sensors

MDPI

Article

Performance Optimization for a Class of Petri Nets

Weijie Shi [1], Zhou He [2,*], Chan Gu [2], Ning Ran [3] and Ziyue Ma [4]

[1] School of Electro-Mechanical Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China
[2] School of Electrical and Control Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China
[3] College of Electronic and Information Engineering, Heibei University, Baoding 071002, China
[4] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
* Correspondence: hezhou@sust.edu.cn

Abstract: Petri nets (PNs) are widely used to model flexible manufacturing systems (FMSs). This paper deals with the performance optimization of FMSs modeled by Petri nets that aim to maximize the system's performance under a given budget by optimizing both quantities and types of resources, such as sensors and devices. Such an optimization problem is challenging since it is nonlinear; hence, a globally optimal solution is hard to achieve. Here, we developed a genetic algorithm combined with mixed-integer linear programming (MILP) to solve the problem. In this approach, a set of candidate resource allocation strategies, i.e., the choices of the number of resources, are first generated by using MILP. Then, the choices of the type and the cycle time of the resources are evaluated by MILP; the promising ones are used to spawn the next generation of candidate strategies. The effectiveness and efficiency of the developed methodology are illustrated by simulation studies.

Keywords: flexible manufacturing systems; Petri nets; evolutionary techniques; resource allocation

Citation: Shi, W.; He, Z.; Gu, C.; Ran, N.; Ma, Z. Performance Optimization for a Class of Petri Nets. *Sensors* **2023**, *23*, 1447. https://doi.org/10.3390/s23031447

Academic Editor: Mengchu Zhou

Received: 27 December 2022
Revised: 19 January 2023
Accepted: 21 January 2023
Published: 28 January 2023

1. Introduction

Performance optimization in flexible manufacturing systems has received great attention in recent decades [1]. Such automated systems with synchronization, concurrency, and time delay can usually be modeled by timed marked graphs (TMGs), an important subclass of PNs that are proven to be useful in modeling manufacturing systems, queue systems, and railway transportation systems [2–10]. The performance of a system modeled with TMGs is usually characterized by the cycle time that represents the normalized time for one cycle of production. This issue was first studied in ordinary TMG (i.e., TMGs in which the weights on the arcs are unitary). In the literature, ordinary TMGs are simply called "TMGs" by omitting the term "ordinary". The cycle time can be obtained by exploring the periodic dynamic evolution of a TMG [11]. On the other hand, the exact cycle time of a TMG can also be computed analytically by using the linear programming technique [12]. Algebraic approaches based on (max,+) or (min,+) algebra have also been applied for computing the cycle time of TMGs [13]. Based on these results, various methods have been developed for the performance optimization problems in TMGs, i.e., to minimize the total cost of the system's resources (machines, devices, tasks, etc.) without compromising the desired level of throughput. The optimization targets include the initial resource allocation [14–20] and the server-type selection under a given budget [21–23].

Timed weighted marked graphs (TWMGs) are more general than ordinary TMGs and are useful in modeling systems with batch processes (i.e., instead of processing tasks one by one, they wait until a batch of tasks are available and process them simultaneously) [24,25]. Nevertheless, obtaining the exact cycle time of a TWMG is much more complicated than that of a TMG. In [26], the authors prove that for a given TWMG, the optimization of the server type is NP-hard. A widely used method for computing the cycle time of a TWMG is to first transform it into an equivalent TMG [27,28] followed by the linear programming

technique in [12]. However, such a transformation is computationally heavy, initially marking-dependent, and usually yields a TMG much larger than the original TWMG. To overcome such a problem, in [29], a method is proposed to transform a TWMG (with an undetermined initial marking) into an equivalent parametric TMG so that the scale of the corresponding linear programming remains solvable. Some approximation methods [30] are also developed to estimate the upper bound of the cycle time of a TWMG. A linearization method for TWMGs to obtain linear representation in (min,+) was developed [31]. Based on the obtained linear model, the performance of the TWMGs can be analyzed.

The performance optimization problem in TWMGs is also more complex than that of TMGs. Relatively few works have been conducted on the performance optimization of TWMGs in the literature. Since the equivalent TMG classes of a TWMG is finite, by solving a mixed-integer linear programming problem (MILPP) for each equivalent TMG class, a globally optimal initial marking can be obtained [32,33]. Nevertheless, the computational load of this precise algorithm is unsatisfactory, since the number of equivalent TMG classes is exponential in the number of places in the corresponding TWMG. Therefore, people turn to hybrid algorithms, such as heuristics combined with simulation and MILPPs, which are fast and provide near-optimal solutions [34–36]. In [34], a TWMG is initialized with a sufficiently large initial marking so that the performance requirement is satisfied. Then, a greedy algorithm is applied to decrease the total cost in which tokens are iteratively removed until the cycle time reaches a lower bound threshold. A heuristic method combined with integer linear programming was proposed in [35]. However, both heuristics in [34,35] rely on the detection of the so-called critical places and, hence, a time-consuming simulation of the TWMG has to be performed in each iteration. This method is further improved in [36] in which the critical places are determined by linear programming, which greatly improves the efficiency of the heuristic algorithm. Computational efficient solutions for the performance optimization of TWMGs are still missing.

In practice, the execution time of a process may vary when different types of resources (machine, robot, transporting vehicle, etc.) are deployed. Therefore, the optimization of the types and quantities of resources, (so that the performances are maximized) has also received considerable attention in different areas, such as manufacturing systems [37–39] and enterprise systems [40]. However, as far as we know, no work has been done on such a problem in TWMGs. Solving this problem is challenging since it is a nonlinear quadratic optimization (which we will see in Section 3) and, thus, a globally optimal solution is hard to achieve. The search space for selecting the types and quantities of resources is quite huge even though the system size is small.

In this paper, we explore the performance optimization problem in TWMGs where both quantities and types of resources are simultaneously optimized. The exact value of the cycle time of TWMGs is efficiently evaluated by using linear algebraic methods. Precisely speaking, our aim is to minimize the cycle time of a given TWMG subject to a given budget limit, while both quantities and types of resources are variables.

The main contribution of this work is summarized as follows. We developed a hybrid method that is a genetic heuristic combined with MILP. Initially, a set of candidate resource allocation strategies, i.e., the choices of the number of resources, are first generated by solving a MILPP. Then, among all of the possible types of resources whose numbers grow exponentially with respect to the system size, we originally developed a MILPP to determine the optimal one. This was done by transforming the determined TWMG of the candidate strategy into an equivalent TMG with parametric firing delays. In the latter, a MILPP was applied to compute its cycle time and the gross cost (scored as fitness). The promising strategies, i.e., those with high fitness, were used to spawn the next generation of candidate strategies. The procedure above was repeated for a pre-set number of generations, and eventually, a near-optimal resource allocation strategy was obtained. Simulation results show that the developed approach can provide a good trade-off between the near-optimality of the solution and the computational load.

This paper is structured as follows. The basic concepts used in this paper are discussed in Section 2. Section 3 formulates the performance optimization problem studied in this paper. In Section 4, a genetic algorithm-based approach is presented. Simulation studies are investigated in Section 5. Conclusions and future work are presented in Section 6.

2. Background

2.1. Generalities

A PN is a 4-tuple $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$, where P is a set of n places; T is a set of m transitions; $\boldsymbol{Pre} : P \times T \to \mathbb{N}$ and $\boldsymbol{Post} : P \times T \to \mathbb{N}$ are the pre- and post-incidence functions that specify the arcs, respectively, and are also denoted by matrices in $\mathbb{N}^{n\times m}$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$; $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre} \in \mathbb{Z}^{n\times m}$ is the incidence matrix, where $\mathbb{Z} = \{0, \pm 1, \pm 2, \ldots\}$. A PN is said to be ordinary when all of its arc weights are unitary, i.e., $\boldsymbol{Pre}, \boldsymbol{Post} \in \{0, 1\}^{n\times m}$.

A vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{N}^n$ (resp., $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^T \in \mathbb{N}^m$) such that $\boldsymbol{y} \neq \boldsymbol{0}$ and $\boldsymbol{y}^T \cdot \boldsymbol{C} = \boldsymbol{0}$ (resp., $\boldsymbol{x} \neq \boldsymbol{0}$ and $\boldsymbol{C} \cdot \boldsymbol{x} = \boldsymbol{0}$) is a P-semiflow (resp., T-semiflow). The support of a P-semiflow (resp., T-semiflow) is defined by $||\ \boldsymbol{y}\ ||=\{p_i \in P \mid y_i > 0\}$ (resp., $||\ \boldsymbol{x}\ ||=\{t_i \in T \mid x_i > 0\}$). A P-semiflow $\boldsymbol{y}$ (resp., T-semiflow $\boldsymbol{x}$) is minimal if (i) $||\ \boldsymbol{y}\ ||$ (resp., $||\ \boldsymbol{x}\ ||$) is not a superset of the support of any other P-semiflow (resp., T-semiflow), and (ii) all y_i, y_j (resp., x_i, x_j) where $i \neq j$ are mutually prime.

The set of output (resp., input) places of transition $t_i \in T$ is defined as $t_i^\bullet = \{p \in P \mid Post(p, t_i) > 0\}$ (resp., ${}^\bullet t_i = \{p \in P \mid Pre(p, t_i) > 0\}$). The notions for ${}^\bullet p$ and $p^\bullet$ are analogously defined.

A marking is a mapping $\boldsymbol{M} : P \to \mathbb{N}$ that assigns to each place of a PN a non-negative integer number of tokens, which is also described as an n-component vector $\boldsymbol{M} \in \mathbb{N}^n$. The number of tokens of place p at marking $\boldsymbol{M}$ is denoted by $M(p)$. A Petri net system $\langle N, \boldsymbol{M_0}\rangle$ is a net N with an initial marking $\boldsymbol{M_0}$.

A weighted marked graph (WMG) is a PN in which each place has exactly one input and one output transition. A marked graph (MG) is a WMG whose weights on arcs are unitary.

Given a PN N, a path is a sequence of nodes $o_1 o_2 \cdots o_q$ where $o_i \in P \cup T$ for all $i \in \{1, \ldots, q\}$ and $o_{i+1} \in o_i^\bullet$ holds for all $i \in \{1, \ldots, q-1\}$. A PN is said to be strongly connected if, for any $o, o' \in P \cup T$, there exists a path from o to o'. A path $o_1 o_2 \cdots o_q$ is a circuit if $o_1 = o_q$. A circuit $o_1 o_2 \cdots o_{q-1} o_1$ is an elementary circuit, denoted by γ, if for all $i, j \in \{1, \ldots, q\}$, $i \neq j$ indicates $o_i \neq o_j$. The set of all elementary circuits of N is denoted by Γ.

A WMG is neutral if for each elementary circuit γ it holds that $\prod_{p_j \in \gamma} \frac{Pre(p_j, p_j^\bullet)}{Post(p_j, {}^\bullet p_j)} = 1$. In the rest of this paper, we limit our study to neutral and strongly connected WMGs.

2.2. Timed Petri Nets

In the literature, two types of timed Petri nets (TPNs) [41] are mainly studied, namely transition-timed PNs (TTPNs) and place-timed PNs (PTPNs) [42]. A transition-timed Petri net is a pair $N_d = (N, \boldsymbol{\delta})$ where

- $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ is a PN;
- $\boldsymbol{\delta} = [\delta(t_1), \delta(t_2), \ldots, \delta(t_m)]^T \in \mathbb{Q}_{\geq 0}^m$ is an m-component vector, called the firing delay vector of N_d, which assigns a non-negative rational value to each transition to represent its firing delay, where $\mathbb{Q}$ denotes the set of rational numbers.

For a PTPN, a non-negative rational duration $\delta(p)$ is assigned to each place p to represent the residual time that a token must spend in p before it becomes available for its output transitions.

Given a TTPN $N_d = (N, \boldsymbol{\delta})$, the enabling degree of a transition t logically enabled at a marking $\boldsymbol{M}$, denoted as $en(\boldsymbol{M}, t)$, is the largest integer r, such that $r \cdot \boldsymbol{Pre}(\cdot, t) \leq \boldsymbol{M}$. The TTPNs considered in this paper follow the so-called infinite server semantics [42]. In plain words, each transition represents an operation that can be executed as many times as

the number of available servers, i.e., each enabled transition can fire as many times as its enabling degree.

A TTPN $(N, \boldsymbol{\delta})$ is called a TWMG if net N is a weighted marked graph. A TWMG system is a triple $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M}_0 \rangle$, where $(N, \boldsymbol{\delta})$ is a TWMG and $\boldsymbol{M_0}$ is an initial marking.

2.3. Cycle Time of TWMGs

The cycle time of a TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M}_0 \rangle$, denoted by $\chi(G)$, is the average period to fire one time the minimal T-semiflow $\boldsymbol{x} = [x_1, \ldots, x_m]$ [28,34]. The steady evolution of a strongly connected TWMG system is repetitive. Let f_i denote the firing frequency of transition t_i during the repetitive period. Then, the cycle time of a TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M}_0 \rangle$ can be defined as:

$$\chi(G) = \frac{x_i}{f_i}, \forall t_i \in T. \tag{1}$$

We denote by $\chi_\gamma(G)$ the cycle time of elementary circuit γ and by $\chi_{\gamma^*}(G) = \max_{\gamma \in \Gamma} \chi_\gamma(G)$ the critical time. For a TMG system, the cycle time is equal to the critical time, i.e.,

$$\chi(G) = \chi_{\gamma^*}(G).$$

However, for a TWMG system, reference [35] shows that the cycle time is greater than or equal to the critical time, i.e.,

$$\chi(G) \geq \chi_{\gamma^*}(G).$$

The cycle time of a TWMG can be determined by the simulation. The work of [12,30] provides a lower bound and an upper bound of the cycle times that are easy to compute. However, in general, such approximations may be far from the actual cycle time and, hence, cannot be used as guidelines for performance optimization. On the other hand, a TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M} \rangle$ can be transformed into an equivalent place-timed marked graphs (PTMGs) system $\hat{G} = \langle \hat{N}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{M}} \rangle$ as shown in Algorithm 1, which describes the same behavior, transition firing language, and cycle time [28]. Then, the cycle time $\chi(G)$ of the original TWMG system can be obtained by solving the following linear programming problem (LPP) for the equivalent PTMG system $\hat{G}$ [12,33]:

$$\begin{aligned} &\min \ \chi(\hat{G}) \\ &\text{s.t.} \quad \hat{\boldsymbol{C}} \cdot \boldsymbol{\alpha} + \chi(\hat{G}) \cdot \hat{\boldsymbol{M}} \geq \boldsymbol{D}_p \cdot \hat{\boldsymbol{Post}} \cdot \boldsymbol{v} \end{aligned} \tag{2}$$

where $\chi(G) = \chi(\hat{G}) \in \mathbb{R}^+$, $\boldsymbol{\alpha} \in \mathbb{R}^{\hat{m}}$, $\boldsymbol{v} = \vec{\boldsymbol{1}}_{\hat{m} \times 1}$ is the visit ratio vector, and $\boldsymbol{D}_p \in \mathbb{N}^{\hat{n} \times \hat{n}}$ is a diagonal matrix such that

$$D_p(i,j) = \begin{cases} \hat{\delta}(p_i), & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Note that $t_{in(p_j)}$, $t_{out(p_j)}$, $\lfloor \cdot \rfloor$, and $\lceil \cdot \rceil$ in Algorithm 1 represent the unique input transition of place p_j, the unique output transition of place p_j, the floor operator, and the ceiling operator, respectively.

Algorithm 1 Transformation of a TWMG into a PTMG [12].

Input: A TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M} \rangle$ and the minimal T-semiflow $\boldsymbol{x} = (x_1, \ldots, x_m)^T$ of net N;

Output: An equivalent PTMG system $\hat{G} = \langle \hat{N}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{M}} \rangle$ such that $\chi(\hat{G}) = \chi(G)$;

for each transition $t_i \in T$ **do**

delete t_i and its corresponding arcs;

add x_i transitions $t_i^1, t_i^2, \ldots, t_i^{x_i}$ and x_i places $q_i^1, q_i^2, \ldots, q_i^{x_i}$ such that ${}^\bullet q_i^a = t_i^a$, $q_i^{a\bullet} = t_i^{a \bmod x_i + 1}$, $\forall a = 1, \ldots, x_i$;

$$\text{let} \begin{cases} \hat{M}(q_i^1) = \hat{M}(q_i^2) = \cdots = \hat{M}(q_i^{x_i - 1}) := 0, \\ \hat{M}(q_i^{x_i}) := 1, \\ \hat{\delta}(q_i^1) = \hat{\delta}(q_i^2) = \cdots = \hat{\delta}(q_i^{x_i}) := 0. \end{cases} \tag{4}$$

end for

for each place $p_j \in P$ begin **do**

delete p_j and its corresponding arcs;

let $a := 0$, $s := 1$;

repeat

$$b := \left\lfloor \frac{M(p_j) + Post(p_j, t_{in(p_j)}) \cdot a}{Pre(p_j, t_{out(p_j)})} + 1 \right\rfloor \tag{5}$$

$$a := \left\lceil \frac{Pre(p_j, t_{out(p_j)}) \cdot b - M(p_j)}{Post(p_j, t_{in(p_j)})} \right\rceil \tag{6}$$

if $a \leq x_{in(p_j)}$ **then**

Add place p_j^s such that ${}^\bullet p_j^s = t_{in(p_j)}^a$, $p_j^{s\bullet} = t_{out(p_j)}^{(b-1) \bmod x_{out(p_j)} + 1}$

$$\hat{M}(p_j^s) := \left\lfloor \frac{b-1}{x_{out(p_j)}} \right\rfloor \tag{7}$$

$$\hat{\delta}(p_j^s) := \delta(t_{in(p_j)}) \tag{8}$$

$$n_j := s \tag{9}$$

$s := s + 1$

end if

until $a \geq x_{in(p_j)}$

end for

3. Problem Statement

In a TWMG that models a practical system, each transition t_i models an operation performed by machines, robots, etc., which we call the servers. A server is modeled by a server place p_{s_i} self-looped with t_i. We consider the performance optimization problem in TWMGs where both quantities and types of servers are variables to be simultaneously optimized. Precisely speaking:

- For each transition, t_i with $k_i \in \mathbb{N}_{>0}$ choices of server types, we use a k_i-component binary vector $\boldsymbol{Z}_i = [Z_i(1), Z_i(2), \ldots, Z_i(k_i)]^T \in \{0,1\}^{k_i}$ to describe the server type selection of it, where

$$Z_i(j) = \begin{cases} 1, & \text{the } j\text{-th type of server is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, to ensure that the operation is consistent, we require that—among all of the k_i types of servers of transition t_i—only one type can be selected, i.e.,

$$Z_i(1) + Z_i(2) + \cdots + Z_i(k_i) = 1$$

- For each transition t_i, different servers have different performances (i.e., the firing delay) and costs. We use $\delta(t_i, j)$ and $\lambda(t_i, j)$ ($j \in \{1, \ldots, k_i\}$) to denote the firing delay and the unit cost of server type j, respectively. We denote the firing delay vector of transition t_i and the unit cost vector of transition t_i by $\boldsymbol{\delta}(t_i) = [\delta(t_i,1), \delta(t_i,2), \ldots, \delta(t_i,k_i)]^T \in \mathbb{Q}_{\geq 0}^{k_i}$, and $\boldsymbol{\lambda}(t_i) = [\lambda(t_i,1), \lambda(t_i,2), \ldots, \lambda(t_i,k_i)]^T \in \mathbb{Q}_{\geq 0}^{k_i}$, respectively.
- For each transition t_i, we use $M(p_{s_i})$ to denote the server quantity of it, i.e., the number of servers to equip. Then, we use $\boldsymbol{\mathcal{M}} = [M(p_{s_1}), M(p_{s_2}), \ldots, M(p_{s_m})]^T \in \mathbb{N}^m$ to represent the server quantity of all servers in the gross TWMG system.

Therefore, the firing delay $\delta(t_i)$ and the unit cost $\lambda(t_i)$ of transition t_i are determined by the following equation:

$$\begin{aligned} &\lambda(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\lambda}(t_i), \\ &\delta(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\delta}(t_i), \\ &Z_i(1) + Z_i(2) + \ldots + Z_i(k_i) = 1, \\ &i = 1, \ldots, m. \end{aligned} \tag{10}$$

Moreover, the choices of server types and quantities must be subject to a given budget limit $R \in \mathbb{R}$. Then, we can formulate the performance optimization problem as the following.

Problem 1 (Performance optimization problem)**.** *Given a TWMG model N, a budget $R \in \mathbb{R}$, a unit cost vector $\boldsymbol{\lambda}(t_i)$, and a firing delay vector $\boldsymbol{\delta}(t_i)$, determine a server-type selection $\boldsymbol{Z}_i = [Z_i(1), Z_i(2), \ldots, Z_i(k_i)]^T$ and a server quantity $\boldsymbol{\mathcal{M}} = [M(p_{s_1}), M(p_{s_2}), \ldots, M(p_{s_m})]^T$, such that $\boldsymbol{\lambda}^T \cdot \boldsymbol{\mathcal{M}} \leq R$ while the cycle time $\chi(G)$ is minimized (i.e., the performance is maximized).*

In plain words, solving Problem 1 is equivalent to solving the following optimization problem

$$\begin{aligned} &\min\ \chi(G) \\ &\text{s.t.} \begin{cases} \boldsymbol{\lambda}^T \cdot \boldsymbol{\mathcal{M}} \leq R, & \text{(11a)} \\ \left.\begin{aligned} &\lambda(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\lambda}(t_i), \\ &\delta(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\delta}(t_i), \\ &Z_i(1) + Z_i(2) + \ldots + Z_i(k_i) = 1, \\ &i = 1, \ldots, m, \end{aligned}\right\} & \text{(11b)} \end{cases} \end{aligned} \tag{11}$$

where

- $\chi(G)$ represents the cycle time of the TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M} \rangle$,
- $\boldsymbol{\lambda} = [\lambda(t_1), \ldots, \lambda(t_m)]^T$ represents the cost vector of the TWMG,
- $\boldsymbol{\mathcal{M}} = [M(p_{s_1}), \ldots, M(p_{s_m})]^T$ is the server quantity of the TWMG,
- $\boldsymbol{\delta} = [\delta(t_1), \ldots, \delta(t_m)]^T$ is the firing delay vector of the TWMG.

However, we note that solving this problem is challenging since it is a quadratic optimization: in general, a globally optimal solution is hard to achieve. Therefore, in the next section, we develop a hybrid method composed of genetic heuristics and MILP. The following example will be used as a running example in the sequel of this paper.

Example 1. *Consider the TWMG model N depicted in Figure 1 that represents a cyclic manufacturing system. It is composed of two different operations performed by machines $\mathcal{MA}_1$ and $\mathcal{MA}_2$, respectively. Transitions t_1 and t_2 represent the operation performed by machines $\mathcal{MA}_1$ and $\mathcal{MA}_2$, respectively. Place p_1 is an idle place that represents the raw material to proceed, place p_2 is an activity place that represents the manufacturing process, and places p_{s_1} and p_{s_2} are the server places corresponding to t_1 and t_2, respectively. Tokens in places p_{s_1} and p_{s_2} represent the*

numbers of machines $\mathcal{MA}_1$ and $\mathcal{MA}_2$, respectively. Initially, there are one hundred raw materials to be proceeded, i.e., $M(p_1) = 100$, and no semi-products are being produced, i.e., $M(p_2) = 0$.

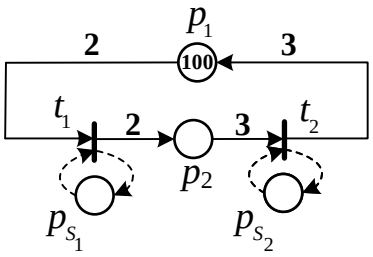

Figure 1. The TWMG model for Example 1.

Assume that we have $k_1 = 3, k_2 = 2$, i.e., transition t_1 has three choices of server types and transition t_2 has two choices. The firing delay and the unit cost of each type choice are listed in Table 1.

$$\begin{cases} \boldsymbol{\lambda}(t_1) = [\lambda(t_1,1), \lambda(t_1,2), \lambda(t_1,3)]^T = [4,10,15]^T, \\ \boldsymbol{\lambda}(t_2) = [\lambda(t_2,1), \lambda(t_2,2)]^T = [5,9]^T, \\ \boldsymbol{\delta}(t_1) = [\delta(t_1,1), \delta(t_1,2), \delta(t_1,3)]^T = [15,4,2]^T, \\ \boldsymbol{\delta}(t_2) = [\delta(t_2,1), \delta(t_2,2)]^T = [20,18]^T. \end{cases}$$

The performance optimization problem (11) is to determine the quantities of servers of transitions $\boldsymbol{\mathcal{M}} = [M(p_{s_1}), M(p_{s_2})]^T$ and the server-type selections $\boldsymbol{Z}_1 = [Z_1(1), Z_1(2), Z_1(3)]$ and $\boldsymbol{Z}_2 = [Z_2(1), Z_2(2)]$ subject to a given budget R so that the cycle time of the system is minimized.

Table 1. The unit cost and the firing delays of each server on the transitions for Example 1.

Transition t_i	Server Type j	Unit Cost of Server Type j $\boldsymbol{\lambda}(t_i, j)$	Firing Delay $\boldsymbol{\delta}(t_i, j)$ [s]
t_1	1	4	15
	2	10	4
	3	15	2
t_2	1	5	20
	2	9	18

4. Genetic MILP Approach for Performance Optimization in TWMGs

Due to the existence of weights on arcs of the TWMGs and the nonlinear constraint (11a), it is quite difficult to find an analytical approach to solve problem (11). Hence, in this section, we develop a hybrid method composed of genetic heuristics and mixed-integer linear programming, which makes the problem solvable.

As mentioned before, a possible server allocation strategy consists of two parts $(\boldsymbol{\mathcal{M}}, \boldsymbol{Z})$ where $\boldsymbol{\mathcal{M}}$ characterizes the choice of the quantity while $\boldsymbol{Z} = (\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_m)$ characterizes the choice of the server type. In our method, the two parts are optimized in an alternative manner. For small systems, it could be possible to enumerate the server type $\boldsymbol{Z}$ first and determine the quantity $\boldsymbol{\mathcal{M}}$ subsequently. However, it is not possible for practical systems even with medium sizes. In addition, it was shown in [33] that for a TWMG whose server type $\boldsymbol{Z}$ is given, the optimal quantity $\boldsymbol{\mathcal{M}}$ can be obtained by transforming it into a finite number of equivalent PTMG classes and solving a MILPP for each equivalent class. The computational cost of this approach is high since the number of equivalent PTMG classes increases exponentially w.r.t. the number of places of the original TWMG. In practice, it is inefficient to solve the performance optimization in TWMGs by exploring all the equivalent PTMGs.

In this paper, the quantity-component $\boldsymbol{\mathcal{M}}$, i.e., the initial token distribution for server places in the TWMG, is first treated as a chromosome in the genetic heuristics. Then, among all the possible server types $\boldsymbol{Z}$ whose number grows exponentially with respect to

the system size, we originally develop a MILPP to determine the optimal one. Therefore, the computational cost can be reduced significantly compared with the existing methods in [33]. Moreover, the result from the MILPP is also used to score the chromosome, i.e., the fitness of the quantity–component $\mathcal{M}$, which provides a guideline for generating the next generation of candidate chromosomes. This alternated optimization is done when a pre-given number (*Ge*) that represents the maximum number of generation is reached. The main steps of the approach are sketched in the flow chart in Figure 2.

In the following subsections, we will discuss the main elements of the developed approach:

- Coding and decoding;
- Initial population generation;
- Calculation of the objective function and fitness value;
- The overall genetic algorithm: selection, crossover, and mutation.

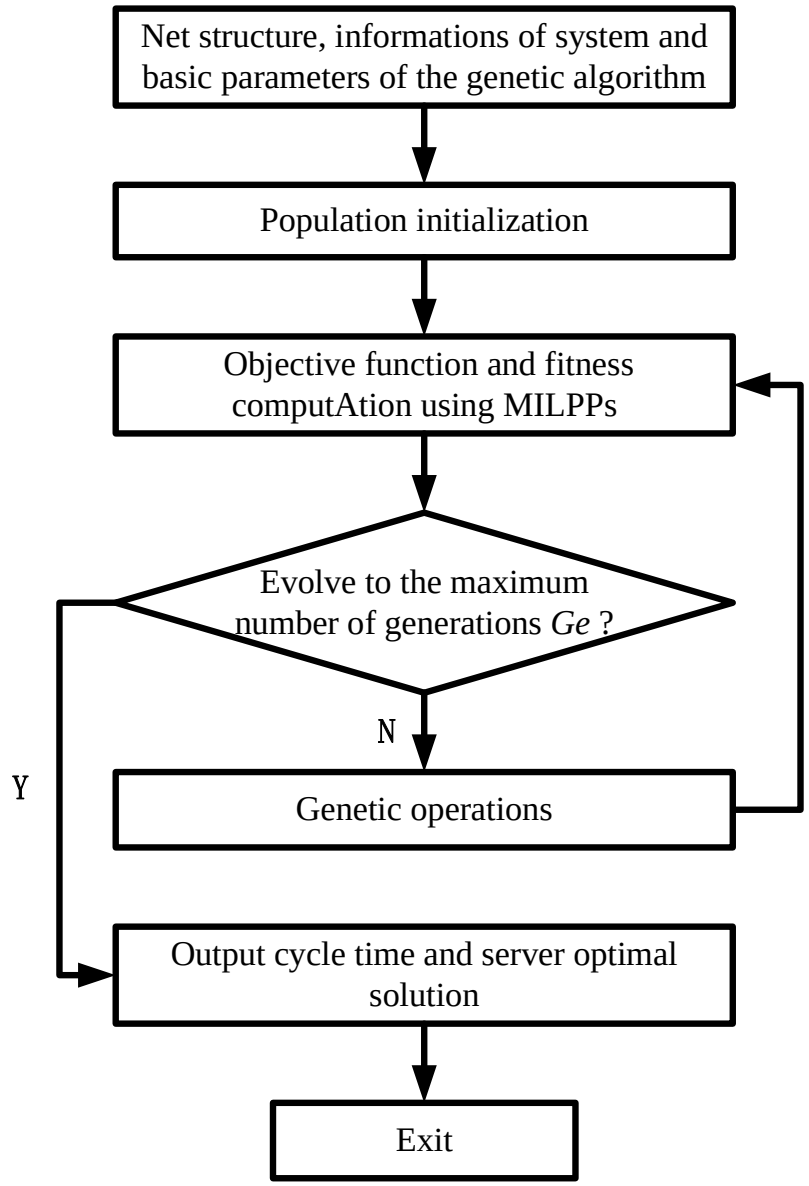

Figure 2. Flow chart of the developed approach based on genetic algorithm.

4.1. Coding and Decoding

A server allocation strategy (i.e., server quantity) $\boldsymbol{\mathcal{M}}$ can be coded as a chromosome $\boldsymbol{Chm}$ that is an m-dimensional nonnegative integer vector such that

$$\boldsymbol{Chm} = [e_1, e_2, \ldots, e_m]^T = \boldsymbol{\mathcal{M}}, \tag{12}$$

where e_i represents the quantities of servers of transition t_i, i.e., the number of tokens in the server place p_i^s. For instance, the chromosome of the TWMG model N depicted in Figure 1 can be expressed as $\boldsymbol{Chm} = [e_1, e_2]^T$, where $e_1 = M(p_1^s)$ and $e_2 = M(p_1^2)$. Therefore, we can decode $\boldsymbol{Chm}$ into a marking $\boldsymbol{M_{Chm}} = [4, 0, e_1, e_2]^T$ of the TWMG model.

4.2. Population Generation and Feasibility Screening

In each iteration in the genetic algorithm, a set of candidate chromosomes are generated from promising ones in the previous generation. However, not all of the newly generated chromosomes—in our case the initial server distribution—are feasible. For example, some initial server distribution may result in the system eventually dying. Hence, a pre-screening step is performed to remove such unfeasible chromosomes from the pool of candidates.

First, a marking $\boldsymbol{M_{Chm}}$ that is decoded by a chromosome $\boldsymbol{Chm}$ should guarantee the liveness of a TWMG. A TWMG is live if and only if each elementary circuit is live [24]. Hence, a feasible chromosome necessarily satisfies the following condition:

$$(\forall \gamma \in \Gamma)\ \boldsymbol{y}^T \cdot \boldsymbol{M_{Chm}} > \boldsymbol{y}^T \cdot \boldsymbol{M_D^\gamma} \tag{13}$$

where $\boldsymbol{y}$ is the minimal T-semiflow corresponding to γ and $\boldsymbol{M_D^\gamma} = [Pre(p_1, p_1^\bullet) - 1, \ldots, Pre(p_n, p_n^\bullet) - 1]^T$ is a marking restricted to γ.

On the other hand, a chromosome $\boldsymbol{Chm}$ is feasible only if there exists at least one server-type selection $\boldsymbol{Z}_i = [Z_i(1), \ldots, Z_i(k_i)]$ for every transition $t_i \in T$, such that the total cost of servers does not beyond the budget R. This indicates that the following constraint is necessarily feasible:

$$\begin{cases} \boldsymbol{\lambda}^T \cdot \boldsymbol{\mathcal{M}} \leq R, \\ \boldsymbol{\mathcal{M}} = \boldsymbol{Chm}, \\ \lambda(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\lambda}(t_i), \\ Z_i(1) + Z_i(2) + \ldots + Z_i(k_i) = 1, \end{cases} \tag{14}$$

Combining the results in (13) and (14), both the liveness and feasibility of a chromosome $\boldsymbol{Chm}$ can be guaranteed by the following constraints:

$$\begin{cases} \boldsymbol{y} \cdot \boldsymbol{M_{Chm}} > \boldsymbol{y} \cdot \boldsymbol{M_D^\gamma}, \forall \gamma \in \Gamma, \\ \boldsymbol{\lambda}^T \cdot \boldsymbol{\mathcal{M}} \leq R, \\ \boldsymbol{\mathcal{M}} = \boldsymbol{Chm}, \\ \lambda(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\lambda}(t_i), \\ Z_i(1) + Z_i(2) + \ldots + Z_i(k_i) = 1, \end{cases} \tag{15}$$

where $\boldsymbol{M_{Chm}}$ is a marking decoded from chromosome $\boldsymbol{Chm}$.

4.3. Objective Function and Fitness Score: A MILPP Approach

Given a chromosome $\boldsymbol{Chm}$ and its decoded marking $\boldsymbol{M_{Chm}}$ of a TWMG model N, we aim to determine the optimal server-type selection $\boldsymbol{Z} = (\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_m)$, such that the cycle time $\chi(G)$ of the TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M_{Chm}} \rangle$ is minimized while the cost of the servers corresponding to $(\boldsymbol{\mathcal{M}}, \boldsymbol{Z})$ do not exceed the budget R.

As we discussed in Section 2.3, a TWMG system $G = \langle N, \boldsymbol{\delta}, \boldsymbol{M_{Chm}} \rangle$ with a known initial marking $\boldsymbol{M_{Chm}}$ can be transformed into an equivalent PTMG system $\hat{G} = \langle \hat{N}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{M}}_{Chm} \rangle$ with the same cycle time. Hence, the cycle time $\chi(G)$ of G can be obtained by solv-

ing LPP (2) for its equivalent PTMG system $\hat{G}$. Note that here the firing delay vector $\boldsymbol{\delta} = [\delta(t_1), \ldots, \delta(t_m)]^T$ of G is dependent on the server-type selection $\boldsymbol{Z}$ which is undetermined yet. Hence, the firing delays in the corresponding equivalent PTMG system $\hat{G}$ $\hat{\boldsymbol{\delta}} = [\hat{\delta}(p_1), \ldots, \hat{\delta}(p_{\hat{n}})]$ is also to be determined. Now we show such an optimal server-type selection can be obtained by solving a MILPP for the equivalent PTMG.

Proposition 1. *Given a performance optimization problem (11) for a TWMG system G whose equivalent PTMG system is $\hat{G}$, let $(\chi(\hat{G}), \boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\delta}, \hat{\boldsymbol{\delta}}, \boldsymbol{D}_p, \boldsymbol{Z}_i)$ be the optimal solution of the following MILPP:*

$$
\begin{aligned}
&\min\ \chi(\hat{G}) \\
&\text{s.t.}\ \begin{cases}
\hat{\boldsymbol{C}} \cdot \boldsymbol{\alpha} + \chi(\hat{G}) \cdot \hat{\boldsymbol{M}}_{\boldsymbol{Chm}} \geq \boldsymbol{D}_p \cdot \hat{\boldsymbol{Post}} \cdot \boldsymbol{v}, & \text{(16a)} \\
\left.\begin{aligned} \boldsymbol{\lambda}^T \cdot \boldsymbol{\mathcal{M}} \leq R, \\ \boldsymbol{\mathcal{M}} = \boldsymbol{Chm}, \end{aligned}\right\} & \text{(16b)} \\
\left.\begin{aligned} \lambda(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\lambda}(t_i), \\ \delta(t_i) = \boldsymbol{Z}_i^T \cdot \boldsymbol{\delta}(t_i), \end{aligned}\right\} & \text{(16c)} \\
Z_i(1) + Z_i(2) + \ldots + Z_i(k_i) = 1, & \text{(16d)} \\
i = 1, \ldots, m, & \text{(16e)} \\
\left.\begin{aligned} \hat{\delta}(q_i^a) = 0,\ a = 1, \ldots, x_i, \\ \hat{\delta}(p_j^s) = \delta(t_q),\ \forall^\bullet p_j = t_q,\ s = 1, \ldots, n_j \end{aligned}\right\} & \text{(16f)}
\end{cases}
\end{aligned}
\tag{16}
$$

Then, $\boldsymbol{Z}_i$ $(i = 1, \ldots, m)$ is an optimal server-type selection of problem (11) with respect to a given server quantity $\boldsymbol{\mathcal{M}} = \boldsymbol{Chm}$.

Proof. Constraint (16b) ensures that the choice of server types subject to server quantity $\boldsymbol{\mathcal{M}} = \boldsymbol{Chm}$ does not exceed the budget R. Constraints (16c), (16d), and (16e) jointly enforce Equation (10). Combining the results in Equations (2) and (6), constraint (16f) ensures the correctness of the firing delay equivalence between the original TWMG and the PTMG. According to the results in [12,33], constraint (16a) can provide an optimal solution for performance optimization if $\hat{\boldsymbol{C}}$ and $\boldsymbol{M}_{\boldsymbol{Chm}}$ are given. Therefore, $\boldsymbol{Z}_i$ $(i = 1, \ldots, m)$ is an optimal server-type selection of problem (11) restricted to a given server quantity $\boldsymbol{\mathcal{M}} = \boldsymbol{Chm}$. □

Given a chromosome $\boldsymbol{Chm}$, the cycle time $\chi(\hat{G})$ in the optimal solution of MILPP (16) is denoted as $F(\boldsymbol{M}_{\boldsymbol{Chm}}) = \chi^*$. Let $\mathcal{N}_i$ denote the pool of chromosomes in the i-th iteration, i.e., the i-th generation, we define

$$F_{min} = \min_{\boldsymbol{Chm} \in \mathcal{N}_i} F(\boldsymbol{Chm})$$

and

$$F_{max} = \max_{\boldsymbol{Chm} \in \mathcal{N}_i} F(\boldsymbol{Chm})$$

the minimum and maximum objective functions (i.e., cycle time) of the population in the current generation, respectively. Then, the fitness of a chromosome $\boldsymbol{Chm}$ is defined as

$$fitness(\boldsymbol{Chm}) = 1 - \frac{F(\boldsymbol{Chm}) - F_{min}}{F_{max} - F_{min}} \tag{17}$$

Example 2. *Let us consider again the TWMG model N discussed in Example 1 with a minimal T-semiflow $\boldsymbol{x} = [3, 2]^T$ and a chromosome $\boldsymbol{Chm} = [4, 5]^T$. Then, the decoded initial marking is $\boldsymbol{M}_{\boldsymbol{Chm}} = [100, 0, 4, 5]^T$. The equivalent PTMG system $\hat{G} = \langle \hat{N}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{M}}_{\boldsymbol{Chm}} \rangle$ corresponding to $\boldsymbol{M}_{\boldsymbol{Chm}}$ is shown in Figure 3, where the firing delay vector $\hat{\boldsymbol{\delta}}$ is to be determined.*

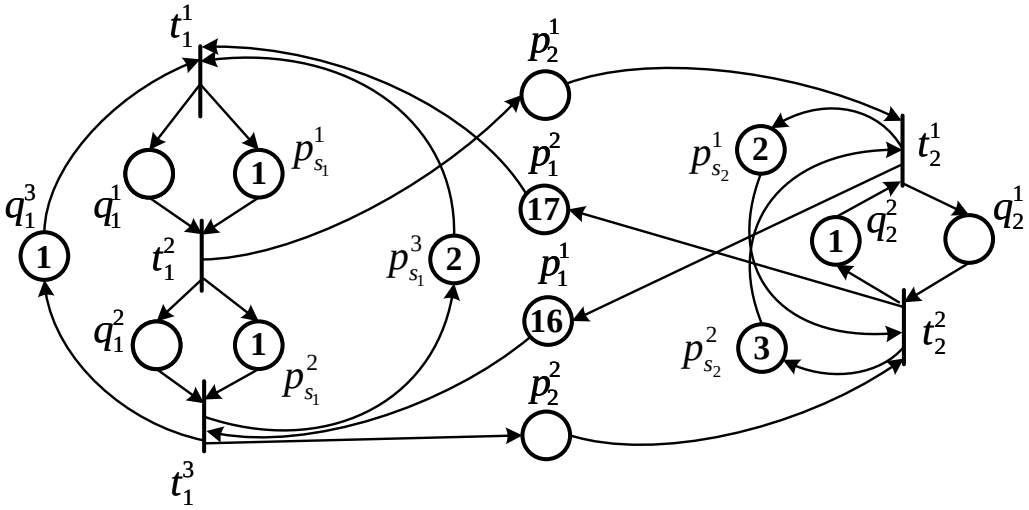

Figure 3. The equivalent PTMG corresponding to the TWMG model N in Figure 1 with an initial marking $\boldsymbol{M_{Chm}} = [100, 0, 4, 5]$.

Let $R = 100$ be the budget limit. According to Proposition 1, a MILPP can be established as shown in Equation (18). By solving MILPP (18), we obtain $\boldsymbol{Z} = (\boldsymbol{Z}_1, \boldsymbol{Z}_2)$ where $\boldsymbol{Z}_1 = [0, 1, 0]^T$ and $\boldsymbol{Z}_2 = [0, 1]^T$. Thus, to optimize the performance of the PTMG (which is equal to that of the original TWMG), transition t_1 should be equipped with the second type of server while transition t_2 should be equipped with the second type. The optimal cycle time $\chi(G) = \chi(\hat{G}) = 7.2$ of the TWMG system $G = \langle N, , \boldsymbol{\delta}, \boldsymbol{M_{Chm}} \rangle$. Finally, the objective function of chromosome $\boldsymbol{Chm} = [4, 5]^T$ is $F(\boldsymbol{Chm}) = 7.2$, and its fitness can be computed according to Equation (17).

$$
\begin{aligned}
&\min\ \chi(\hat{G})\\
&s.t.\begin{cases}
\left.\begin{aligned}
&\hat{\boldsymbol{C}} \cdot \boldsymbol{\alpha} + \chi(\hat{G}) \cdot \hat{\boldsymbol{M}}_{\boldsymbol{Chm}} \geq \boldsymbol{D}_p \cdot \hat{\boldsymbol{Post}} \cdot \boldsymbol{v},\\
&\boldsymbol{\lambda}^T \cdot \boldsymbol{\mathcal{M}} \leq R,\\
&\boldsymbol{\mathcal{M}} = \boldsymbol{Chm},
\end{aligned}\right\}\\
\left.\begin{aligned}
&\lambda(t_1) = Z_1(1) \cdot \lambda(t_1, 1) + Z_1(2) \cdot \lambda(t_1, 2) + Z_1(3) \cdot \lambda(t_1, 3),\\
&\delta(t_1) = Z_1(1) \cdot \delta(t_1, 1) + Z_1(2) \cdot \delta(t_1, 2) + Z_1(3) \cdot \delta(t_1, 3),\\
&\lambda(t_2) = Z_2(1) \cdot \lambda(t_2, 1) + Z_2(2) \cdot \lambda(t_2, 2),\\
&\delta(t_2) = Z_2(1) \cdot \delta(t_2, 1) + Z_2(2) \cdot \delta(t_2, 2),
\end{aligned}\right\}\\
\left.\begin{aligned}
&Z_1(1) + Z_1(2) + Z_1(3) = 1,\\
&Z_2(1) + Z_2(2) = 1,
\end{aligned}\right\}\\
\left.\begin{aligned}
&i = 1, 2,\\
&\hat{\delta}(q_1^1) = 0, \hat{\delta}(q_1^2) = 0, \hat{\delta}(q_1^3) = 0,\\
&\hat{\delta}(q_2^1) = 0, \hat{\delta}(q_2^2) = 0,\\
&\hat{\delta}(p_1^1) = \delta(t_2), \hat{\delta}(p_1^2) = \delta(t_2),\\
&\hat{\delta}(p_2^1) = \delta(t_1), \hat{\delta}(p_2^2) = \delta(t_1),\\
&\hat{\delta}(p_{s_1}^1) = \delta(t_1), \hat{\delta}(p_{s_1}^2) = \delta(t_1),\\
&\hat{\delta}(p_{s_1}^3) = \delta(t_1),\\
&\hat{\delta}(p_{s_2}^1) = \delta(t_2), \hat{\delta}(p_{s_2}^2) = \delta(t_2),
\end{aligned}\right\}
\end{cases}
\end{aligned}
\tag{18}
$$

4.4. The Overall Genetic Algorithm

Now we are ready to introduce the overall genetic algorithm in our approach.

4.4.1. Selection

The purpose of the selection operation is to save chromosomes with higher fitness values in the parent population and to eliminate chromosomes with low fitness values. The selection rule we used here combines the classical roulette wheel selection [43] (also called the proportional selection, which ensures the diversity of genes inherited by the next generation) and the optimal retention [43] (that can effectively retain the optimal individual of each generation). In a roulette selection, the probability of each chromosome $\boldsymbol{Chm}$ to be selected is proportional to its fitness value $fitness(\boldsymbol{Chm})$. In generation i with population $\mathcal{N}_i$, the probability of selecting a chromosome $\boldsymbol{C}hm$ is given by:

$$Prob(\boldsymbol{Chm}_i) = \frac{fitness(\boldsymbol{Chm})}{\sum_{\boldsymbol{Chm}_j \in \mathcal{N}_i} fitness(\boldsymbol{Chm}_j)}. \tag{19}$$

Note that it may eliminate chromosomes with large fitness values during the roulette wheel selection operation. Therefore, we retain the chromosome with the highest fitness value through the optimal retention method before the roulette wheel selection operation.

4.4.2. Crossover

In the genetic algorithm, new chromosomes are generated through the crossover operation by exchanging gene segments from two parent chromosomes to find better offspring. In the literature, several crossover operations have been proposed, such as single-point crossover, multi-point crossover, and uniform crossover [43]. In this paper, we used a single-point crossover operation for the genetic algorithm.

4.4.3. Mutation

The mutation operation is to randomly—usually with a pre-set probability—perturb one or several genes of a chromosome. The mutation operation is of great significance in the genetic algorithm to maintain population diversity, prevent falling into the local optimal solution, and enrich the gene pool. The mutation operation uses site mutation, i.e., one or more genes of a chromosome are randomly selected and changed with a mutation probability. In this paper, we use the single-point mutation operation, i.e., at most one gene in each chromosome is mutated for each operation.

Note that during the crossover operation and mutation operation, a newly generated chromosome $\boldsymbol{Chm}$ may not always satisfy the constraints in Equation (15), i.e., the TWMG may not be live at the decoded marking $\boldsymbol{M_{Chm}}$, or the total cost of servers corresponding to $\boldsymbol{M_{Chm}}$ exceeds the budget. In such a case, the chromosome that leads to an infeasible solution is immediately discarded, and a fully randomized chromosome, which satisfies the constraint (15) is added to the pool.

The entire hybrid method that combines genetic heuristics with MILP is presented as Algorithm 2, whose inputs are: a TWMG model N, the budget R, the unit cost vector $\boldsymbol{\lambda}(t_i)$, the firing delay vector $\boldsymbol{\delta}(t_i)$, the population size $\mathcal{N}_p$ (i.e., $\mathcal{N}$), the maximum number of generations Ge, the crossover probability $\mathcal{P}_c$, and the mutation probability $\mathcal{P}_m$.

Algorithm 2 Genetic MILP method for performance optimization in TWMGs.

Input: A TWMG model N, $R \in \mathbb{R}$, $\boldsymbol{\lambda}(t_i) \in \mathbb{Q}^m_{\geq 0}$, $\boldsymbol{\delta}(t_i) \in \mathbb{Q}^m_{\geq 0}$, $\mathcal{N}_p \in \mathbb{N}$, $Ge \in \mathbb{N}$, $\mathcal{P}_c \in \mathbb{R}_{\geq 0}$, $\mathcal{P}_m \in \mathbb{R}_{\geq 0}$, $\mathcal{T}_0 \in \mathbb{R}_{\geq 0}$, $\mathcal{T}_e \in \mathbb{R}_{\geq 0}$, and $\varepsilon \in \mathbb{R}_{\geq 0}$;

Output: Server-type selection $\boldsymbol{Z} = [\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_m]^T$ and server quantity $\boldsymbol{\mathcal{M}} = [M(p_{s_1}), \ldots, M(p_{s_m})]^T$;

while $k \leq Ge$ **do**

for $j = 1, \ldots, \mathcal{N}_p$ **do**

Transform $\langle N, \boldsymbol{M}_{\boldsymbol{Chm}^k_j} \rangle$ into $\langle \hat{N}, \hat{\boldsymbol{M}}_{\boldsymbol{Chm}^k_j} \rangle$ according to Algorithm 1;

Solve MILPP (16) for $\langle \hat{N}, \hat{\boldsymbol{M}}_{\boldsymbol{Chm}^k_j} \rangle$;

Calculate $fitness(\boldsymbol{Chm}^k_j)$ according to Equation (17);

end for

Let $j^* = arg \max_{\boldsymbol{Chm}^k_j \in \mathcal{N}_k} fitness(\boldsymbol{Chm}^k_j)$;

Let $\boldsymbol{\mathcal{M}} := \boldsymbol{M}_{Chm^k_{j*}}$ be the server quantity and $\boldsymbol{Z} := (\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_m)$ be the server type selection corresponding to chromosome $\boldsymbol{Chm}^k_{j^*}$;

if $k + 1 \leq Ge$ **then**

$k := k + 1$;

Execute selection, crossover, and mutation to generate an offspring generation $\mathcal{N}_k = \{\boldsymbol{Chm}^k_1, \ldots, \boldsymbol{Chm}^k_{\mathcal{N}_p}\}$;

else

Output $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{Z}$

end if

end while

5. Illustrative Examples

In this section, the developed approach in Section 4 is illustrated by investigating the TWMG model discussed in Example 1 and a real flexible manufacturing system. The algorithms are implemented by MATLAB 2017 with YALMIP R20181012 subroutines on a computer installed Windows 10 operating system with CPU Intel Core i5 at 3.0 GHz and RAM 8 GB.

5.1. First Example

Example 3. *Let us continue Example 1 and assume that the upper bound on the cost of servers is $R = 100$. Parameters in the developed approach are set as the population size $\mathcal{N}_p = 10$, the crossover probability $\mathcal{P}_c = 0.7$, and the mutation probability $\mathcal{P}_m = 0.1$.*

In Table 2, we show the best solution (i.e., the minimal cycle time), the worst solution (i.e., the maximal cycle time), the average solution, and the average CPU time for the developed approach (i.e., genetic MILP) by testing a different maximum number of generations Ge. For each generation, we executed the developed approach ten times. An optimal/suboptimal solution is found as follows:

$$\boldsymbol{\mathcal{M}} = [2, 14]^T,\ \boldsymbol{Z}_1 = [0, 0, 1]^T,\ \boldsymbol{Z}_2 = [1, 0]^T,$$

which means that the optimal quantity of server of t_1 and t_2 are 2 and 14, respectively, while the third type of server of t_1 and the first type of server of t_2 are selected. By enumerating all possible server-type selections and server quantities, we found that the solutions obtained by the proposed approach are optimal. In addition, we also implemented the developed approach without MILP (i.e., genetic in Table 2), which means that both quantities and types of servers are generated by the genetic algorithm. It can be observed that the qualities of the solutions are improved by using the MILP we developed.

Table 2. Simulation results of Example 3.

Approaches / Obtained Results	Genetic MILP				Genetic			
Maximal number of generation Ge	1	2	5	15	1	2	5	15
Best solution	3.0	3.0	3.0	3.0	3.3	3.3	3.3	3.1
Worst solution	4.5	4.0	3.0	3.0	6.0	5.0	5.0	4.0
Average solution	3.7	3.2	3.0	3.0	4.4	4.2	3.7	3.7
Average CPU time [s]	7.5	11.0	20.5	59.4	3.1	4.6	9.1	24.2

5.2. Application to a Real Flexible Manufacturing System

In this subsection, we apply the developed approach to a hydraulic torque converter production line that is taken from [36]. The hydraulic torque converter is assembled with four different products as shown in Table 3. The production process of these products include 25 operations operated on 17 different machines (denoted by $\mathcal{MA}_1, \ldots, \mathcal{MA}_{17}$). We mention that the turbine is also assembled by two different products. Note that some machines are shared by one or more production lines, such as $\mathcal{MA}_1$, $\mathcal{MA}_3$, $\mathcal{MA}_4$, and $\mathcal{MA}_6$.

Table 3. Production process of the flexible manufacturing system.

Pump wheel	$\mathcal{MA}_1 \to \mathcal{MA}_2 \to \mathcal{MA}_3 \to \mathcal{MA}_4 \to \mathcal{MA}_5 \to \mathcal{MA}_6$
Turbine	$\mathcal{MA}_1 \to \mathcal{MA}_7 \to \mathcal{MA}_3 \to \mathcal{MA}_8 \to \mathcal{MA}_{10} \to \mathcal{MA}_{11} \to \mathcal{MA}_{12}$ $\mathcal{MA}_3 \to \mathcal{MA}_9 \to \mathcal{MA}_6 \to \mathcal{MA}_{10} \to \mathcal{MA}_{11} \to \mathcal{MA}_{12}$
Guide wheel	$\mathcal{MA}_1 \to \mathcal{MA}_4 \to \mathcal{MA}_{13}$
Cover wheel	$\mathcal{MA}_{14} \to \mathcal{MA}_4 \to \mathcal{MA}_{15} \to \mathcal{MA}_6$
Hydraulic torque converter	$\mathcal{MA}_{16} \to \mathcal{MA}_{17}$

The TWMG model is shown in Figure 4, which has 25 transitions and 54 places. It is composed of three different types of elementary circuits, i.e., process circuits that represent the manufacturing process, server circuits that represent the usage of machines, and mixed circuits that consist of both process circuits and server circuits. Tokens in places p_1, p_8, p_{13}, p_{20}, and p_{24} represent the raw materials for each production line that are assumed to be 1000, i.e., $M(p_j) = 1000$ ($j = 1, 8, 13, 20, 24$).

In Table 4, we present the unit costs of servers and the firing delays of the corresponding transitions. The budget is set to $R = 300$. Simulation results for different cases are presented in Table 5. Parameters in the developed approach are the same as in Example 3.

For all of the cases, we execute each approach ten times. Since the best solution for the maximum number of generations $Ge = 5$ is identical to the solution obtained for generation $Ge = 10$, there is no need to test more generations for the genetic MILP approach. An optimal/suboptimal solution is found by the developed approach with the cycle time $\chi(G) = 68$, and the server-type selection and server quantity are as follows:

$$\begin{aligned}
&\boldsymbol{\mathcal{M}} = [1,1,1,2,1,1,2,1,2,1,1,2,1,2,1,1,1,1,1,1,1,1,0,1,1]^T,\\
&Z_1(1) = 1, Z_2(1) = 1, Z_3(3) = 1, Z_4(1) = 1, Z_5(1) = 1,\\
&Z_6(5) = 1, Z_7(1) = 1, Z_8(1) = 1, Z_9(3) = 1, Z_{10}(1) = 1,\\
&Z_{11}(3) = 1, Z_{12}(1) = 1, Z_{13}(5) = 1, Z_{14}(1) = 1, Z_{15}(1) = 1,\\
&Z_{16}(2) = 1, Z_{17}(1) = 1, Z_{18}(1) = 1, Z_{19}(1) = 1, Z_{20}(2) = 1,\\
&Z_{21}(1) = 1, Z_{22}(1) = 1, Z_{23}(5) = 1, Z_{24}(3) = 1, Z_{25}(1) = 1.
\end{aligned}$$

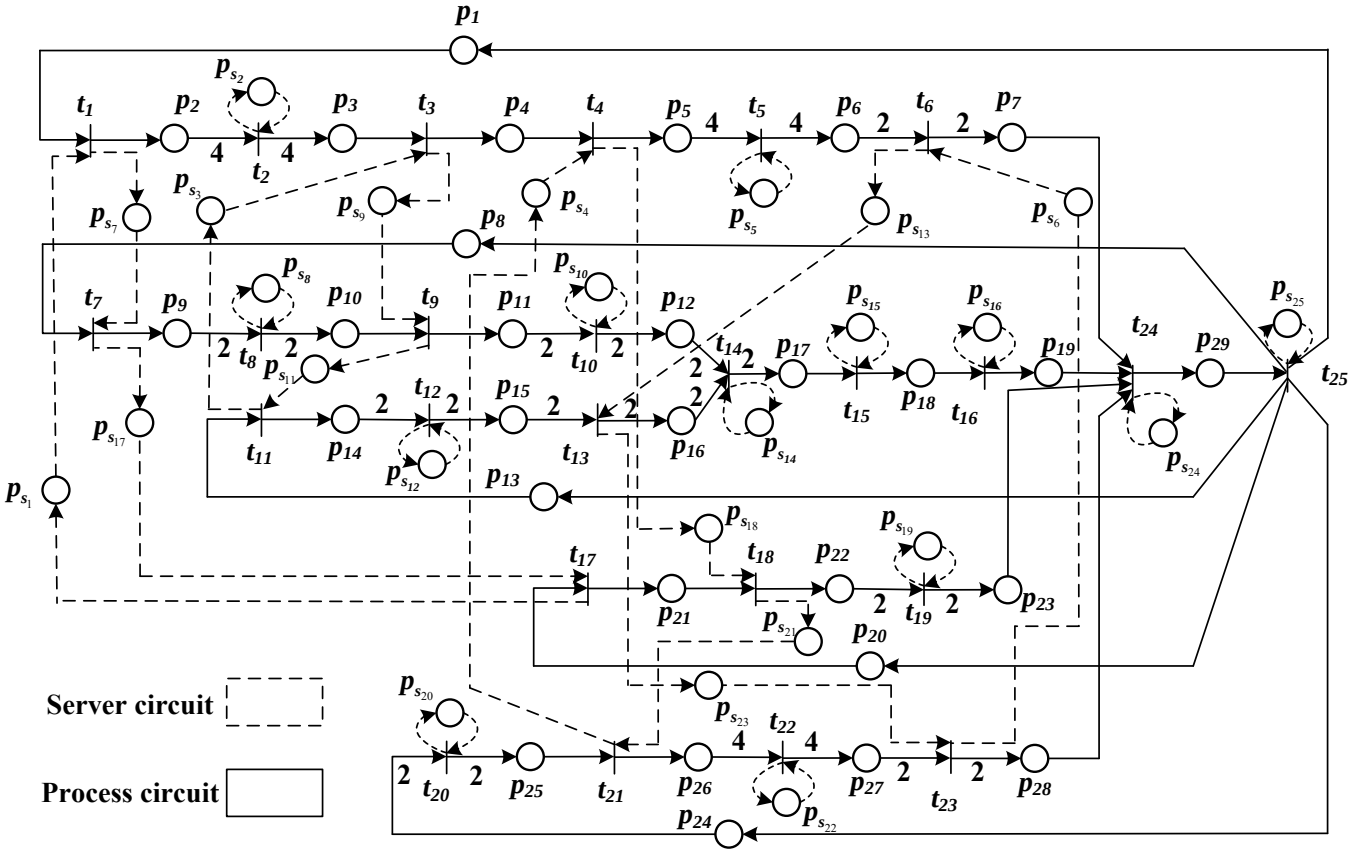

Figure 4. The TWMG model of the flexible manufacturing system.

Note that due to the shared usage of resources, some different operations (transitions) are executed by the same machine $\mathcal{MA}_i$ ($i = 1, 3, 4, 6$). Therefore, the server quantities of machine $\mathcal{MA}_i$ equals the sum of the server quantities of all the transitions corresponding to $\mathcal{MA}_i$; the server types of all the transitions corresponding to $\mathcal{MA}_i$ should be identical.

$$\begin{aligned}
&\text{Server quantity of } \mathcal{MA}_1 : M(p_{s_1}) + M(p_{s_7}) + M(p_{s_{17}}) = 4,\\
&\text{Server quantity of } \mathcal{MA}_3 : M(p_{s_3}) + M(p_{s_9}) + M(p_{s_{11}}) = 4,\\
&\text{Server quantity of } \mathcal{MA}_4 : M(p_{s_4}) + M(p_{s_{18}}) + M(p_{s_{21}}) = 4,\\
&\text{Server quantity of } \mathcal{MA}_6 : M(p_{s_6}) + M(p_{s_{13}}) + M(p_{s_{23}}) = 2.
\end{aligned}$$

From the simulation results, we can conclude that in the case of the same maximum number of generations, the cycle time (i.e., the objective function) obtained by the genetic MILP approach is smaller than that of the Genetic approach, i.e., the solution obtained by the genetic approach is improved by using the MILP.

We conclude this section by discussing the previous works. The approaches developed in [33–36] can provide optimal or near-optimal solutions under the condition that the types of resources are given. In this paper, we further extend these works by assuming that the types of resources are variables that also have to be optimized. Therefore, the existing approaches in [33–36] cannot provide a solution for the problem considered in this paper.

Table 4. Unit costs of the servers and the firing delays of each operation.

Transition t_i	Machine $\mathcal{MA}$	Server Type j	Unit Cost of Server $\lambda(t_i, j)$	Firing Delay $\delta(t_i, j)$ [s]	Transition t_i	Machine $\mathcal{MA}$	Server Type j	Unit Cost of Server $\lambda(t_i, j)$	Firing Delay $\delta(t_i, j)$ [s]
t_1	$\mathcal{MA}_1$	1	7	27	t_{13}	$\mathcal{MA}_6$	2	7	34
		2	15	19			3	8	29
		3	20	15			4	10	26
		4	23	13			5	11	20
t_2	$\mathcal{MA}_2$	1	4	25	t_{14}	$\mathcal{MA}_{10}$	1	10	45
		2	8	22			2	14	38
t_3	$\mathcal{MA}_3$	1	7	40	t_{15}	$\mathcal{MA}_{11}$	1	8	17
		2	8	37			2	12	13
		3	9	25	t_{16}	$\mathcal{MA}_{12}$	1	5	18
t_4	$\mathcal{MA}_4$	1	8	16			2	9	12
		2	9	15	t_{17}	$\mathcal{MA}_1$	1	9	15
		3	10	9			2	15	14
		4	12	6			3	20	10
t_5	$\mathcal{MA}_5$	1	3	29			4	23	6
		2	6	25	t_{18}	$\mathcal{MA}_4$	1	8	18
		3	8	20			2	9	16
t_6	$\mathcal{MA}_6$	1	6	38			3	10	16
		2	7	35			4	12	13
		3	8	32	t_{19}	$\mathcal{MA}_{13}$	1	13	24
		4	10	28			2	17	19
		5	11	25			3	20	13
t_7	$\mathcal{MA}_1$	1	9	26	t_{20}	$\mathcal{MA}_{14}$	1	7	38
		2	15	23			2	11	31
		3	20	18	t_{21}	$\mathcal{MA}_4$	1	8	25
		4	23	12			2	9	22
t_8	$\mathcal{MA}_7$	1	2	31			3	10	20
		2	5	26			4	12	14
		3	7	20	t_{22}	$\mathcal{MA}_{15}$	1	10	36
t_9	$\mathcal{MA}_3$	1	7	33			2	14	30
		2	8	25	t_{23}	$\mathcal{MA}_6$	1	6	37
		3	9	16			2	7	32
t_{10}	$\mathcal{MA}_8$	1	2	11			3	8	29
		2	7	8			4	10	24
t_{11}	$\mathcal{MA}_3$	1	7	26			5	11	20
		2	8	22	t_{24}	$\mathcal{MA}_{16}$	1	14	58
		3	9	10			2	21	42
t_{12}	$\mathcal{MA}_9$	1	5	47			3	30	20
		2	8	42	t_{25}	$\mathcal{MA}_{17}$	1	7	13
		3	13	35			2	8	10
t_{13}	$\mathcal{MA}_6$	1	6	35			3	12	6

Table 5. Simulation results of the developed approach for the flexible manufacturing system.

Approaches / Obtained Results	Genetic MILP				Genetic					
Maximal number of generation *Ge*	1	2	5	10	2	5	10	40	70	100
Best solution	76.0	72	68.0	68.0	130.0	102.0	102.0	90.0	90.0	90.0
Worst solution	84.0	77.3	76.0	76.0	212.0	204.0	180.0	168.0	130.0	130.0
Average solution	79.7	75.2	71.3	70.6	171.8	159.4	152.2	126.2	120.8	111.7
Average CPU time [s]	72.9	152.9	396.5	1059.3	4.7	12.7	28.8	142.8	433.3	843.8

6. Conclusions

In this paper, we study the performance optimization for a class of PNs to maximize the throughput of the system subject to a given budget, while both quantities and typesof servers are decision variables. A genetic algorithm combined with mixed-integer linear programming was originally developed to obtain a near-optimal solution. The developed approach is based on linear programming techniques and provides an efficient solution for solving the performance optimization problem where both quantities and types of resources are simultaneously optimized. Our future work will focus on exploring an analytical solution for the considered optimization problem. On the other hand, we will consider improving the efficiency of the developed approach by using some advanced hybrid methods [44].

Author Contributions: Conceptualization, Z.H. and W.S.; methodology, Z.H. and C.G.; software, W.S. and Z.M.; validation, N.R. and Z.M.; formal analysis, Z.H. and W.S.; investigation, Z.H. and W.S.; resources, Z.H.; data curation, not applicable; writing-original draft preparation, W.S.; writing—review and editing, Z.H. and Z.M.; visualization, C.G. and N.R.; supervision, Z.H.; project administration, Z.H.; funding acquisition, Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under grant nos. 61703321, 61803246, 61903119, and 62003201, the China Postdoctoral Science Foundation under grant no. 2019M663608, Shaanxi Provincial Natural Science Foundation under grant nos. 2022JM-323 and 2023-JC-YB-564, the Hebei Province Foundation for Returned Overseas Chinese Scholars under grant C20190319, and the Fundamental Research Funds for the Central Universities under grant JB210413.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors express their gratitude to every reviewer of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PN	Petri net
FMS	flexible manufacturing system
MILP	mixed-integer linear programming
TMG	timed marked graph
TWMG	timed weighted marked graph
MILPP	mixed-integer linear programming problem
WMG	weighted marked graph
MG	marked graph
TPN	timed Petri net
TTPN	transition-timed Petri net
PTPN	place-timed Petri net
PTMG	place-timed marked graph

References

1. He, Z.; Tang, B.; Luan, F. An improved African vulture optimization algorithm for dual-resource constrained multi-objective flexible job shop scheduling problems. *Sensors* **2023**, *23*, 90. [CrossRef]
2. Declerck, P. Optimization of the time durations by exploiting time margins in time interval models. *IEEE Trans. Control Syst. Technol.* **2022**, *30*, 755–766. [CrossRef]
3. Huang, B.; Zhou, M.; Lu, X. Scheduling of Resource Allocation Systems with Timed Petri Nets: A Survey. *ACM Comput. Surv.* **2023**, 1–28 . [CrossRef]
4. Benabid-Najjar, A. Periodic schedules for bounded timed weighted event graphs. *IEEE Trans. Autom. Control* **2012**, *57*, 1222–1232. [CrossRef]

5. Kim, H.; Lee, J.; Lee, T. Schedulability analysis for noncyclic operation of time-constrained cluster tools with time variation. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1409–1414. [CrossRef]
6. Huang , B.; Zhou, M. Symbolic Scheduling of Robotic Cellular Manufacturing Systems with Timed Petri Nets. *IEEE Trans. Autom. Sci. Eng.* **2022**, *30*, 1876–1887. [CrossRef]
7. Huang, B.; Zhou, M.; Abusorrah, A. Scheduling Robotic Cellular Manufacturing Systems with Timed Petri Net, A* Search and Admissible Heuristic Function. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 243–250. [CrossRef]
8. Liu, C. Formal modeling and discovery of multi-instance business processes: A cloud resource management case study. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 2151–2160. [CrossRef]
9. Zhao, Z.; Liu, S.; Zhou, M. Heuristic scheduling of batch production processes based on Petri nets and iterated greedy algorithms. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 25–261. [CrossRef]
10. You, D.; Wang, S.; Zhou, M. Supervisory control of Petri nets in the presence of replacement attacks. *IEEE Trans. Autom. Sci. Eng.* **2021**, *67*, 1466–1473. [CrossRef]
11. Millo, J.; Simone, R.D. Periodic scheduling of marked graphs using balanced binary words. *Theor. Comput. Sci.* **2012**, *458*, 113–130. [CrossRef]
12. Campos, J.; Chiola, G.; Colom, J. Properties and performance bounds for timed marked graphs. *IEEE Trans. Fundam. Theory Appl.* **1992**, *39*, 386–401. [CrossRef]
13. Baccelli, F.; Cohen, G.; Olsder, G. *Synchronization and Linearity: An Algebra for Discrete Event Systems*; Wiley: New York, NY, USA, 1992.
14. He, Z.; Ma, Z.; Tang, W. Performance safety enforcement in strongly connected timed event graphs. *Automatica* **2021**, *128*, 109605. [CrossRef]
15. Panayiotou, C.; Cassandras, C. Optimization of kanban-based manufacturing systems. *Automatica* **1999**, *35*, 1521–1533. [CrossRef]
16. Li, R.; Reveliotis, S. Performance optimization for a class of generalized stochastic Petri nets. *Discret. Event Dyn. Syst.* **2015**, *25*, 387–417. [CrossRef]
17. Rodriguez, R.; Julvez, J.; Merseguer, J. On the performance estimation and resource optimization in process Petri nets. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 1385–1398. [CrossRef]
18. Ma, Z.; Li, Z.; Giua, A. Marking estimation in a class of time labeled Petri nets. *IEEE Trans. Autom. Control* **2020**, *65*, 493–506. [CrossRef]
19. He, Z.; Li, Z.; Giua, A. Some remarks on State estimation and fault diagnosis of labeled time Petri net systems with unobservable transitions. *IEEE Trans. Autom. Control* **2019**, *64*, 5253–5259. [CrossRef]
20. Seatzu, C. Modeling, analysis, and control of automated manufacturing systems using Petri nets. In Proceedings of the 24th IEEE International Conference on Emerging Technologies and Factory Automation, Zaragoza, Spain, 10–13 September 2019; pp. 27–30.
21. Lafit, S.; Proth, J.; Xie, X. *Marking Optimization in Timed Event Graphs*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 281–300.
22. Giua, A.; Piccaluga, A.; Seatzu, C. Firing rate optimization of cyclic timed event graph. *Automatica* **2002**, *38*, 91–103. [CrossRef]
23. He, Z.; Liu, M.; Ran, N. Firing rate optimization of deterministic timed event graphs by server performance improvement. *IEEE Access* **2019**, *6*, 70866–70873. [CrossRef]
24. Teruel, E.; Chrzastowski-Wachtel, P.; Colom, J. On weighted T-Systems. *Appl. Theory Petri Nets* **1992**, *616*, 348–367.
25. Cottenceau, B.; Hardouin, L.; Boimond, J. Modeling and control of weight-balanced timed event graphs in dioids. *IEEE Trans. Autom. Control* **2014**, *59*, 1219–1231. [CrossRef]
26. Marchetti, O.; Munier, A. Complexity results for weighted timed event graphs. *Discrete Optim.* **2010**, *7*, 166–180. [CrossRef]
27. Munier, A. Régime asymptotique optimal d'un graphe d'événements temporisé généralisé: Application à un problème d'assemblage. *RAIPO-APII* **1992**, *27*, 487–513.
28. Nakamura, M.; Silva, D.M. Cycle time computation in deterministically timed weighted marked graphs. In Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factory Automation, Barcelona, Spain, 18–21 October 1999; Volume 2, pp. 1037–1046.
29. He, Z.; Ma, Z.; Li, Z. Parametric transformation of timed weighted marked graphs: Applications in optimal resource allocation. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 179–188. [CrossRef]
30. Kahouadji, H.; Hamaci, S.; Labadi, K. A new upper bound of cycle time in weighted marked graphs. In Proceedings of the International Conference on Control, Decision and Information Technologies (CoDIT), Hammamet, Tunisia, 6–8 May 2013; pp. 137–142. [CrossRef]
31. Benfekir, A.; Hamaci, S.; Boimond, J.L. Performance evaluation of nonlinear weighted T-system. *Int. J. Syst. Sci.* **2013**, *44*, 1948–1955. [CrossRef]
32. He, Z.; Li, Z.; Giua, A. Cycle time optimization of deterministic timed weighted marked graphs by transformation. *IEEE Trans. Control Syst. Technol.* **2017**, *25*, 1318–1330. [CrossRef]
33. He, Z.; Li, Z.; Giua, A. Performance optimization for timed weighted marked graphs under infinite server semantics. *IEEE Trans. Autom. Control* **2018**, *63*, 2573–2580. [CrossRef]
34. Sauer, N. Marking optimization of weighted marked graphs. *Discret. Event Dyn. Syst.* **2003**, *13*, 245–262. [CrossRef]
35. He, Z.; Li, Z.; Giua, A. Optimization of deterministic timed weighted marked graphs. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 1084–1095. [CrossRef]

36. He, Z.; Liu, M.; Ma, Z. An improved approach for marking optimization of timed weighted marked graphs. *Discret. Event Dyn. Syst.* **2019**, *29*, 127–143. [CrossRef]
37. Qudeiri, J.A.; Yamamoto, H.; Ramli, R. Genetic algorithm for buffer size and work station capacity in serial-parallel production lines. *Artif. Life Rob.* **2008**, *12*, 102–106. [CrossRef]
38. Nahas, N.; Nourelfath, M.; Gendreau, M. Selecting machines and buffers in unreliable assembly/disassembly manufacturing networks. *Int. J. Prod. Econ.* **2014**, *154*, 113–126. [CrossRef]
39. Liu, L.C.; Yan, C.B.; Li, J.S. Modeling, analysis, and improvement of batch-discrete manufacturing systems: A systems approach. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1567–1585. [CrossRef]
40. Zhang, W.; Wang, J.; Lin, Y. Integrated design and operation management for enterprise systems. *Enterp. Inf. Syst.* **2019**, *13*, 424–429. [CrossRef]
41. Wang, J. Charging information collection modeling and analysis of GPRS networks. *IEEE Trans. Syst. Man Cybern.* **2007**, *37*, 473–481. [CrossRef]
42. Schuppen, J.V.; Silva, M.; Seatzu, C. Control of discrete-event systems-Automata and Petri Net perspectives. *Lect. Notes Control Inf. Sci.* **2012**, *433*, 319–340.
43. Mirjalili, S. Genetic algorithm. *Evol. Algorithms Neural Netw.* **2019**, *780*, 43–55.
44. Bi, J.; Yuan, H.; Zhai, J. Self-adaptive bat algorithm with genetic operations. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 1284–1294. [CrossRef]