}^{arr}$ and $x_{1,1}^{\tilde{proc}}$ take value 1 in the MPR solution. The starting time of the transitions t^{arr} and $\tilde{t}^{proc}$ are equal to the clock time, and variables $e_2^{arr,s}$ and $e_1^{\tilde{proc},s}$ are equal to 2.3. With a sample from the arrival process, the next customer will arrive at time 11.1, i.e., the finishing time of transition t^{arr} is equal to 11.1, which is the value of variable $e_2^{arr,f}$. Since transition $\tilde{t}^{proc}$ is zero-firing-time, the firing will finish at time 2.3, which is the value of variable $e_1^{\tilde{proc},f}$. There are two firings to be finished in the queue, and that of $\tilde{t}^{proc}$ has earlier time, so it will be the firing that will be finished in this iteration and, hence, variable $w_{1,1}^{\tilde{proc}}$ is equal to one. After that, the markings are changed to $(0, 0, 1, 1)$ and the iteration ends.

Similarly for iterations 2 to 10, the values of the MPR variables in the solution and the simulation realizations, which are reported in Table 1, can be explained as done for iterations 0 and 1.

Table 1. Simulation run and MPR solution of G/G/m queue.

k	*clock*	Markings	Start firings	Unfinished firings	Finish Firing
0	0	$(1,0,0,2)$	t^{arr}	$t_{arr}:2.3$	$t_{arr}:2.3$
	$\mathcal{E}_0=0$	$\mathbf{m}_0=(1,0,0,2)$	$z_0^{arr}=1, x_{1,0}^{arr}=1, e_1^{arr,s}=0, e_1^{arr,f}=2.3$		$w_{1,0}^{arr}=1$
1	2.3	$(1,1,0,2)$	$t^{arr}, \tilde{t}^{proc}$	$\tilde{t}_{proc}:2.3, t^{arr}:11.1$	$\tilde{t}_{proc}:2.3$
	$\mathcal{E}_1=2.3$	$\mathbf{m}_1=(1,1,0,2)$	$z_1^{arr}=1, x_{2,1}^{arr}=1, e_2^{arr,s}=2.3, e_2^{arr,f}=11.1$		$w_{1,1}^{p\tilde{r}oc}=1$
			$z_1^{p\tilde{r}oc}=1, x_{1,1}^{p\tilde{r}oc}=1, e_1^{p\tilde{r}oc,s}=2.3, e_1^{p\tilde{r}oc,f}=2.3$		
2	2.3	$(0,0,1,1)$	$\bar{t}_{proc}$	$\bar{t}_{proc}:6, t^{arr}:11.1$	$\bar{t}_{proc}:6$
	$\mathcal{E}_2=2.3$	$\mathbf{m}_2=(0,0,1,1)$	$z_2^{p\bar{r}oc}=1, x_{1,2}^{p\bar{r}oc}=1, e_1^{p\bar{r}oc,s}=2.3, e_1^{p\bar{r}oc,f}=6$		$w_{1,2}^{p\bar{r}oc}=1$
3	6	$(0,0,0,2)$		$t^{arr}:11.1$	$t^{arr}:11.1$
	$\mathcal{E}_3=6$	$\mathbf{m}_3=(0,0,0,2)$			$w_{2,3}^{arr}=1$
4	11.1	$(1,1,0,2)$	$t^{arr}, \tilde{t}^{proc}$	$\tilde{t}^{proc}:11.1, t^{arr}:12.1$	$\tilde{t}_{proc}:11.1$
	$\mathcal{E}_4=11.1$	$\mathbf{m}_4=(1,1,0,2)$	$z_4^{arr}=1, x_{3,4}^{arr}=1, e_3^{arr,s}=11.1, e_3^{arr,f}=12.1$		$w_{2,4}^{p\tilde{r}oc}=1,$
			$z_4^{p\tilde{r}oc}=1, x_{2,4}^{p\tilde{r}oc}=1, e_2^{p\tilde{r}oc,s}=11.1, e_2^{p\tilde{r}oc,f}=11.1$		
5	11.1	$(0,0,1,1)$	$\bar{t}_{proc}$	$t^{arr}:12.1, \bar{t}_{proc}:16.9$	$t^{arr}:12.1$
	$\mathcal{E}_5=11.1$	$\mathbf{m}_5=(0,0,1,1)$	$z_5^{p\bar{r}oc}=1, x_{2,5}^{p\bar{r}oc}=1, e_2^{p\bar{r}oc,s}=11.1, e_2^{p\bar{r}oc,f}=16.9$		$w_{3,5}^{arr}=1$
6	12.1	$(1,1,0,1)$	$t^{arr}, \tilde{t}^{proc}$	$\tilde{t}^{proc}:12.1, t^{arr}:15.2, \bar{t}_{proc}:16.9$	$\tilde{t}^{proc}:12.1$
	$\mathcal{E}_6=12.1$	$\mathbf{m}_6=(1,1,0,1)$	$z_6^{arr}=1, x_{4,6}^{arr}=1, e_4^{arr,s}=12.1, e_4^{arr,f}=15.2$		$w_{3,6}^{p\tilde{r}oc}=1$
			$z_6^{p\tilde{r}oc}=1, x_{3,6}^{p\tilde{r}oc}=1, e_3^{p\tilde{r}oc,s}=12.1, e_3^{p\tilde{r}oc,f}=12.1$		
7	12.1	$(0,0,1,0)$	$\bar{t}^{proc}$	$t^{arr}:15.2, \bar{t}^{proc}:16.9, \bar{t}^{proc}:20.1$	$t^{arr}:15.2$
	$\mathcal{E}_7=12.1$	$\mathbf{m}_7=(0,0,1,0)$	$z_7^{p\bar{r}oc}=1, x_{3,7}^{p\bar{r}oc}=1, e_3^{p\bar{r}oc,s}=12.1, e_3^{p\bar{r}oc,f}=20.1$		$w_{4,7}^{arr}=1$
8	15.2	$(1,1,0,0)$	t^{arr}	$\bar{t}^{proc}:16.9, t^{arr}:17.8, \bar{t}^{proc}:20.1$	$\bar{t}^{proc}:16.9$
	$\mathcal{E}_8=15.2$	$\mathbf{m}_8=(1,1,0,0)$	$z_8^{arr}=1, x_{5,8}^{arr}=1, e_5^{arr,s}=15.2, e_5^{arr,f}=17.8$		$w_{2,8}^{\bar{t}^{proc}}=1$
9	16.9	$(0,1,0,1)$	$\tilde{t}^{proc}$	$\tilde{t}^{proc}:16.9, t^{arr}:17.8, \bar{t}^{proc}:20.1$	$\tilde{t}^{proc}:16.9$
	$\mathcal{E}_9=16.9$	$\mathbf{m}_9=(0,1,0,1)$	$z_9^{p\tilde{r}oc}=1, x_{4,9}^{p\tilde{r}oc}=1, e_4^{p\tilde{r}oc,s}=16.9, e_4^{p\tilde{r}oc,f}=16.9$		$w_{4,9}^{p\tilde{r}oc}=1$
10	16.9	$(0,0,1,0)$	$\bar{t}^{proc}$	$t^{arr}:17.8, \bar{t}^{proc}:20.1, \bar{t}^{proc}:25.5$	$t^{arr}:17.8$
	$\mathcal{E}_{10}=16.9$	$\mathbf{m}_{10}=(0,0,1,0)$	$z_{10}^{p\bar{r}oc}=1, x_{4,10}^{p\bar{r}oc}=1, e_4^{p\bar{r}oc,s}=16.9, e_4^{p\bar{r}oc,f}=25.5$		$w_{5,10}^{arr}=1$

4.2. Comparison with Schruben's model of G/G/m systems

In Chan and Schruben (2008), an MPR of DEVS simulation based on ERG is presented. The ERG of the $G/G/2$ queue is shown in Figure 5. In the ERG, nodes and arcs represent the events and the triggering relationships. The simulation of the $G/G/2$ queue is composed of three events, which are *arrival*, *starting process* and *finishing process*, respectively. The three events are presented by nodes *Arr*, *Start proc* and *Fin proc* in the ERG. The node *Run* indicates the launch of the simulation. There are two state variables for $G/G/2$ queue, *queue* and *idle*, representing the number of jobs in the queue and the number of idle servers. The equations below each node state how the state variables are changed upon the execution of the events. For instance, when event *Arr* occurs, one job enters the system and *queue* is incremented by one. When there is an arc connecting two nodes, it means that it is possible to execute the destination event only after the source event is executed. The execution of an event can also be subject to certain conditions, reported as labels on the connecting arc. The conditions can be a time delay or a logic expression. The arc pointing from *Arr* to *Arr* is accompanied with a time delay τarr, indicating that the next job will arrive with a time delay equal to the inter-arrival time from the previous one. The arc pointing from *Arr* to *Start proc* is labeled with the logic expression $idle \geq 1$, indicating that after a job is arrived, it can start to be processed only if there is at least one idle server.

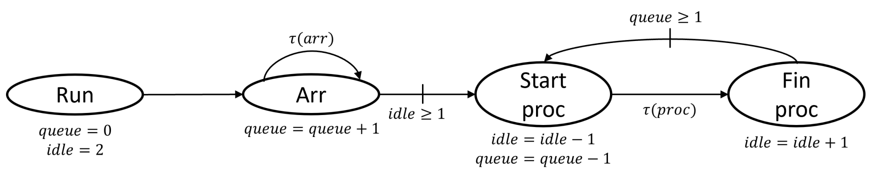

Figure 5. ERG of G/G/2.

The MPR derived from the ERG in Figure 5 is as follows:

$$\min \textstyle\sum_{i=1}^{n}(e_i^{Arr} + e_i^{Start\ proc} + e_i^{Finish\ proc}) \tag{1}$$

$$e_{i+1}^{Arr} - e_i^{Arr} \geq \tau_i^{Arr} \qquad \forall i = 1, ..., n-1 \tag{2}$$

$$e_i^{Start\ proc} - e_i^{Arr} \geq 0 \qquad \forall i = 1, ..., n \tag{3}$$

$$e_{i'}^{Fin\ proc} - e_i^{Start\ proc} \geq \tau_i^{proc} + M(\delta_{i,i'}^{Start\ proc, Fin\ proc} - 1) \qquad \forall i = 1, ..., n \tag{4}$$

$$\textstyle\sum_{i=1}^{n} \delta_{i,i'}^{Start\ proc, Fin\ proc} = 1 \qquad \forall i' = 1, ..., n \tag{5}$$

$$\textstyle\sum_{i'=1}^{n} \delta_{i,i'}^{Start\ proc, Fin\ proc} = 1 \qquad \forall i = 1, ..., n \tag{6}$$

$$e_{i+2}^{Start\ proc} - e_i^{Fin\ proc} \geq 0 \qquad \forall i = 1, ..., n-2 \tag{7}$$

$$e_{i+1}^{Start\ proc} - e_i^{Start\ proc} \geq 0 \qquad \forall i = 1, ..., n-1 \tag{8}$$

$$e_{i+1}^{Fin\ proc} - e_i^{Fin\ proc} \geq 0 \qquad \forall i = 1, ..., n-1 \tag{9}$$

$$e_i^{Arr} \geq 0, e_i^{Start\ proc} \geq 0, e_i^{Fin\ proc} \geq 0 \qquad \forall i = 1, ..., n \tag{10}$$

$$\delta_{i,i'}^{Startproc, Finproc} \in \{0, 1\} \tag{11}$$

The objective function (1) is the minimization of the execution time of all the events. Constraints (2) to (7) are derived from the arcs in the ERG, and indicate the event triggering relationships. The arc pointing from *Arr* to *Arr* generates constraints (2), which represent the inter-arrival time between two adjacent arrivals. The arc pointing from *Arr* to *Start proc* generates constraints (3), representing that a job can be processed only after it arrives. The arc pointing from *Start proc* to *Fin proc* generates the constraints (4) to (6), representing that a job can be finished only after the process starts and a certain processing time is elapsed. Since the processing time is generated from probability distributions, it can take different values for different jobs. A job started later can be finished before an earlier stated one, i.e., the ranking of the i-th started job in the finishing sequence can be different from i. Therefore, binary variables $\delta_{i,i'}^{Start\ proc,Fin\ proc}$ are introduced, and the i-th started job is the i'-th to finish if and only if $\delta_{i,i'}^{Start\ proc,Fin\ proc}$ is equal to one. The arc pointing from *Fin proc* to *Start proc* generates the constraints (7), and represent that a job can be started when there is an idle server. The subscripts i of variables $e_i^{Start\ proc}$ and $e_i^{Fin\ proc}$ indicates the occurrence sequence of the events, as in constraints (8) and (9).

This MPR model is different from the one proposed in this work in the following aspects. First, the system states, which are the decision variables in the model proposed in this work, are not explicitly modeled in the Chan and Schruben's model. Second, in the Chan and Schruben's model, the objective is to minimize all the event execution time. Instead, the formulation of objective function can be more flexible in the model proposed in this work, since executing events as soon as possible is already assured by the constraints. Third, Chan and Schruben (2008) only proposed the descriptive procedure to generate an MPR, without well-formulated mapping from ERG to MPR, which makes the application difficult. In this work, instead, the proposed method is well-formulated, and the only required input is the timed Petri net.

4.3. MPR optimizing the server number of G/G/m queue

An MP model optimizing the server number in a G/G/m queue, based on the MPR proposed in Section 4.1, can be formulated as follows. As known, the larger the number of servers, the smaller the waiting time. However, servers have a cost, and this is why their number should be minimized. The objective function (12) is then the minimization of the number of servers, which is the initial marking in place p^{idle}, i.e., variable m_0^{idle}. However, a constraints must be set on the maximum allowed waiting time, otherwise the obvious solution of number of servers equal to 1 is the one reached by the MPR solution.

$$\min m_0^{idle} \tag{12}$$

$$\textstyle\sum_{i=1}^{n}(e_i^{t,f} - e_i^{t,f}) \leq wt \cdot n \tag{13}$$

$$(A1) - (A7),\ (B1) - (B7),\ (C1) - (C4),\ (D1)$$

Constraint (13) indicates that the average waiting time does not have to exceed a target value wt. Constraints (A1) - (A7), (B1) - (B7), (C1) - (C4), (D1) are the same as presented in Section 4.1.

5. Conclusion

This work proposes a well-formulated framework to translate general TPNs into MP models. The convenience of the formulation procedure is demonstrated using the example of a $G/G/m$ queue.

This work lays a common foundation for developing MP–based approaches that can be used to analyze and optimize TPNs, based on the vast literature available in the mathematical programming field.

In fact, even though many existing works in the literature are related to this research direction, they are mainly focused on specific applications, especially in manufacturing systems, and a methodological work is still mussing, which will be the object of future research development.

References

Alfieri, A., & Matta, A. (2012). Mathematical programming formulations for approximate simulation of multistage production systems. *European Journal of Operational Research*, *219*(3), 773–783.

Alfieri, A., Matta, A., & Pastore, E. (2020). The time buffer approximated buffer allocation problem: A row–column generation approach. *Computers & Operations Research*, *115*, 104835.

Basile, F., Chiacchio, P., & De Tommasi, G. (2012). On k-diagnosability of petri nets via integer linear programming. *Automatica*, *48*(9), 2047–2058.

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, *35*(3), 407–427.

Chan, W. K., & Schruben, L. (2008). Optimization models of discrete-event system dynamics. *Operations Research*, *56*(5), 1218–1237.

Di Marino, E., Su, R., & Basile, F. (2020). Makespan optimization using timed petri nets and mixed integer linear programming problem. *IFAC-PapersOnLine*, *53*(4), 129–135.

Fu, M. (Ed.). (2015). *Handbook of simulation optimization.* Springer.

Law, A. M. (2014). *Simulation modeling and analysis* (Vol. 5). McGraw-Hill New York.

Matta, A. (2008). Simulation optimization with mathematical programming representation of discrete event systems. In *2008 winter simulation conference* (pp. 1393–1400).

Pedrielli, G., Alfieri, A., & Matta, A. (2015). Integrated simulation–optimisation of pull control systems. *International Journal of Production Research*, *53*(14), 4317–4336.

Tan, B. (2015). Mathematical programming representations of the dynamics of continuous-flow production systems. *IIE Transactions*, *47*(2), 173–189.

Weiss, S., & Stolletz, R. (2015). Buffer allocation in stochastic flow lines via sample-based optimization with initial bounds. *OR spectrum*, *37*(4), 869–902.

Zeigler, B. P., Muzy, A., & Kofman, E. (2018). *Theory of modeling and simulation: discrete event & iterative system computational foundations.* Academic press.

Zhang, M., & Matta, A. (2020). Models and algorithms for throughput improvement problem of serial production lines via downtime reduction. *IISE Transactions*, 1–15.

Zhang, M., Matta, A., & Alfieri, A. (2020). Sample-path algorithm for global optimal solution of resource allocation in queueing systems with performance constraints. In *2020 winter simulation conference (wsc)* (pp. 2767–2778).

Zhang, M., Pastore, E., Matta, A., & Alfieri, A. (2021). Buffer allocation problem in production flow lines: a new benders decomposition based exact solution approach. *under review*.