applied sciences

MDPI

Article

A Theoretical Foundation for Context-Aware Cyber-Physical Production Systems

Fu-Shiung Hsieh

Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan; fshsieh@cyut.edu.tw

Abstract: The complex workflows and interactions between heterogeneous entities in Cyber-Physical Production Systems (CPPS) call for the use of context-aware computing technology to operate effectively and meet the order requirements in a timely manner. In addition to the objective to meet the order due date, due to resource contention between production processes, CPPS may enter undesirable states. In undesirable states, all or part of the production activities are in waiting states or blocked situation due to improper allocation of resources. The capability to meet the order due date and prevent the system from entering an undesirable state poses challenges in the development of context-aware computing applications for CPPS. In this study, we formulate two situation awareness problems, including a Deadline Awareness Problem and a Future States Awareness Problem to address the above issues. In our previous study, we found that Discrete Timed Petri Nets provide an effective tool to model and analyze CPPS. In this paper, we present a relevant theory to support the operation of CPPS by extending the Discrete Timed Petri Nets to lay a foundation for developing context-aware applications of CPPS with deadline awareness and future states awareness capabilities. We illustrate the theory developed in this study by an example and conduct experiments to verify the computational feasibility of the proposed method.

Keywords: context-aware application; cyber-physical production systems; situation-aware; deadline; discrete timed; Petri net

Citation: Hsieh, F.-S. A Theoretical Foundation for Context-Aware Cyber-Physical Production Systems. *Appl. Sci.* **2022**, *12*, 5129. https://doi.org/10.3390/app12105129

Academic Editors: Andrea Prati and Hui Yu

Received: 5 March 2022
Accepted: 16 May 2022
Published: 19 May 2022

1. Introduction

The term Cyber-Physical Systems (CPS) was introduced in [1] to refer to the paradigm that supervises, manages and controls entities in the physical world through the use of cyber world models. The widespread use of the Internet of things (IoT) as well as relevant Information and Communication Technologies (ICTs) have encouraged the adoption of the CPS paradigm by manufacturers [2,3]. The CPS for manufacturing systems are referred to as Cyber-Physical Production Systems (CPPS) [4]. A five-level classification of CPPS according to the level of integration was discussed in [5]. These five levels include the Connection level, Conversion level, Cyber level, Cognition level and Configuration level. This classification was extended to CPPS applications in [6].

The Supervisory Control and Data Acquisition (SCADA) system in CPPS can access the real-time data from sensors based on the IoT infrastructure. The SCADA system in CPPS may provide the features of Connection level, Conversion level, Cyber level, and Cognition level services in CPPS, depending on the implementation. Rich accessible real-time data from sensors have several implications for managers and workers and open the door for the creation of many potentially innovative context-aware applications [2]. For example, CPPS enable managers to monitor and supervise the status of production activities and operations at their fingertips. For workers on the shop floor in CPPS, real-time data provide feedback information to guide their operations. The widely accessible real-time data enable and facilitate the development of context-aware applications for CPPS. Context-aware applications for CPPS make it easier for entities on the shop floor in CPPS to acquire

contextual information at the point of need. Therefore, the development of context-aware applications for CPPS is an important research area in the literature [2–4].

In ICTs, context awareness is a property with which computers or applications can both sense and react based on the environment. Context awareness refers to the capability to consider the situation of entities [7]. Therefore, a context-aware application should be aware of the situation of entities in the system. Situation awareness refers to the perception of an environment and relevant events to comprehend the current situation and predict its future status [8]. CPPS aims to fulfill the order requirements by meeting the due date. However, due to the complex production processes and interactions between entities in CPPS, CPPS may enter undesirable states in which all or part of the production activities are in waiting states or blocked due to the resources not being allocated properly. Therefore, situation awareness in CPPS includes two properties: deadline awareness and future states awareness. These two situation awareness properties should be considered in the development of an effective context-aware application for CPPS.

Existing studies on context-aware applications for CPS primarily focus on the generation of contextual information for entities in CPS [9]. The challenge of attaining situation awareness in CPPS is not addressed. As a result, although the real-time data are readily available to managers and workers in CPPS, managers and workers cannot respond to the environment efficiently without an effective decision support tool. As a result, managers and workers in CPPS cannot make the right decisions and perform the operations as needed in the CPPS to meet customers' order requirements. Motivated by this need, this paper attempts to develop the theory to analyze the situation of CPPS in terms of deadline awareness and future states awareness to support decision makers and generate contextual information in CPPS. The goal of this paper is to realize the situation awareness of context-aware applications for CPPS. The complexity of developing an effective context-aware application depends on the complexity of identifying the "context" and analyzing the "situation" of the entities in a system. For CPPS, "context" is equivalent to the current states of entities, whereas "situation" is relevant to the evolution of future states. The complexity of analyzing the "situation" is much greater than the complexity of identifying the "context" of entities in CPPS. Therefore, we focus on the theory of analyzing the "situation" of CPPS to pave the way for the development of context-aware and situation-aware applications for CPPS.

In scientific studies, researchers tend to propose a new solution method based on the knowledge from the relevant literature instead of reinventing the wheel. Reinventing the wheel is only necessary when there is no one that can meet the requirements. The strategy of borrowing existing models and tailoring them to fit the requirements of studies is common in scientific research. In this paper, we adopt a class of Petri nets as the model of CPPS. Petri nets were invented by Carl Adam Petri to describe chemical processes [10]. Ramchandani borrowed the Petri net model invented by Carl Adam Petri and extended it to a timed Petri net [11] by considering the time factor (which was not considered in the original Petri net model). In a timed Petri net, the time it takes to fire a transition is described by a delay. Another researcher, Merlin, also borrowed the Petri net model invented by Carl Adam Petri and extended it to a time Petri net [12] by considering the time factor in the Petri net in a different way. In a time Petri net, a transition can only be fired with a specific time interval since it is enabled. In terms of the model used in this paper, we borrow characteristics from the existing discrete timed Petri net model in the literature from [13]. The way to discretize time is similar to the one used by Lefebvre and Daoui in [14]. In terms of the solution methodology in this paper, we propose a new solution approach without reinventing the wheel. The model used in this paper extends the class of deterministic timed Petri nets in [13] by allowing more complex resource sharing patterns in CPPS. In this paper, we define the problems relevant to the situation awareness of CPPS and develop the theory needed to lay a foundation for achieving situation awareness of CPPS, and for developing deadline-aware and future states-aware applications in CPPS.

We illustrate the computational feasibility of our approach by presenting our computational experience based on experiments of the proposed method to analyze the situation of CPPS.

The contributions of this paper are summarized as follows: (1) the conceptualization of deadline awareness and future states awareness of CPPS, (2) a formulation of the deadline awareness problem and future states awareness problem of CPPS, (3) an optimization approach to the deadline awareness problem and future states awareness problem of CPPS and (4) a computational complexity analysis and verification based on the results. The problem addressed in this paper is obviously different from the existing studies of context-aware applications for CPPS. Several concepts are proposed in this paper, including deadline awareness and future states awareness based on a class of discrete timed Petri net models. The time factor and deadline awareness issues addressed in this study are not studied in the context-aware workflow management method proposed in [15,16]. The discrete timed Petri net model used in this paper is also different from the temporal reasoning time interval model in [16]. This study extends the method proposed in [9,13] to generate contextual information by exploiting the discrete timed Petri net models for CPPS.

In terms of the use of Petri nets in context-aware applications, in this paper, we adopt Petri nets as a tool to generate control policy instead of using Petri nets as models for the evaluation of performance [17]. This paper focuses on achieving the deadline awareness and future states awareness of CPPS based on a class of discrete timed Petri nets. This is different from the study [18] that uses Petri nets in the modeling of a context-aware Human-Computer System. The research problem addressed in this paper concerns decision making in CPPS which is different from the literature on Petri net decision-making modelling for cloud service composition in [19].

The remainder of this paper is organized as follows. A review of the related literature is provided in Section 2. We state the situation awareness problem of CPPS in Section 3 and present the problem formulation in Section 4. Our approach to solving the situation awareness problem is presented in Section 5. Results obtained based on experiments of the proposed method and discussion are reported in Sections 6 and 7, respectively. We conclude this paper in Section 8.

2. Literature Review

Context-aware computing technologies refer to the use of sensor, information and communication technologies to sense an environment and adapt the behaviors of applications and devices accordingly, in order to provide the relevant information at the point of need for users [20]. To understand context-aware computing technologies, the meaning of "context" should be clarified. There are many definitions of the word "context" as mentioned by Alegre et al. in [21]. However, there is no consensus on the definition of the word "context" [22]. Despite this fact, Dey gave a definition of the word "context" as "any information that can be used to characterize the situation of an entity" [23]. This definition is easy to understand and has been widely accepted in the literature. Context-aware computing relies on the use of contexts. With the advancement of sensor, information and communication technologies, more and more contexts can be accessed and used in the development of context-aware applications. These contexts include time context, location context, environment context, device context and user context, among others [24]. The prevailing Internet of Things have created innovative research issues and application areas in context-aware computing. For example, the study of [25] presents a survey of context awareness issues on movement and activity tracking, and localization and environmental sensing for mobile platforms. Recent development of context-aware computing for guidance support in disaster evacuation can be found in the survey paper [26]. The application of context-aware computing technologies in recommendation systems has been a hot area of research in recent years. The study [27] provides a review of the state-of-the-art techniques for context-aware recommendation systems. With the availability of real-time sensor data, this enables the use of time context and environmental context in the development of context-aware scheduling applications. Hence, context-aware scheduling is an important

research area [28]. Recently, a study was conducted on the methodology to support the realization of emerging context-aware social-technical applications from conception, design, and development, through to evolution, based on an activity theory-based approach [29].

The design of context-aware applications must consider the factors of time, location and activities of users. Therefore, time, locations and activities are the basic contextual elements in context-aware computing technologies. A concept directly relevant to context-aware computing is awareness. Awareness refers to the state of being conscious of something. It also refers to the ability to perceive, feel, or be cognizant of events [30]. Despite extensive studies on context-aware computing in the literature, there are only a few studies that are relevant to the situation awareness properties of CPS. These include the study of [9] on context-aware CPS and the study of the impact of resource failures on the operation of CPS [10]. Although CPPS can be viewed as a special class of CPS, some of the characteristics of CPPS are different from CPS. The way in which resources are shared among different workflows in CPPS is different from that of CPS. This calls for a study on the situation awareness of CPPS. In the context of CPPS, situation awareness includes two elements: deadline awareness and future states awareness. These two situation awareness properties should be considered in the development of an effective context-aware application for CPPS. In this paper, we develop a theory to lay the foundation for situation awareness of CPPS and pave the way for deadline-aware and future states-aware applications in CPPS.

To achieve these goals, the characteristics of CPPS should be studied first. The review paper by Napoleone et al. provides an in-depth survey of state-of-the-art CPPS literature [31]. In [31], several characteristics of CPPS were highlighted to develop a solution methodology. Among these characteristics, the ability to deal with complexity and heterogeneity in CPPS is essential. To deal with complexity and heterogeneity in CPPS, proper tools and methods should be used. In [32], Harrison et al. reviewed the engineering approaches and tools available for CPPS. Selection of a proper modeling language is needed to handle complexity and heterogeneity in CPPS.

To cope with the changing requirements in the business environment of CPPS, the concept of Model Driven Architecture (MDA) was proposed [33]. The MDA approach to the development of software systems focuses on platform-independent models instead of the platform dependent models of an application. A generic transformation procedure is then applied to transform platform-independent models into platform dependent models for the target application. The Unified Modeling Language (UML) and the Systems Modeling Language (SysML) are two popular modeling languages that can be used with the MDA approach. UML is a widely accepted and general-purpose modeling language in software engineering [34]. As an industrial standard, UML provides a standard way to visualize the design of software systems. In [35], Thramboulidis proposed a model driven approach for control and automation software based on UML. SysML is a general-purpose architecture modeling language for Systems Engineering applications [36]. The application of SysML to support embedded systems engineering can be found in [37]. In [38], a model and simulation for mechatronic systems with SysML was studied. Despite the powerful modeling tools provided by UML and SysML, they lack the support to address the situation awareness issues of systems. To analyze the situation awareness of systems, a formal modeling language that supports the analysis of system dynamics should be used.

Petri nets and different variants of Petri nets are graphically oriented and formal modeling languages that have been used for the simulation, verification and analysis of software systems. According to Wikipedia, Petri nets were invented to describe chemical processes in August 1939 by Carl Adam Petri [39]. In the literature, the Petri net model was documented in the dissertation of Carl Adam Petri in 1962. Since its invention, the Petri net model has been widely applied in different problem domains, including the computer, communication manufacturing and chemical sectors [40]. The adoption of Petri nets as the platform-independent models in MDA has been studied in [15]. In [15], the authors adopted untimed Petri nets as the platform-independent models to develop context-aware workflow systems. However, the time factor is not considered in [15]. In

the literature, many variants of Petri net models considering time semantics have been proposed to deal with timing issues in real world systems. The paper [41] by Zuberek surveys several variants of timed Petri nets proposed in the literature. There are different ways to model the time factor in timed Petri nets. For example, Zuberek introduced a class of deterministic timed Petri nets by specifying a fixed time interval for each transition [42]. In [43], Sifaki proposed a class of timed Petri nets by specifying a fixed time interval for each place in [38] to evaluate performance. Tokens are considered to be unavailable during the specified time interval of the corresponding place. Another way to model the stochastic nature of firing time is to assign a random variable to represent the firing time of each transition. This class of timed Petri nets are called Stochastic Petri Nets (SPN) [44]. In a continuous time SPN, the transitions are fired based on transition rates and the firing time is exponentially distributed. In a discrete time SPN, the transitions are fired based on the specified conditional probabilities and the firing time is geometrically distributed [45]. SPNs are primarily used in performance evaluation whereas deterministic timed Petri nets are used for scheduling or the control of systems. In this paper, we adopt a class of deterministic timed Petri nets as the model of CPPS to formulate the Deadline Awareness Problem and Future States Awareness Problem.

In this study, we consider a class of deterministic timed Petri nets by borrowing the timed Petri nets that were invented by Ramchandani [11] and tailoring them properly to Discrete Timed Petri Net (DTPN) models based on the discretization of time. The way to discretize time is similar to the one used by Lefebvre and Daoui in [14]. The model used in this paper extends the class of Petri nets in [13] by allowing more complex resource sharing patterns in CPPS. We propose a transformation method to solve the problem. We conduct experiments to study the computational feasibility of the proposed method.

3. Situation Awareness Problem in CPPS

Due to the complex workflows and heterogeneous resources involved in the operation of CPPS, identification of the "situation" of CPPS is a challenging problem as "situation" is characterized by evolution of the future states of the system. To concretely characterize the "situation" of CPPS, we first clarify the objectives of CPPS and factors that may impact the achievement of these objectives. Based on the objectives of CPPS and factors that may impact on the achievement of the objectives we can state the situation awareness problem for CPPS.

CPPS aims to produce products to meet the order requirements of customers. Therefore, the objectives of CPPS are to fulfill order requirements. Each order is specified by product demand and the associated deadline for delivering the requested products. Therefore, the "situation" of CPPS is often linked to whether the deadline of an order can be met. In this paper, we introduce the term "deadline awareness" to refer to the situation about whether the deadline of an order can be met.

In CPPS, due to resource sharing and contention between production processes, the systems may enter undesirable states in which all or part of the production activities are in a waiting state or blocked, due to the resources not being allocated properly. To characterize whether a CPPS can avoid such undesirable states, we introduce another term ("future states awareness") to refer to the situation about whether undesirable states can be avoided and the target state can be reached.

Based on the terms defined above, we state the following situation awareness problems.

Deadline Awareness Problem: Given a CPPS and an order, determine whether the order can be fulfilled by the deadline.

Future States Awareness Problem: Given a CPPS and an order, determine whether undesirable states can be avoided and the target state can be reached.

To address the problems stated above, we need to formulate these problems formally. We introduce proper models for CPPS first. We then formulate the above problems based on the models for CPPS. Finally, we develop the theory for solving these problems.

In order to analyze the properties of CPPS, a suitable modeling tool must be used to construct the model of CPPS. Several requirements of modeling tools must be satisfied. These requirements include the capabilities to model the operations, events and timing of the entities in CPPS. The modeling tool selected to model CPPS should be able to capture interactions between entities such as the synchronization of resources and operations, and concurrent and asynchronous events in CPPS. In particular, the modeling tool should be easy to learn and use. In addition, there must be analysis methods or theory accompanied with or developed for the selected modeling tool to facilitate analysis of the properties of CPPS. In computer science, several tools have been proposed and used to build models for systems. Among these modeling tools, Petri nets are a graphical modeling language that satisfy the requirements discussed above.

Considering time in Petri nets poses challenges for the analysis of Petri nets with time semantics in terms of time and space complexity. Many researchers contributed a lot by proposing more efficient methods to tackle the complexity issue. Different methods of analyzing time in time Petri nets or timed Petri nets have been proposed in the literature. In the early work [46] of Berthomieu and Menasche, the concept of State Class was first proposed in conjunction with an algorithm for the enumeration of State Classes for the analysis of Merlin's time Petri nets. A State Class graph preserves markings and traces with finite representations of states. It enables the analysis of time Petri nets. The concept of State Class was also used in the work [47] by Berthomieu and Diaz to verify time dependent systems. Although the State Class graph is suitable for reachability analysis, the branching structure of the state graph is not preserved. Berthomieu and Vernadat proposed the graph of atomic classes which preserves branching structure [48]. In the work [49], a Timed Aggregate Graph was proposed by Klai et al. to abstract the reachability state space of a time Petri net. In the Timed Aggregate Graph, the nodes are referred to as aggregates which represent grouped sets of states with associated time information encoded inside. Klai et al. illustrated the advantage of the Timed Aggregate Graph by experimental results. As the approaches were proposed primarily for model checking and the verification of systems, Lefebvre proposed a Timed Extended Reachability Graph to encode markings and temporal constraints, as well as the earliest firing policy for control and scheduling issues [50]. Lefebvre proposed an Approximated Timed Reachability Graph in [51] to reduce the complexity of the Timed Extended Reachability Graph by aggregating the states in which the markings are equal and the temporal constraints are not far from each other. A common property of the methods mentioned above is the need to explore state space or reduced state space by the aggregation of states to reduce complexity. However, all these methods suffer from state explosion problems as the complexity for constructing or exploring the variants of reachability graphs (State Class graphs or Timed Extended Reachability Graphs) grows exponentially with the problem size. This motivates us to develop a novel method for a subclass of nets without constructing or exploring any variants of reachability graphs.

In the next section, we introduce the way to construct models for CPPS based on a class of Petri nets. We then formulate the problems stated in the previous section based on the models constructed for CPPS.

4. CPPS Models

In this paper, to study the Deadline Awareness Problem and the Future States Awareness Problem of CPPS, we adopt a variant of timed Petri nets called discrete timed Petri nets (DTPN) to construct models for CPPS. We follow a bottom-up approach to building models for CPPS by constructing individual models of the entities in CPPS first and then constructing the overall model for CPPS, taking into account the interactions between resources and operations in the workflows. The models of individual entities in CPPS are divided into two categories: task subnets and resource subnets. To present the modeling of CPPS, we list the symbols and notations used in this paper in Table 1.

Table 1. Symbols and notations.

Symbol/Variable	Meaning
J	The total number of different task subnets
$\boldsymbol{J}$	The set of indices of task subnets, $\boldsymbol{J}= \{1,2,3,...,J\}$
j	A type of task subnet, $j \in \boldsymbol{J}$
Π	The number of periods in the time horizon
τ	The index of a period, $\tau \in \{1,2,...,\Pi\}$
D_j	The requested quantity of the products (demand) to be produced by type-j task subnets, $j \in \boldsymbol{J} = \{1,2,3,...,J\}$
Φ	The order deadline
GJ_j	The model of type-j task subnet, $GJ_j = (P_j, T_j, F_j, m_{j0}, \mu_j)$
N_j	Total operations in GJ_j
R	Total resource types
$\boldsymbol{R}$	The set of indices of different types of resources, $\boldsymbol{R} = \{1,2,3,...,R\}$
r	The index of a resource type, $r \in \boldsymbol{R} = \{1,2,3,...,R\}$
k	The k-th operation performed by a resource
$\|$	A merging operator to merge two or more Petri nets
GR_r	A model of type-r resource subnet, $GR_r = (P_r, T_r, F_r, m_{r0}, \mu_r)$
C_{rt}	The nominal capacity of type-r resources in period t; C_{rt} is set to $m_{r0}(r)$ for all t
G	A CPPS model, $G = (P, T, F, m_0, \mu) = GJ \| GR$, where $GJ = \mathop{\Vert}_{j \in \boldsymbol{J}} GJ_j$ and $GR = \mathop{\Vert}_{r \in \boldsymbol{R}} GR_r$
m	A marking of G
$STN_j(V_j, A_j)$	The spatial–temporal network of type-j task subnet, $j \in \boldsymbol{J}= \{1,2,3,...,J\}$
$JSTN(V, A)$	A Joint Spatial–Temporal network
$V(v)$	The set of nodes in V directed connected to v
$V_j(s)$	The set of nodes in V_j directed connected to s
$V_j(e_j)$	The set of nodes in V_j directed connected to e_j
a	An arc in $STN(V, A)$, where $a \in A$ or an arc in $STN_j(V_j, A_j)$, where $a \in A_j$.
$a(v,w)$, (v,w)	An arc from node v to node w is represented explicitly by $a(v,w)$. The arc $a(v,w)$ is abbreviated as (v,w) or a whenever it is clear from the context
t_a	The transition associated with arc a
f	A solution of the Nominal Optimization Problem (NOP)
$f(a)$, $f(a(v,w))f(v,w)$	$f(a(v,w))$ represents the value of the solution f to NOP on the arc $a= a(v,w)= (v,w)$ of $STN(V, A)$. $f(a(v,w))$ is abbreviated as $f(v,w)$ or $f(a)$ with $a= (v,w)$ whenever it is clear from the context
$g(\tau)$	The earliness penalty function for $\tau \leq \Phi$
$h(\tau)$	The lateness penalty function for $\tau > \Phi$

A CPPS consists of different types of entities, including resources and tasks. We propose task subnets and resource subnets for different types of resources and tasks. Task subnets and resource subnets in CPPS are described by a class of discrete timed Petri nets (DTPN) [9]. Resources and tasks in a system are represented by tokens in a DTPN. The number of tokens in each place can be represented by a vector called a marking. An event in a system is called a transition in a DTPN. The occurrence of an event changes the marking of a DTPN. A DTPN evolves from the initial marking due to the firing of transitions. In the DTPN model, we assume the time horizon is divided into Π periods and the duration of each period is δ. The choice of δ depends on its application. Note that the method proposed in this study works regardless of the choice of δ so long as $\delta > 0$. The value of the firing time of a transition is described by $\mu\delta$, where $\mu \in Z$ and Z is the set of nonnegative integers. Note that continuous time Petri nets are limiting cases of the discrete time Petri nets as $\delta \to 0$.

A formal definition of DTPN is as follows.

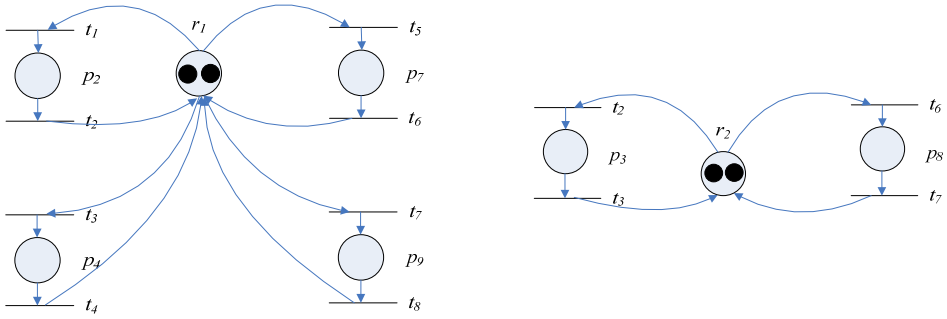
Definition 1. *We assume that the time horizon is divided into* Π *periods and the duration of each time period is* δ. *A DTPN is a five-tuple* $G = (P, T, F, m_0, \mu)$ *which consists of a set of places:* P, *to describe the states, a set of transitions;* T, *to represent events, flow relation;* $F \subseteq (P \times T) \cup (T \times P)$, *to characterize the connection between places and transitions, an initial marking;* m_0, *to describe*

the initial state and the firing time of a transition $t \in T$ is $\mu(t)\delta$, where $\mu : T \to Z$, Z is the set of nonnegative integers and δ is real.

To describe the dynamics of a DTPN, we need the following definition about transition firing.

Definition 2. *We use $^\bullet t$ to denote the set of input places of transition t and use $t^\bullet$ to denote the set of output places of t. If $m(p) \geq F(p,t) \forall p \in {}^\bullet t$, transition t is enabled under marking m. An enabled transition t can be fired. A transition t selected for firing in period τ under a control action will be fired at the beginning of period τ and the firing will be completed at the end of the designated period $\tau + \mu(t) - 1$. After firing an enabled transition t, the number of tokens in each input place in $^\bullet t$ will be reduced by one and the number of tokens in each output place in $t^\bullet$ will be increased by one.*

Figure 1a is an example of DTPN, where $^\bullet t_1 = \{r_1\}$ and $t_1^\bullet = \{p_2\}$.



(**a**) (**b**)

Figure 1. Examples of GR_r, where $r \in \boldsymbol{R} = \{r_1, r_2\}$: (**a**) GR_{r_1} (**b**) GR_{r_2}.

Definition 3. *A resource subnet GR_r for a type-r resource is a discrete timed Petri net $GR_r = (P_r, T_r, F_r, m_{r0}, \mu_r)$ that consists of a number of activities of type-r resources, where $r \in \boldsymbol{R}$ and the function μ_r is used to specify the firing time of each transition in T_r. Each activity is represented by a circuit [36] in which a number of transitions and places forms a closed loop. A resource subnet GR_r has an idle state place denoted by p_r. The set of all idle state places of resources is denoted by $P_o = \{p_r, r \in \boldsymbol{R} = \{1, 2, ..., R\}\}$. Without loss of generality, we will also refer to the type-r resource by $r \in \boldsymbol{R}$ whenever it is clear from the context.*

Figure 1a,b show two resource subnets, GR_{r_1} and GR_{r_2}. There are four circuits, $r_1 t_1 p_2 t_2$, $r_1 t_3 p_4 t_4$, $r_1 t_5 p_7 t_6$ and $r_1 t_7 p_9 t_8$ in GR_{r_1}. There are two circuits, $r_2 t_2 p_3 t_3$ and $r_2 t_6 p_8 t_7$ in GR_{r_2}.

Definition 4. *The capacity of type-r resources in period τ is denoted as $C_{r\tau}$, where $C_{r\tau} = m_{r0}(p_r)$ for all $\tau \in \{1, 2, ..., \Pi\}$, where Π is the time horizon.*

A CPPS may process different types of tasks. Let $\boldsymbol{J}$ denote the set of task types. Each task consists of a number of operations. Each operation can be specified by transitions. A type of task is described by a task subnet. Therefore, the task subnet considered in this study consists of a sequence of transitions. Let T_j denote the set of transitions in a task subnet. A transition may involve multiple resources.

Definition 5. *A type j task subnet $GJ_j = (P_j, T_j, F_j, m_{j0}, \mu_j)$, $j \in \boldsymbol{J}$, is a discrete timed Petri net which consists of a sequence $t_{j0}, t_{j1}, t_{j2}, ..., t_{jN_j}, t_{jN_j+1}$ of $N_j + 2$ transitions in the set T_j and a sequence $p_{j0}, p_{j1}, p_{j2}, p_{j3}, \ldots, p_{jN_j}$ of $N_j + 1$ places in P_j, where $\mu_j : T_j \to Z$ is a function that specifies the lower bound of the firing time of each transition $t \in T_j$. The first transition t_{j0} is a fictitious transition that represents the release of a task. The transition t_{jN_j} represents completion of*

the final operation of the task. The last transition t_{jN_j+1} is a fictitious transition that represents the removal of a completed task.

Note that the set of transitions in $T_j \backslash \left\{ t_{j0}, t_{jN_j} t_{jN_j+1} \right\}$ requires allocation of resources, i.e., $|{}^\bullet t \cap P_o| \geq 1 \, \forall t \in T_j \backslash \left\{ t_{j0}, t_{jN_j} t_{jN_j+1} \right\}$ and $|{}^\bullet t \cap P_o| = 0 \, \forall t \in \left\{ t_{j0}, t_{jN_j}, t_{jN_j+1} \right\}$. A transition $t \in T_j$ involving multiple resources is characterized by $|{}^\bullet t \cap P_r| \geq 2$. The number of resources required to fire transition t, where $t \in T_j$, is denoted by $R_{jt}(r) \forall r \in \boldsymbol{R}$.

Figure 2a,b show two task subnets, GJ_1 and GJ_2.

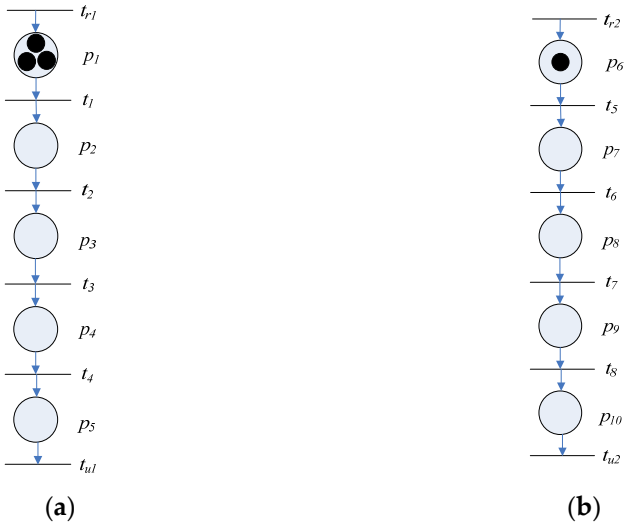


Figure 2. Examples of GJ_j, where $j \in \boldsymbol{J} = \{1, 2\}$: (**a**) GJ_1; (**b**) GJ_2.

To convey the idea to construct the model, we introduce a composition operator " $\|$" to merge two or more DTPN models through common transitions, places and arcs. There are many composition or synthesis methods in the literature (e.g., [13,52–55]). These include composition by merging transitions or places. Several operations that preserve the liveness property of the composed nets have been proposed in [52,53]. These operations include the fusion of a series of places or transitions and parallel places or transitions, and the elimination of self-loop places or transitions. Some of these studies focus on the synthesis of Petri nets for manufacturing systems (e.g., [13,55]). For this paper, the composition operation is used to capture the synchronization of resources and workflows. Therefore, the composition operation used in this paper is the same as the one used in [13]. It is defined as follows.

Definition 6 ([13]). *Given two discrete timed PNs, $G_1 = (P_1, T_1, F_1, m_{10}, \mu_1)$ and $G_2 = (P_2, T_2, F_2, m_{20}, \mu_2)$, the operator "$\|$" to combine G_1 and G_2 is defined as follows:*

$G_1 \| G_2 = (P, T, F, W, m_0, \mu)$, *where* $P = P_1 \cup P_2$, $T = T_1 \cup T_2$,

$$F(p,t) = \begin{cases} F_1(p,t) \, if \, p \in P_1 \, and \, t \in T_1 \\ F_2(p,t) \, if \, p \in P_2 \, and \, t \in T_2 \end{cases}, \; F(t,p) = \begin{cases} F_1(t,p) \, if \, p \in P_1 \, and \, t \in T_1 \\ F_2(t,p) \, if \, p \in P_2 \, and \, t \in T_2 \end{cases},$$

$$m_0(p) = \begin{cases} m_{10}(p) \, if \, p \in P_1 \\ m_{20}(p) \, if \, p \in P_2 \end{cases} \; and \; \mu(t) = \begin{cases} \max(\mu_1(t), \mu_2(t)) \, if \, t \in T_1 \cap T_2 \\ \mu_1(t), \, if \, t \in T_1 \backslash T_2 \\ \mu_2(t), \, if \, t \in T_2 \backslash T_1 \end{cases}.$$

Although the composition operation is the same as the one in [13], the properties and structure of the resulting models obtained by the composition operation are different from the ones obtained in [13]. The DTPN models considered in this paper allow more complex interactions patterns between resources and tasks. Hence, the liveness property of the composed DTPN models may not be preserved (This point is discussed in Section 4 of this

paper). This is the reason why a control policy is needed to maintain the liveness property of the composed DTPN models.

Definition 7. *A nominal model is described by a discrete timed Petri net* $G = (P, T, F, m_0, \mu) = GJ \| GR$, *where* $GJ = \underset{j \in \boldsymbol{J}}{\|} GJ_j$ *and* $GR = \underset{r \in \boldsymbol{R}}{\|} GR_r = (P_R, T_R, F_R, m_{R0}, \mu_R)$, *where* $m_0 : P \to Z^{|P|}$. *The state of* G *is called a marking and it is represented by a vector* $m \in Z^{|P|}$.

The overall model obtained by combining Figure 1a,b with Figure 2a,b is shown in Figure 3a.

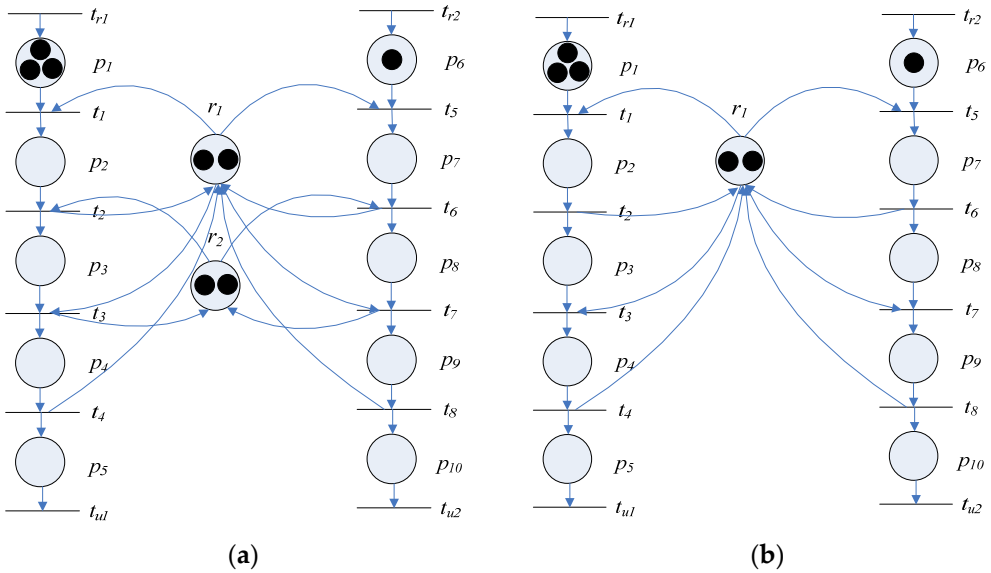


Figure 3. (**a**) Overall model obtained by merging Figure 1 with Figure 2; (**b**) A model which allows at most one resource involved for each transition.

To make it clear for readers to understand the differences between the models used in this paper and the one proposed in [13], we illustrate their differences by an example. Figure 3b is a model that is similar to the one proposed in [13]. Note that the model in Figure 3b only allows at most one resource involved for each transition. By contrast, there are two types of resources involved in transitions t_2, t_3, t_6 and t_7 in the model of Figure 3a. This is due to the synchronization of operations with the two resources in CPPS. Obviously, this means that the model proposed in this paper allows more complex interactions patterns between resources and tasks. That is, the model proposed in this paper extends the one proposed in [13]. However, this powerful modeling capability should be used in conjunction with a proper control method. Note that the overall model in Figure 3b will not evolve to any undesirable state no matter what sequence of transitions is fired. However, the overall model in Figure 3a may evolve to the circular waiting state in Figure 4. In Figure 4, all resources are held by tasks in the production processes and wait for the resources to be released. Therefore, the development of a method to supervise and control the usage of resources is an important issue.

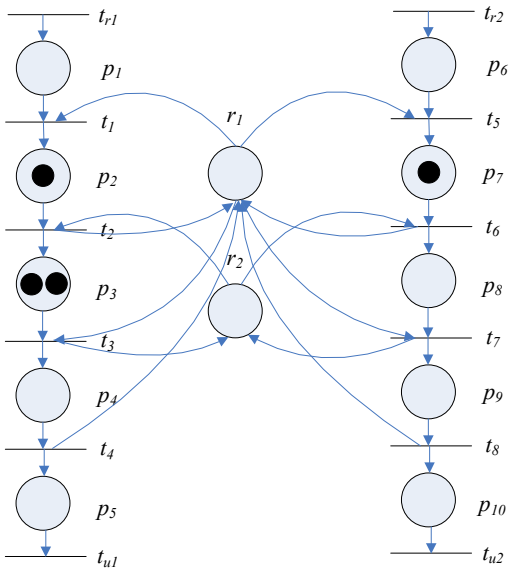


Figure 4. An undesirable state in which the two types of resources are in a circular waiting situation and a subset of transitions can no longer be fired.

Before defining the control policy, we first define the initial marking and the final marking of the CPPS model. The initial marking is set according to the order requirements. The requirements of an order can be described by product demands, D_j, where $D_j > 0 \forall j \in \boldsymbol{J}$ and deadline Φ. The initial marking (state) and final marking (state) is defined as follows.

Definition 8. *Suppose the order requirements for type-j tasks is D_j. The number of tokens in the first place p_{j0} under the initial marking m_{j0} of the type-j task subnet $GJ_j = (P_j, T_j, F_j, m_{j0}, \mu_j)$ is $m_{j0}(p_{j0}) = D_j$, where $D_j > 0$.*

The final marking of the CPPS model is defined as follows.

Definition 9. *For a task subnet $GJ_j\left(m_{jf}\right) = (P_j, T_j, F_j, m_{jf}, \mu_j)$ in the final state m_{jf} in which the order requirements for type-j tasks is fulfilled, the number of tokens in the last place p_{jN_j} of the final marking is $m_{jf}(p_{jN_j}) = D_j$.*

Definition 10. *Let r denote the idle place of type-r resource subnet GR_r. The final state of the type-r resource subnet $GR_r\left(m_{rf}\right) = (P_r, T_r, F_r, m_{rf}, \mu_r)$ satisfies $m_{rf}(p_r) = m_{r0}(p_r)$. That is, resources are in an idle state after processing the required tasks.*

Based on the initial states of tasks and resources, we define the CPPS model in the initial state as $G(m_0) = (P, T, F, m_0, \mu) = GJ \| GR$, where $GJ = \underset{j \in \boldsymbol{J}}{\|} GJ_j(m_{j0})$ and $GR = \underset{r \in \boldsymbol{R}}{\|} GR_r(m_{r0}) = (P_R, T_R, F_R, m_{R0}, \mu_R)$. Based on the final states of tasks and resources, we define the CPPS model in the final state as $G(m_f) = (P, T, F, m_f, \mu) = GJ \| GR$, where $GJ = \underset{j \in \boldsymbol{J}}{\|} GJ_j\left(m_{jf}\right)$ and $GR = \underset{r \in \boldsymbol{R}}{\|} GR_r\left(m_{rf}\right) = (P_R, T_R, F_R, m_{Rf}, \mu_R)$.

In the CPPS model $G = (P, T, F, m_0, \mu)$, a controlled transition is a transition that requires an allocation of resources to be fired. Let T_c, where $T_c \subseteq T$, be the set of controlled transitions in G. We define a control action c as a vector in $Z^{|T_c|}$ that specifies the number of times each transition in T_c to be fired concurrently. A sequence of control actions $\{c_n\}$ is called a control policy u. The behaviors of $G = (P, T, F, m_0, \mu)$ under a control policy u is denoted by $G_c = (P, T, F, m_0, \mu, u)$.

Definition 11. *A CPPS model $G = (P, T, F, m_0, \mu)$ under a control policy u is represented by a controlled DTPN $G_c = (P, T, F, m_0, \mu, u)$ and is abbreviated as $G_c(m_0, u)$, where u is a control policy that is defined by a mapping u: $R(m_0) \rightarrow Z^{|T_c|}$ that generates a sequence $\{c_n\}$ of control actions for G_c.*

A control action c is a vector in $Z^{|T_c|}$ that specifies the number of times each transition in T_c to be fired concurrently. Suppose $T_c = \{t_1, t_2, t_3, t_5, t_6, t_7\}$. For the marking in Figure 3a, the control action c = [2 0 0 0 0 0] can be applied to fire transition t_1 twice. A sequence of control actions to be executed in the CPPS model define a control policy. Not all control policies can preserve the liveness property of the system. For the example in Figure 3a, c_1 = [2 0 0 0 0 0], c_2 = [0 2 0 0 0 0] and c_3 = [1 0 0 1 0 0] are three control actions that can be applied consecutively from the marking in Figure 3a. Applying the sequence of control actions c_1, c_2 and c_3 will fire t_1 three times, fire t_2 two times and fires t_5 one time. The sequence of control actions c_1, c_2 and c_3 define a control policy. However, this control policy brings the CPPS to an undesirable circular waiting situation in Figure 4. Therefore, generation of the control actions that can maintain the liveness of the CPPS model is an important issue. The Context-Generation Algorithm for CPPS to be introduced in Section 5.3 serves to generate the sequence of control actions to be executed.

When an order is released to CPPS, processing of the order can be represented by the evolution of the states of the model G_c of CPPS under a control policy. The requirements of an order can be described by product demands, D_j, where $D_j > 0 \forall j \in \boldsymbol{J}$, and deadline Φ, where the deadline is specified by a period in the time horizon Π.

In the model of CPPS, a marking represents the state of the system. Before processing the given order, the CPPS is in its initial state with the initial marking m_0. For the given order requirements, the number of different types of products to be produced to fulfill the product demands of the order can be represented by a final marking m_f in the model of CPPS. Whether the order deadline can be met can be calculated by determining whether the marking m_f can be reached from m_0 by the deadline Φ. Throughout the evolution from m_0, the CPPS may visit undesirable states. An improper control policy may bring the system to an undesirable state under which some or all the transitions can no longer be fired. Figure 4 shows an example of an undesirable state in which transitions can no longer be fired, with the exception of t_{r1} and t_{r2}. There is a research issue and question that needs to be addressed: can the CPPS model reach the target (final) state by the deadline without visiting any undesirable state? To answer this question, we formulate the Deadline Awareness Problem and the Future States Awareness Problem.

Based on the model of CPPS, we formulate the Deadline Awareness Problem. As the Deadline Awareness Problem is to determine whether the requirements of a given order can be fulfilled by the order deadline, the Deadline Awareness Problem is to determine whether final marking m_f can be reached by the deadline Φ. Formally, this problem can be stated as:

Deadline AwarenessProblem

Given the model G of a CPPS with an initial marking m_0, determine whether G can be brought to the target marking m_f from m_0 by the deadline Φ under some control policy, where m_f and the deadline Φ are specified by the order requirements.

For the target marking m_f and the deadline Φ corresponding to the given order requirements, the Future States Awareness Problem can be formulated as follows.

Future States AwarenessProblem

Given the model G of a CPPS with an initial marking m_0, determine whether there exists a control policy to bring G to the target marking m_f without visiting any undesirable states, where the marking m_f is the marking corresponding to the given order requirements.

5. Approach to Deadline Awareness Problem and Future States Awareness Problem

To solve the Deadline Awareness Problem and Future States Awareness Problem, we must analyze the evolution of the markings of the CPPS model. We propose an optimization

approach to determine a control policy for CPPS. In this section, we first analyze the liveness condition of CPPS in Sections 5.1 and 5.2 as the liveness property to be defined in Definition 12 is not directly related to the timing aspect. Analysis of the timing or temporal properties of CPPS is presented in Section 5.3 of this paper.

5.1. Analysis of the CPPS Model

Note that the model G of a CPPS is an abstraction of the underlying production system. The minimal resource requirements must be satisfied for the CPPS to run and manufacture the products of an order. The minimal resource requirements for the model G of a CPPS can be represented by a marking m^*, where $m^*(p) = \begin{cases} \sum_{j \in \boldsymbol{J}} \sum_{t \in T} R_{jt}(r) \forall p \in P_o \\ 0 \; p \in P \backslash P_o \end{cases}$.

The Future States Awareness Problem is concerned with whether the target marking m_f can be reached without visiting any undesirable states. In Petri net theory, the concept of liveness property ensures that undesirable states will not be visited. Therefore, we solve the Future States Awareness Problem based on the concept of liveness in Petri nets. We define the liveness of $G_c(m_0, u)$ as follows.

Definition 12. *$G_c(m_0, u)$ is live if for each marking reached from m_0 under u, each transition can still be fired ultimately by progressing through some further firing sequence.*

For the existence of a control policy u under which $G_c(m_0, u)$ is live, there must be sufficient resources under the initial marking m_0. In this paper, it is assumed that the initial marking m_0 satisfies the condition: $m_0 \geq m^*$. Under this assumption, the following property states a liveness condition for the model G of a CPPS.

Property 1. *$G_c(m_0, u)$ is live under a control policyu if and only if there is a sequence of control actions that can bring each marking m reached under control policy u to m' with $m' \geq m^*$.*

Proof.

Sufficiency:

We must prove that if there is a sequence of control actions $c_1c_2c_3...c_n$ that can bring each marking m reached under control policy u to m' with $m' \geq m^*$, $G_c(m_0, u)$ is live under the control policy u. To show that $G_c(m_0, u)$ is live under the control policy u, we must show that each transition can still be fired by progressing through some further firing sequence from m'. Note that $m' \geq m^*$. The resources under m' satisfy the minimal resource requirements. Therefore, we can construct a sequence of control actions to fire transition t_{jn} in type-j task subnet by firing $t_{j0}, t_{j1}, t_{j2}, ..., t_{j(n-1)}, t_{jn}, ..., t_{jN_j}, t_{j(N_j+1)}$ sequentially. As the requirements to fire transition t in T_j is $R_{jt}(r) \leq m^*(r) \forall r \in \boldsymbol{R}$, the firing sequence $t_{j0}, t_{j1}, t_{j2}, ..., t_{j(n-1)}, t_{jn}, ..., t_{jN_j}, t_{j(N_j+1)}$ can be fired. Therefore, transition t_{jn} can be fired by progressing through the firing sequence $t_{j0}, t_{j1}, t_{j2}, ..., t_{j(n-1)}, t_{jn}$.

Let m'' be the marking reached after firing $t_{j0}, t_{j1}, t_{j2}, ..., t_{j(n-1)}, t_{jn}, ..., t_{jN_j}, t_{j(N_j+1)}$. Under m'', all the resources return to the idle state. Hence $m'' \geq m^*$ and every transition can be fired again by progressing through some firing sequence again. Based on the above reasoning, each transition can still be fired by progressing through some further firing sequence from m'.

Therefore, $G_c(m_0, u)$ is live under the control policy u.

Necessity:

Let m be a marking reached under the given control policy u. We prove by constructing a firing sequence that can bring m to m' with $m' \geq m^*$. We first construct a firing sequence that can bring m to m' under which all the resources are returned to an idle state. As $G_c(m_0, u)$ is live under a control policy u, for each marking reached from m_0 under u, each transition can still be fired by progressing through some further firing sequence. There must exist a firing sequence s from m such that the last transition $t_{j(N_j+1)}$ is fired at least

$\sum_{n=0}^{N_j} m(p_{jn})$ times for each type-j task subnet, $j \in \boldsymbol{J}$. We construct a firing sequence s' based on s to complete all the remaining tasks under m without introducing new tasks as follows.

Consider a type-j task subnet, where $j \in \boldsymbol{J}$. As transition $t_{j(N_j+1)}$ of type-j task subnet is fired at least $\sum_{n=0}^{N_j} m(p_{jn})$ times, firing sequence s from m can be represented as $s = s_1 s_2$, where transition $t_{j(N_j+1)}$ appear in s_1 exactly $\sum_{n=0}^{N_j} m(p_{jn})$ times. Note that firing sequence s_1 may contain transition t_{j0} and transition t_{j0} may appears a number of times in s_1. This means that transition t_{j0} may also be fired in s_1 a number of times. Firing transition t_{j0} will bring new task (s) into type-j task subnet and may make the transitions at the downstream fired in s_1. We construct a firing sequence s_1' based on s_1 by removing transition t_{j0} and all relevant transitions at the downstream of t_{j0} fired in s_1. Removing the transitions of new tasks from s_1 reduces the number of resources to be used for firing transitions. As s_1 can be fired from m, the resulting firing sequence s_1' can also be fired from m. Note that all the remaining operations in the existing tasks in type-j task subnet under m will be completed after firing s_1'. Therefore, all the resources required by type-j task subnet will return to the idle state. In this way, we have constructed a firing sequence s_1' that can be fired from m.

Note that the firing sequence $s' = s_1' s_2$ is still a valid firing sequence that can be fired from m and transition $t_{j'(N_{j'}+1)}$ of type-j' task subnet is fired at least $\sum_{n=0}^{N_{j'}} m(p_{j'n})$ times in $s' = s_1' s_2$ for each type-j' task subnet, where $j' \in \boldsymbol{J} \backslash \{j\}$. Therefore, we can follow a similar procedure to construct a firing sequence s'' to complete all the remaining operations in the existing tasks in each type-j' task subnet under m for each $j' \in \boldsymbol{J} \backslash \{j\}$ one by one. After firing s'' from m, a marking m' will be reached and all the resources required by each type of task subnets will return to the idle state under m'. Due to the conservation of resources, the set of resources in an idle state under m' is the same as that under m_0, therefore, $m' \geq m_0$. As $m_0 \geq m^*$, $m' \geq m^*$. Therefore, we have proved that the firing sequence s'' that can bring m to m' with $m' \geq m^*$. Let $c_1, c_2, \ldots, c_\tau$ be the sequence of control actions associated with the firing sequence s''. Hence, there is a sequence of control actions $c_1, c_2, \ldots, c_\tau$ that can bring each marking m reached under control policy u to m' with $m' \geq m^*$.

This completes the proof. □

5.2. An Optimization Approach to Determining a Control Policy for CPPS

Direct application of the existing reachability analysis method to determine the condition of Property 1 about whether there is a sequence of control actions that can bring a marking m to m' with $m' \geq m^*$ is not feasible as the state space of reachable markings of the CPPS model grows with problem size. The way we work around this problem is to construct a Joint Spatial-Temporal Network (JSTN) to capture the spatial and temporal dynamics of workflows in the networks. Timing information is taken into consideration in constructing the JSTN to capture the spatial and temporal constraints of workflows. The firing time of each transition is used in construction of a JSTN.

Based on JSTN, we will formulate a problem to determine whether there is a sequence of control actions that can bring a marking m to m' with $m' \geq m^*$. The algorithm to construct a JSTN is based on Algorithm 1 to construct the Spatial-Temporal Network (STN) for each type of task subnet. Algorithm 1 shows the Spatial–Temporal Network Construction Algorithm for a Type-j Task Subnet. The logic of Algorithm 1 to construct a STN for each type of task subnet is highly intuitive. Algorithm 1 first creates a start node s_j and an end node e_j. For a time horizon of Π periods, Algorithm 1 creates $\Pi + 1$ nodes for each $n \in \{1, 2, ..., N_j\}$, iteratively. Algorithm 1 creates an arc to connect start node s_j to the first node and then create horizontal arcs and non-horizontal arcs, iteratively. Finally, Algorithm 1 creates arcs to connect to the end node e_j. The earliness penalty function $g(\tau)$ for period $\tau \leq \Phi$ and the lateness penalty function $h(\tau)$ for $\tau > \Phi$ are used to set the weight of arcs.

Algorithm 1: Spatial–Temporal Network Construction Algorithm for Type-j Task Subnet

Input: GJ_j, Π, Φ, $g(t)$, $h(t)$
Output: $STN_j(V_j, A_j)$, where V_j is the set of nodes and A_j is the set of arcs
Step 0: Create the start node s_j
Create the end node e_j
$V_j \leftarrow \left\{s_j, e_j\right\}$
Step 1: For $n = 1$ to $N_j + 1$
For each $\tau = 1$ to $\Pi + 1$
Create a node with number $v_{(n-1)(\Pi+1)+\tau}$
$V_j = V_j \cup \left\{v_{(n-1)\Pi+\tau}\right\}$
End For
End For
Step 2: Create arc $a(s_j, 1)$ from node s_j to the node v_1
$A_j \leftarrow A_j \cup \left\{a(s_j, 1)\right\}$
Set arc cost $\lambda\left(a(s_j, 1)\right) \leftarrow 0$
Step 3: For $n = 1$ to N_j
For each $\tau = 1$ to Π
Create an arc $a(v_{(n-1)\Pi+\tau}, v_{(n-1)\Pi+\tau+1})$
$A_j \leftarrow A_j \cup \{a\}$
Set arc cost $\lambda\left(a\left(v_{(n-1)\Pi+\tau}, v_{(n-1)\Pi+\tau+1}\right)\right)$ to 0
End For
End For
Step 4: For $n = 1$ to N_j
For each $\tau = 1$ to Π
If $\tau + \mu(t_n) \leq \Pi + 1$
Create arc $a(v_{(n-1)(\Pi+1)+\tau}, v_{n(\Pi+1)+\tau+\mu(t_n)})$
$A_j \leftarrow A_j \cup \left\{a(v_{(n-1)(\Pi+1)+\tau}, v_{n(\Pi+1)+\tau+\mu(t_n)})\right\}$
Set arc cost $\lambda(a(v_{(n-1)(\Pi+1)+\tau}, v_{n(\Pi+1)+\tau+\mu(t_n)})) \leftarrow 0$
Find the resource type r for processing operation n
$A_{jrt} \leftarrow A_{jrt} \cup \left\{a(v_{(n-1)(\Pi+1)+\tau}, v_{n(\Pi+1)+\tau+\mu(t_n)})\right\}$, the set of arcs in the spatial–temporal network of type-j task involved in the use of type-r resource in period τ
End If
End For
End For
Step 5: For each $\tau = 1$ to $\Pi + 1$
Create arc $a(v_{N_j\Pi+\tau}, e_j)$ from node $v_{N_j\Pi+\tau}$ to node e_j
$A_j \leftarrow A_j \cup \left\{a(v_{N_j\Pi+\tau}, e_j)\right\}$
If $\tau > \Phi$
Set arc cost $\lambda(a(v_{N_j\Pi+\tau}, e_j)) \leftarrow h(\tau)$
Else
Set arc cost $\lambda(a(v_{N_j\Pi+\tau}, e_j)) \leftarrow g(\tau)$
End If
End For

The Joint Spatial–Temporal Network Construction Algorithm in Algorithm 2 first iteratively invokes Algorithm 1 to construct the Spatial-Temporal Network (STN) $STN_j(V_j, A_j)$ for each type of task subnet and then construct $JSTN(V, A)$ with $V = \underset{j \in \boldsymbol{J}}{\cup} V_j$ and $A = \underset{j \in \boldsymbol{J}}{\cup} A_j$. Algorithm 2 finally merges all start node s_j for all $j \in \boldsymbol{J}$ into one source node s to obtain the JSTN. Algorithm 2 in shows the Joint Spatial–Temporal Network Construction Algorithm. Note that the structure of a JSTN is different from that of a STN in [10] as JSTN represents the spatial–temporal relation of multiple types of tasks whereas a STN represents only the spatial–temporal relation of a single type of tasks.

Figure 5 shows the structure of a JSTN.

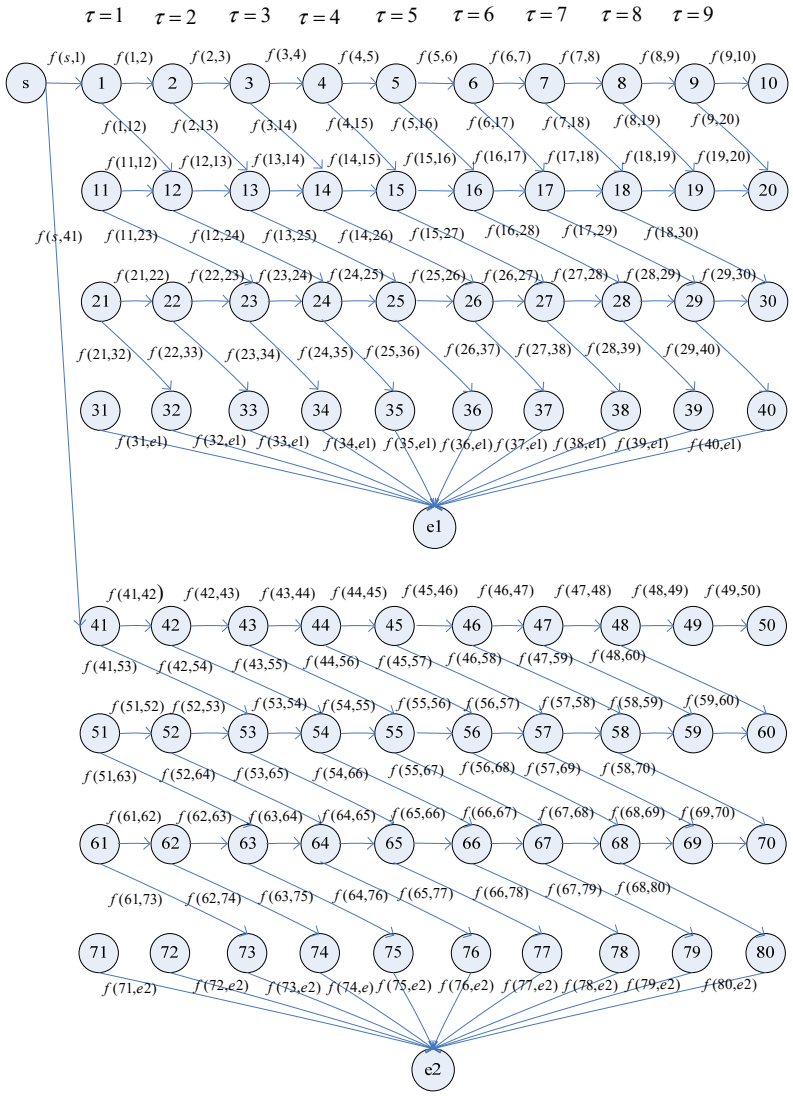


Figure 5. A Joint Spatial–Temporal Network (JSTN).

Algorithm 2: Joint Spatial–Temporal Network Construction Algorithm

Input:
Task subnets: $GJ_j,\ j \in \boldsymbol{J}$,
Time horizon: Π,
Deadline: Φ
Output: Joint Spatial–Temporal Network: $JSTN(V, A)$, where V denotes the set of nodes and A_j denotes the set of arcs
For each $j \in \boldsymbol{J}$
Construct $STN_j(V_j, A_j)$ by applying Algorithm 1, where V_j is the set of nodes and A_j is the set of arcs created.
Let $s_j, e_j \in V_j$ be the start node and end node of $STN_j(V_j, A_j)$, respectively.
Construct $JSTN(V, A)$ with
$V \leftarrow V \cup V_j$
$A \leftarrow A \cup A_j$
End For
Merge all start node $s_j,\ j \in \boldsymbol{J}$, in $JSTN(V, A)$ into one start node s.
Return $JSTN(V, A)$

The STN of type-j task subnet in G is denoted as $STN_j(V_j, A_j)$. We define the subset of arcs in A_j that require type-r resources in period t below.

Definition 13. *$A_{jr\tau}$ is the set of arcs requiring the type-r resources in period t.*

Note that the states across the time horizon of the CPPS model G can be represented in the corresponding STNs. Therefore, we propose an optimization problem formulation to determine whether there exists a control policy that can bring $G(m_0)$ to $G(m_f)$ without visiting undesirable states by the deadline.

The following Nominal Optimization Problem (NOP) is formulated based on the Joint Spatial–Temporal Network to find a solution. The NOP aims to complete the production activities for the given order by the deadline. Therefore, the objective function is the overdue cost described in (1) under the capacity constraints of different types of resources in (2), the flow balance constraints in (3), the product requirements of the order in (4), (5) and the decision variables must be nonnegative integers as described in (6).

Nominal Optimization Problem (NOP) based on the Joint Spatial–Temporal Network

$$\min_{f} \sum_{j \in \boldsymbol{J}} \sum_{r \in \boldsymbol{R}} \sum_{t \in \prod} \sum_{a(v,w) \in A_{jr\tau}} \lambda(a(v,w)) f(a(v,w)) \tag{1}$$

$$\sum_{j \in \boldsymbol{J}} \sum_{a(v,w) \in A_{jr\tau}} f(a(v,w)) \leq C_{r\tau} \forall r \in \boldsymbol{R} \, \forall \tau \in \prod = \{1, 2, ..., \Pi\} \tag{2}$$

$$\sum_{w \in V(v)} f(a(v,w)) = 0 \, \forall v \in V \backslash \{s\} \backslash \{e_j \forall j \in \boldsymbol{J}\} \tag{3}$$

$$\sum_{w \in V_j(s)} f(a(s,w)) = D_j \; \forall j \in \boldsymbol{J} \tag{4}$$

$$\sum_{w \in V_j(e_j)} f(a(w,e_j)) = D_j \; \forall j \in \boldsymbol{J} \tag{5}$$

$$f(a(v,w)) \in Z \; \forall a(v,w) \in A, \tag{6}$$

where Z is the set of nonnegative integers.

A solution of NOP can be used to define the control actions to bring G_c from m_0 to m_f without visiting undesirable states. Whether the deadline can be met depends on the solution found for NOP. Theorem 1 states a condition for ensuring the liveness of $G_c(m_0, u)$. In the proof of this theorem, we first define the control actions based on a solution of NOP. We then prove that the control actions will bring G_c from m_0 to m_f without visiting undesirable states.

Theorem 1. *There exists a solution to the Nominal Optimization Problem if and only if there exists a control policy under which $G_c(m_0, u)$ is live and can reach m_f under u.*

Proof.

Proof of "only if part":

We prove by constructing a control policy under which $G_c(m_0, u)$ is live given the precondition that there exists a solution f to NOP. To construct a control policy under which $G_c(m_0, u)$ is live, we first define the control actions for the given solution f based on the following definitions of control actions. The set of arcs in STN $STN_j(V_j, A_j)$ can be divided into three subsets, W_j, the set of arcs corresponding to waiting states, B_j, the set of arcs corresponding to busy (processing) states. There is a transition $t \in T_j$ corresponding to an arc $a \in B_j$. Suppose the time horizon is divided into Π period. Let $\{1, 2, ..., \Pi\}$ denote the set of time period. A control action for time period $\tau \in \{1, 2, ..., \Pi\}$ is denoted by c_τ and $c_\tau(t)$ denotes the control action to be applied to transition $t \in T_j$ in period $\tau \in \{1, 2, ..., \Pi\}$. For each arc $a \in B_j$, $j \in J$, starting with period $\tau \in \{1, 2, ..., \Pi\}$, let t_a be the transition

associated with arc a. We define the control action $c_\tau(t_a) = f(a)$, which will be applied to fire transition $t \in T_j$ corresponding to an arc $a \in B_j$ in period τ for $f(a)$ times. These control actions c_τ, $\tau \in \{1, 2, ..., \Pi\}$, define a control policy u associated with the solution f. Based on the above definitions, we show that the control actions c_τ, $\tau \in \{1, 2, ..., \Pi\}$ defined above can be executed. That is, for each $t \in T_j$, t can be fired $c_\tau(t_a) = f(a)$ times in period τ. We prove by showing that under the marking m_τ reached after applying the control actions c_1, c_2, c_3, ... , c_τ, there exists a control policy u' under which $G_c(m_\tau, u')$ is live.

For τ = 1, we prove that the control action applied to all relevant transitions in period 1 can be executed. Note that the number of type-r resources required to fire transitions in type-j subnet in period 1 is $\sum_{a \in A_{jr1}} c_1(t_a) = \sum_{a \in A_{jr1}} f(a)$. The total number of type-r resources required to fire transitions in type-j subnet in period 1 is $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr1}} c_1(t_a)$, which is equal to $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr1}} f(a)$. The total number of type-r resources required to fire transitions in type-j subnet in period 1 is C_{r1}. As constraint (2) holds for each $r \in \boldsymbol{R}$ and each period $\tau \in \{1, 2, ..., \Pi\}$, C_{r1} is greater than or equal to $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr1}} f(a)$ and $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr1}} f(a)$ is equal to $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr1}} c_1(t_a)$, all relevant transitions allowed to be fired under control action c_1 can be executed in period 1. Similarly, for period τ = 2, the number of type-r resources required to fire transitions in type-j subnet in period 2 is $\sum_{a \in A_{jr2}} c_2(t_a)$, which is equal to $\sum_{a \in A_{jr2}} f(a)$. The total number of type-r resources required to fire transitions in type-j subnet in period 2 is $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr2}} c_2(t_a)$, which is equal to $= \sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr2}} f(a)$. As C_{r2} is greater than or equal to $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr2}} f(a)$ and $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr2}} f(a)$ is equal to $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr2}} c_2(t_a)$, C_{r2} is greater than or equal to $\sum_{j \in \boldsymbol{J}} \sum_{a \in A_{jr2}} c_2(t_a)$. All relevant transitions under control action c_2 can be executed in period 2.

As the above reasoning holds for all $\tau \in \{1, 2, ..., \Pi\}$, the sequence of control actions c_τ, $\tau \in \{1, 2, ..., \Pi\}$, can be executed. After executing all these control actions, the final marking m_f will be reached. Under m_f, all resources are returned to an idle state. Hence, the set of resources m_f is the same as m_0. As $m_0 \geq m^*$, $m_f \geq m^*$.

Proof of "if part":

If there is a control policy under which $G_c(m_0, u)$ is live and can reach m_f under u, we need to show that there exists a solution to the Nominal Optimization Problem.

If $G_c(m_0, u)$ is live under a control policy u, the initial marking m_0 must satisfy the minimal resource requirements: $m_0 \geq m^*$ and there must be sufficient resources to fire the transitions in each type of task subnets separately. We construct a sequence of control actions to bring G_c from m_0 to m_f as follows. First, we construct a sequence of control actions to bring G_c from m_0 to m_1 under which the number of tokens in the last place $p^1_{N_1+1}$ of type-1 subnet is $m_1\left(p^1_{N_1+1}\right) = D_1$. Let c_1, c_2, ... , c_{τ_1} be the sequence of control actions that repeatedly fire transitions $t_{11}, t_{12}, ..., t_{1N_1}$ in T_1 sequentially for D_1 times from m_0. At period τ_1, the number of tokens in the last place $p^1_{N_1+1}$ of type-1 subnet is $m_1\left(p^1_{N_1+1}\right) = D_1$ and all the resource tokens are returned to an idle state. Therefore, under m_1 the set of resources n idle state is the same as the set of resources under m_0. Then, we construct a sequence of control actions c_{τ_1+1}, c_{τ_1+2}, ... , $c_{\tau_1+\tau_2}$ to bring G_c from m_1 to m_2 under which the number of tokens in the last place $p^2_{N_2+1}$ of type-2 subnet is $m_2\left(p^2_{N_2+1}\right) = D_2$ and the number of tokens in the last place $p^1_{N_1+1}$ of type-1 subnet is $m_1\left(p^1_{N_1+1}\right) = D_1$. All the resource tokens are returned to an idle state under m_2. Therefore, under m_2 the set of resources n idle state is the same as the set of resources under m_0. By following a similar procedure, we can construct a sequence of control actions c_1, c_2, ... , c_{τ_1}, c_{τ_1+1}, c_{τ_1+2}, ... , $c_{\tau_1+\tau_2}$, ... , $c_{\tau_{J-1}+1}$, $c_{\tau_{J-1}+2}$, ... , $c_{\tau_1+\tau_2+...+\tau_{J-1}+\tau_J}$ to bring G_c from m_0 to m_J under which

the number of tokens in the last place $p^j_{N_j+1}$ of type-j subnet is $m_J\left(p^j_{N_j+1}\right) = D_j \; \forall j \in \boldsymbol{J}$ and all the resources are returned to an idle state.

As all resources are returned to an idle state under, m_f, the sequence of control actions $c_1, c_2, \ldots, c_{\tau_1}, c_{\tau_1+1}, c_{\tau_1+2}, \ldots, c_{\tau_1+\tau_2}, \ldots, c_{\tau_{J-1}+1}, c_{\tau_{J-1}+2}, \ldots, c_{\tau_1+\tau_2+\ldots+\tau_{J-1}+\tau_J}$ brings G_c from m_0 to m_f. As f is a solution of the Nominal Optimization Problem, conservation of flows due to constraints (3), (4) and (5) must be satisfied. Therefore, $m_J = m_f$. So m_f can be reached after executing the sequence of control actions $c_1, c_2, \ldots, c_{\tau_1}, c_{\tau_1+1}, c_{\tau_1+2}, \ldots, c_{\tau_1+\tau_2}, \ldots, c_{\tau_{J-1}+1}, c_{\tau_{J-1}+2}, \ldots, c_{\tau_1+\tau_2+\ldots+\tau_{J-1}+\tau_J}$. The sequence of control actions $c_1, c_2, \ldots, c_{\tau_1}, c_{\tau_1+1}, c_{\tau_1+2}, \ldots, c_{\tau_1+\tau_2}, \ldots, c_{\tau_{J-1}+1}, c_{\tau_{J-1}+2}, \ldots, c_{\tau_1+\tau_2+\ldots+\tau_{J-1}+\tau_J}$ is a control policy to reach m_f.

This completes the proof. □

5.3. Situation-Aware Context Generation Algorithm

In this subsection, we first present Theorem 2 for deadline awareness and future states awareness and then a context generation algorithm for CPPS. Based on Theorem 1, we have the following result.

Theorem 2. *Assume that $g(\tau) = 0$ for $\tau \leq \Phi$ and $h(\tau) > 0$ for $\tau > \Phi$. If there exists a solution to the Nominal Optimization Problem, there exists a control policy u under which $G_c(m_0, u)$ can reach m_f without visiting any undesirable states. If the objective function value of the solution to the Nominal Optimization Problem is zero, $G_c(m_0, u)$ can reach m_f by the deadline Φ under control policy u.*

Proof.

Let f be a solution to the NOP. It follows from Theorem 1 that there exists a control policy under which $G_c(m_0, u)$ is live and can reach m_f under u. Hence, undesirable states will not be visited under u. If the objective function value of the solution to the Nominal Optimization Problem is zero, $\sum_{j\in\boldsymbol{J}}\sum_{r\in\boldsymbol{R}}\sum_{t\in\prod}\sum_{a(v,w)\in A_{jrt}} \lambda(a(v,w))f(a(v,w))$ must be 0.

Let $A_{je} = \left\{a(v_{N_j\Pi+t}, e_j), t \in \{1, 2, ..., \Pi\}\right\}$ denote the set of arcs in the type-j task subnet directly connecting to the end node e_j. Note that each term $\sum_{a(v,w)\in A_{jrt}} \lambda(a(v,w))f(a(v,w))$ in $\sum_{j\in\boldsymbol{J}}\sum_{r\in\boldsymbol{R}}\sum_{t\in\prod}\sum_{a(v,w)\in A_{jrt}} \lambda(a(v,w))f(a(v,w))$ can be broken down into two terms, $\sum_{a(v,w)\in A_{jrt}\setminus A_{je}} \lambda(a(v,w))f(a(v,w)$ and $\sum_{a(v,w)\in A_{je}} \lambda(a(v,w))f(a(v,w))$.

As $\lambda(a(v,w))$ is equal to zero for all arcs with the exception of the arcs that connect to the end nodes, the following holds:

$\lambda(a(v,w)) = 0$ for each $a(v,w) \in A_{jrt}\setminus A_{je}$.

Therefore, the term $\sum_{a(v,w))\in A_{jrt}} \lambda(a(v,w))f(a(v,w))$ is reduced to $\sum_{a(v,w)\in A_{je}} \lambda(a(v,w))f(a(v,w))$, which is equal to $\sum_{\tau\in\{1,2,\ldots,\Pi\}} \lambda(a(v_{N_j\Pi+\tau}, e_j))f(a(v_{N_j\Pi+\tau}, e_j))$.

As $g(\tau) = 0$ for $\tau \leq \Phi$, $\lambda(a(v_{N_j\Pi+\tau}, e_j))$ is equal to zero for each $\tau \in \{1, 2, ..., \Phi\}$ and $\sum_{\tau\in\{1,2,\ldots,\Pi\}} \lambda(a(v_{N_j\Pi+\tau}, e_j))f(a(v_{N_j\Pi+\tau}, e_j))$ is reduced to $\sum_{\tau\in\{\Phi+1,\Phi+2,\ldots,\Pi\}} \lambda(a(v_{N_j\Pi+\tau}, e_j))f(a(v_{N_j\Pi+\tau}, e_j))$.

Hence $\sum_{\tau\in\{\Phi+1,\Phi+2,\ldots,\Pi\}} \lambda(a(v_{N_j\Pi+\tau}, e_j))f(a(v_{N_j\Pi+\tau}, e_j))$ is equal to 0.

As $h(\tau) > 0$ for, $\lambda(a(v_{N_j\Pi+t}, e_j)) > 0$ for each $\tau \in \{\Phi + 1, \Phi + 2, ..., \Pi\}$.

As $f(a(v_{N_j\Pi+t}, e_j))$ is nonnegative, $f(a(v_{N_j\Pi+\tau}, e_j))$ must be zero for each $\tau \in \{\Phi + 1, \Phi + 2, ..., \Pi\}$.

Therefore, the deadline Φ can be met. □

Based on the theorem presented above, we propose Algorithm 3 to generate the contextual information (control actions) for Future States-Aware systems. The following algorithm first constructs $JSTN(V, A)$ for the model G of a CPPS with an initial marking m_0 and formulates NOP next. Finally, the NOP is solved to find a solution. If a solution can be found by solving the NOP, the proposed algorithm will generate the Future States-Aware Context. Otherwise, it will exit.

Algorithm 3: Context Generation Algorithm for CPPS

Step 1: Construct $JSTN(V, A)$ for the model G of a CPPS with an initial marking m_0
Step 2: Formulate the nominal optimization problem
  Formulate NOP according to $JSTN(V, A)$
Step 3: Solve the nominal optimization problem
  Find the solution of NOP
  If there is a solution f
    if $\sum_{j\in \boldsymbol{J}} \sum_{r\in \boldsymbol{R}} \sum_{t\in \prod} \sum_{a(v,w)\in A_{jrt}} \lambda(a(v,w)) f(a(v,w)) = 0$
      The deadline Φ can be met.
    Else
      The deadline Φ cannot be met.
    End If
    Go to Step 4
  Else
    Exit
  End If
Step 4: Construct the control actions based on the solution found
  For $j \in \boldsymbol{J}$
    For each $t \in T_j$
      For each $\tau \in \{1, 2, ..., \Pi\}$
      Find the arc $a(v, w)$ associated with transition t and period τ
      $c_\tau(t_a) = f(a(v, w))$
      End For
    End For
  End For

In Property 2, we analyze the complexity of constructing the Joint Spatial-Temporal Network.

Property 2. *The complexity of constructing the Joint Spatial-Temporal Network $JSTN(V, A)$ by applying the Joint Spatial–Temporal Network Construction Algorithm is $O(K\Pi|\boldsymbol{J}|)$.*

Proof.

As a CPPS consists of a number of task subnets, the Joint Spatial–Temporal Network Construction Algorithm constructs $JSTN(V, A)$ by applying Algorithm 1 to construct $STN_j(V_j, A_j)$. As the number of transitions in different task subnets may not be the same, to analyze the complexity of constructing $JSTN(V, A)$, let $K = \max_{j\in \boldsymbol{J}} |T_j|$ denote the number of transitions in the task subnet with the maximum number of transitions. The number operations to create nodes in $STN_j(V_j, A_j)$ is less than or equal to $K\Pi$ for each $j \in \boldsymbol{J}$. The number operations to create arcs in $STN_j(V_j, A_j)$ is less than or equal to $2K\Pi$ for each $j \in \boldsymbol{J}$. Hence, the total number of operations to create $STN_j(V_j, A_j)$ is less than or equal to $3K\Pi$. As there are $|\boldsymbol{J}|$ types of task subnets, the total number of operations to create $STN_j(V_j, A_j)$ for each $j \in \boldsymbol{J}$ is less than or equal to $3K\Pi|\boldsymbol{J}|$. The number of operations to merge all start nodes is $|\boldsymbol{J}|$ and the number of operations to merge all start nodes is $|\boldsymbol{J}|$. Therefore, the total number of operations to construct $JSTN(V, A)$ is less than or equal to $3K\Pi|\boldsymbol{J}| + 2|\boldsymbol{J}|$. Therefore, the complexity of constructing $JSTN(V, A)$ is $O(K\Pi|\boldsymbol{J}|)$. □

We establish a lower bound on the complexity of solving NOP in Property 3.

Property 3. *A lower bound on the complexity of solving NOP is* $O(K\Pi|\boldsymbol{J}|)$.

Proof.

As the Nominal Optimization Problem is formulated based on the Joint Spatial–Temporal Network, it is similar to the classical minimum cost flow problem in the literature as the flow balance constraints must be satisfied. However, the Nominal Optimization Problem is different from the classical minimum cost flow problem in that the additional capacity constraints must be satisfied. Therefore, the complexity of solving the Nominal Optimization Problem is expected to be different from that of the classical minimum cost flow problem. Despite this difference, the complexity of the classical minimum cost flow problem provides a lower bound on the complexity of the Nominal Optimization Problem. As the number of nodes in the Joint Spatial–Temporal Network is proportional to $K\Pi|\boldsymbol{J}|$, $O(K^2\Pi^2|\boldsymbol{J}|^2\sum_{j\in J} D_j)$. □

6. Results

The theory and algorithm presented in the previous sections are illustrated by a small example. In addition, we conducted a series of experiments to assess the computational efficiency of the proposed methods and to illustrate the potential to solve real problems. The test data for the small example is described in details in the text. The test data for large examples can be downloaded from the following link:

https://drive.google.com/drive/folders/1SCcf-VWkrSy_iwUoUw0Ky-xxZCrdKtBo?usp=sharing (accessed on 27 February 2021).

All the experiments were conducted on a personal computer with Intel CoreTM i7-4720 HQ CPU, 2.6 G Hz, and 16 GB of onboard memory.

6.1. An Example

In this subsection, we illustrate the proposed method with an example.

Example 1. *Consider a CPPS with two production processes to produce two types of products. The two production processes are modeled by two task subnets. There are two types of resources in the CPPS. The two task subnets share the two types of resources. The model of the CPPS is depicted in Figure 3a. The firing time of each transition is listed in Table 2. Suppose an order is received and is to be released to the CPPS for processing the products. The requirements of the order are shown in Table 3. The product demands of the order include three type-1 products and one type-2 product. The deadline of the order is the eighth period. For this example,* $J = 2$, $D_1 = 3$, $D_2 = 1$, $\Phi = 8$, $R = 2$. *We set the time horizon* $\Pi = 9$. *Therefore,* $\boldsymbol{J} = \{1, 2\}$ *and* $\boldsymbol{R} = \{1, 2\}$. *The nominal capacity of type-r resources in period* τ *is* $C_{r\tau} = 2$ *for each* $r \in \boldsymbol{R}$ *and* $\tau \in \{1, 2, ..., \Pi\}$. *The total number of type-r resources in each period* τ *is listed in Table 4. The initial marking* $m_{j0}(p_{j0}) = D_j$ *for* $j \in \{1, 2\}$. *In this example, we use set the function* $g(\tau) = 0$ *for* $\tau \leq \Phi$ *and* $h(\tau) > 0$ *for* $\tau > \Phi$ *to penalize overdue. As* $\Phi = 8$, $\lambda(40, e1) = 1$, $\lambda(80, e1) = 1$ *and* $\lambda(a(v, w)) = 0$ *for all the other arc* $a(v, w)$. *For this example,* $A_{jr\tau}$ *is listed in Table 5.*

The NOP for this example is as follows, where (7) is the objective function, (8) is the capacity constraints, (9) is the flow balance constraints, (10), (11) are the demand constraints, (12) and (13) are the final product constraints.

$$\min_{f} \sum_{j\in\boldsymbol{J}} \sum_{r\in R} \sum_{t\in\prod} \sum_{a(v,w)\in A_{jr\tau}} \lambda(a(v,w)) f(a(v,w)) \tag{7}$$

$$\sum_{j\in\boldsymbol{J}=\{1,2\}} \sum_{a(v,w)\in A_{jr\tau}} f(a(v,w)) \leq C_{r\tau} \forall r \in \boldsymbol{R} \forall \tau \in \prod = \{1, 2, ..., \Pi\} \tag{8}$$

$$\sum_{w\in V(v)} f(a(v,w)) = 0 \,\forall v \in V \backslash \{s, e_1, e_2\} \tag{9}$$

$$f(a(s, 1)) = D_1 \tag{10}$$

$$f(a(s,41)) = D_2 \tag{11}$$

$$\sum_{v=31}^{40} f(a(v+11,e1)) = D_1 \tag{12}$$

$$\sum_{v=71}^{80} f(a(v,e1)) = D_2 \tag{13}$$

$$f(a(v,w)) \in Z \forall (a(v,w)) \in A, \tag{14}$$

where Z is the set of nonnegative integers.

Table 2. Firing time of each transition in Figure 3a.

Task Type	Transition	Firing time
1	t_1	$\mu(t_1) = 1$
1	t_2	$\mu(t_2) = 2$
1	t_3	$\mu(t_3) = 1$
1	t_4	$\mu(t_4) = 0$
2	t_5	$\mu(t_5) = 2$
2	t_6	$\mu(t_6) = 2$
2	t_7	$\mu(t_7) = 2$
2	t_8	$\mu(t_8) = 0$

Table 3. Requirements of an order.

Product (Task) Type	Quantity	Deadline
1	$D_1 = 3$	8
2	$D_2 = 1$	8

Table 4. The nominal capacity of each type of resources in each period.

Resource Type (r)	Period (t)	C_{rt}
1	1	2
1	2	2
1	3	2
1	4	2
1	5	2
1	6	2
1	7	2
1	8	2
1	9	2
2	1	2
2	2	2
2	3	2
2	4	2
2	5	2
2	6	2
2	7	2
2	8	2
2	9	2

We apply the CPLEX problem solver [56] to solve the NOP in the Context-Generation Algorithm. By solving the NOP, we find the solution in Table 6, which can be represented in the Joint Spatial-Temporal Network in Figure 6. The objective function value is 0. Therefore, the deadline can be met.

Table 5. The data of $A_{jr\tau}$.

Task Type (j)	**Resource Type** (r)	**Period** (τ)	$A_{jr\tau}$
1	1	1	{(1,12),(21,32)}
1	1	2	{(2,13),(22,33)}
1	1	3	{(3,14),(23,34)}
1	1	4	{(4,15),(24,35)}
1	1	5	{(5,16),(25,36)}
1	1	6	{(6,17),(26,37)}
1	1	7	{(7,18),(27,38)}
1	1	8	{(8,19)(28,39)}
1	1	9	{(9,20),(29,40)}
1	2	1	{(11,23)}
1	2	2	{(11,23),(12,24)}
1	2	3	{(12,24),(13,25)}
1	2	4	{(13,25),(14,26)}
1	2	5	{(14,26),(15,27)}
1	2	6	{(15,27),(16,28)}
1	2	7	{(16,28),(17,29)}
1	2	8	{(17,29)}
2	1	1	{(41,53),(61,73)}
2	1	2	{(41,53),(61,73),(42,54),(62,74)}
2	1	3	{(42,54),(62,74),(43,55),(63,75) }
2	1	4	{(43,55),(63,75),(44,56),(64,76)}
2	1	5	{(44,56),(64,76),(45,57),(65,77)}
2	1	6	{(45,57),(65,77),(46,58),(66,78)}
2	1	7	{(46,58),(66,78),(47,59),(67,79)}
2	1	8	{(47,59),(67,79),(48,60),(68,80)}
2	2	1	{(51,63)}
2	2	2	{(51,63), (52,64)}
2	2	3	{(52,64),(53,65)}
2	2	4	{(53,65),(54,66)}
2	2	5	{(54,66),(55,67)}
2	2	6	{(55,67),(56,68)}
2	1	7	{(56,68),(57,69))}
2	1	8	{(57,69), (58,70)}

Table 6. The solution obtained by solving NOP.

Decision Variable	**Value**	**Decision Variable**	**Value**
$f(s,1)$	3	$f(24,35)$	1
$f(1,2)$	2	$f(28,39)$	2
$f(2,3)$	2	$f(35,e1)$	1
$f(3,4)$	1	$f(39,e1)$	2
$f(4,5)$	1	$f(41,53)$	1
$f(1,12)$	1	$f(53,65)$	1
$f(4,15)$	1	$f(65,66)$	1
$f(5,16)$	1	$f(66,78)$	1
$f(12,24)$	1	$f(78,e2)$	1
$f(16,28)$	2		

The sequence of control actions constructed based on the solution in Figure 6 is listed in Table 7.

Table 7. The sequence of control actions.

Period (τ)	**Control Action** c_τ
1	1 0 0 1 0 0
2	0 1 0 0 0 0
3	0 0 0 0 1 0
4	1 0 1 0 0 0
5	1 0 0 0 0 0
6	0 2 0 0 0 1
7	0 0 0 0 0 0
8	0 0 2 0 0 0
9	0 0 0 0 0 0

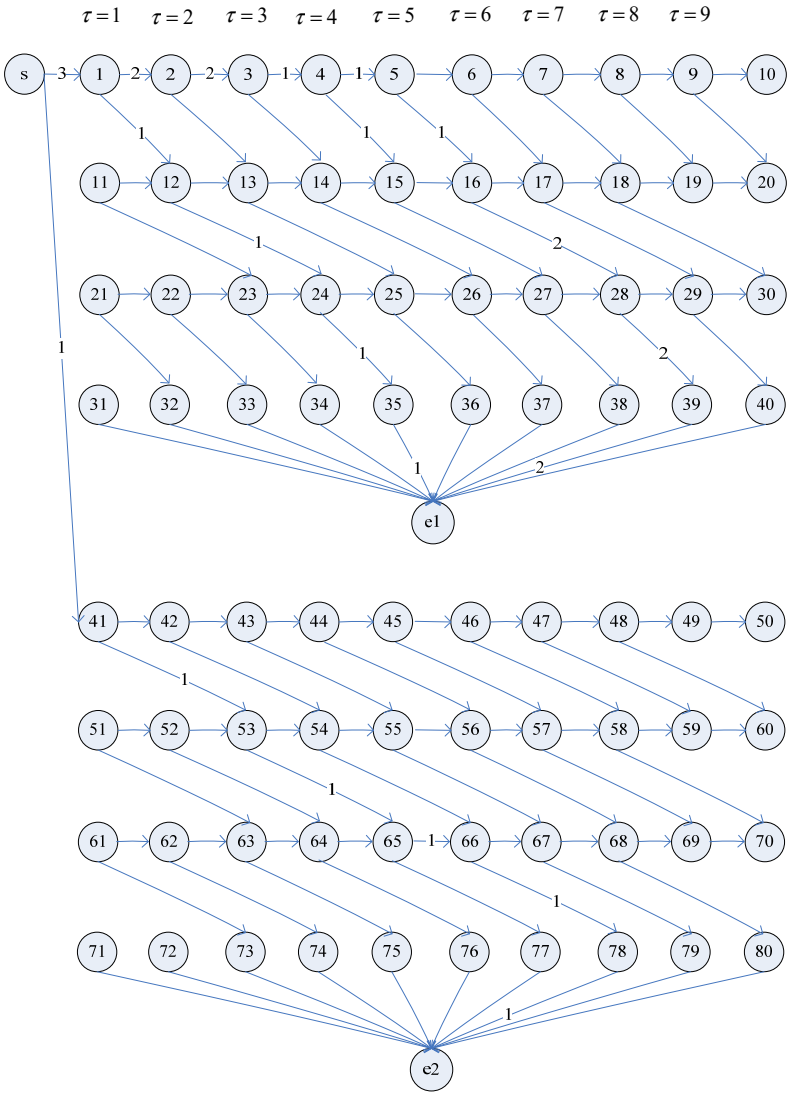


Figure 6. A solution of NOP represented in JSTN.

Following is an explanation of the control actions generated for the first period and the second period. For the first period (τ = 1), the control action c_1 = [1 0 0 1 0 0] represents that transition t_1 and transition t_5 will be fired. This means that the two transitions will be fired concurrently. For the second period (τ = 2), the control action c_2 = [0 1 0 0 0 0] represents that transition t_2 will be fired. Note that all the constraints are satisfied for the generated control actions.

6.2. Computational Experience

In this section, we study how the CPU time grows with the scale of the problems. The data of test cases (Problem 1 through Problem 15) are available for downloading from the following link:

https://drive.google.com/drive/folders/1SCcf-VWkrSy_iwUoUw0Ky-xxZCrdKtBo?usp=sharing (accessed on 27 February 2021).

In CPPS, the problem size parameters include the maximal number of operations in the task subnets, the time horizon and the number of task types. These problem size parameters correspond to the parameters $K = \max_{j \in \boldsymbol{J}} |T_j|$, Π and $|\boldsymbol{J}|$ in this paper. To study the influence of each parameter on the CPU time, we perform many experiments by changing each parameter and recording the results. We illustrate the consistency between the results and our expectations.

First, we perform several experiments by changing the parameter $K = \max_{j \in \boldsymbol{J}} |T_j|$ from 10 to 50 while keeping other parameters unchanged ($|\boldsymbol{J}| = 4$ and $\Pi = 200$). To be specific, we set K to be 10, 20, 30, 40 and 50, respectively. The results are shown in Figure 7. The results indicate that the CPU time grows approximately linearly with K. This result is better than the projection of the polynomial lower bound $O(K^2\Pi^2|\boldsymbol{J}|^2 \sum_{j \in \boldsymbol{J}} D_j)$ on the complexity of solving NOP.

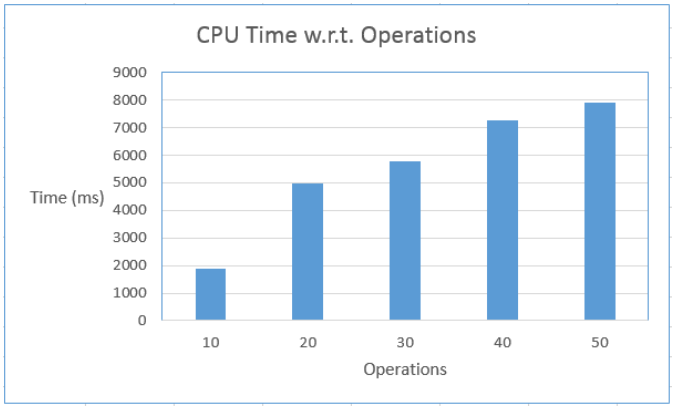


Figure 7. CPU time with respect to the number of operations (Problems 1~5).

To study the growth of CPU time with respect to the time horizon, we perform several experiments by changing the parameter Π from 100 to 500 while keeping other parameters unchanged ($|\boldsymbol{J}| = 4$ and $K = 10$). To be specific, we set Π to be 100, 200, 300, 400 and 500, respectively. The results are shown in Figure 8. The results indicate that the CPU time grows polynominally with Π. This result is consistent with the projection of polynomial lower bound $O(K^2\Pi^2|\boldsymbol{J}|^2 \sum_{j \in \boldsymbol{J}} D_j)$ on the complexity of solving NOP.

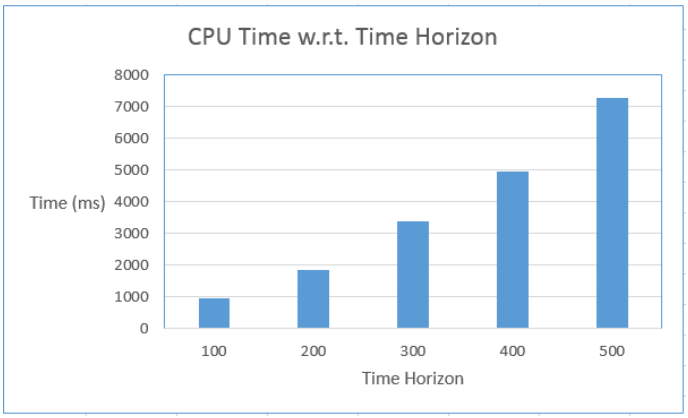


Figure 8. CPU time with respect to time horizon (Problems 11~15).

To study the growth of CPU time with regards to the number of task types, we perform several experiments by changing the parameter $|\boldsymbol{J}|$ from 4 to 20 while keeping other parameters unchanged ($\Pi = 100$ and $K = 10$). To be specific, we set $|\boldsymbol{J}|$ to be 4, 8, 12, 16 and 20, respectively. The results are shown in Figure 9. The results indicate that the CPU time grows polynominally with $|\boldsymbol{J}|$. This result is consistent with the projection of polynomial lower bound $O(K^2\Pi^2|\boldsymbol{J}|^2 \sum_{j \in \boldsymbol{J}} D_j)$ on the complexity of solving NOP.

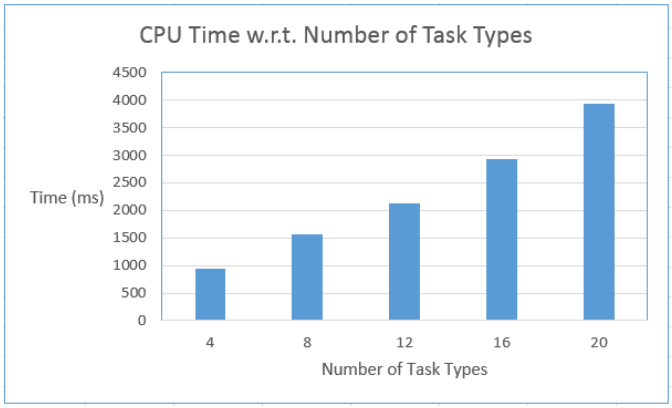


Figure 9. CPU time with respect to number of task types (Problems 21~25).

7. Discussion

In this paper, we propose the theory to lay a foundation for developing context-aware applications in CPPS. Two context awareness properties are crucial for the development of context-aware applications in CPPS. One is deadline awareness and the other is future states awareness. We develop a theory to pave the way for deadline-aware and future states-aware applications in CPPS based on Discrete Timed Petri Nets. The theory and algorithms developed in this paper provide an approach to determining whether the deadline of an order can be met without visiting any undesirable states. Accompanying the theory presented in this paper is an example to illustrate the application of the theory.

As computational feasibility is an important factor in the development of context-aware applications for CPPS, analysis of computational complexity and verification by experiments are important steps to illustrate the practicality of the theory for solving real problems. Therefore, we conduct an analysis of the decision problem and conduct experiments to study the computational feasibility of the proposed method. We study how the CPU time grows with the scale of the problem. In CPPS, the problem size parameters include the maximal number of operations in the task subnets, the time horizon and the number of task types. These problem size parameters correspond to the parameters $K = \max_{j \in J} |T_j|$, Π and $|\boldsymbol{J}|$ in this paper.

The algorithm to construct JSTN is of polynomial complexity (according to Property 2: The complexity to construct the Joint Spatial–Temporal Network $JSTN(V, A)$ by applying the Joint Spatial–Temporal Network Construction Algorithm is $O(K\Pi|\boldsymbol{J}|)$. A lower bound on the complexity of solving NOP, which is defined on JSTN, is $O(K\Pi|\boldsymbol{J}|)$ according to Property 3. Overall, a lower bound on the complexity of constructing JSTN and solving NOP is of polynomial complexity.

To study the influence of each parameter on the CPU time, we perform many experiments by changing each parameter and recording the results. Our experimental results show that the CPU time grows approximately linearly with K and grows polynominally with Π and $|\boldsymbol{J}|$. These results are consistent with the projection of the polynomial lower bound $O(K^2\Pi^2|\boldsymbol{J}|^2 \sum_{j \in \boldsymbol{J}} D_j)$ on the complexity of solving NOP. Based on our experimental results, we illustrate the consistency between the results and our expectations. This also indicates the practicality of the proposed theory in solving real problems in CPPS. Figure 7 through Figure 9 includes the construction of JSTN and solving of NOP. Note that markings of CPPS are neither embedded nor included in the construction process of JSTN. This avoids the state explosion problem.

In the literature, different approaches were proposed to tackle the complexity issue of time Petri nets and timed Petri nets. Berthomieu et al. [46–48] and Klai et al. [49] proposed different methods to reduce complexity in the analysis of time Petri nets. Lefebvre et al. proposed different methods to reduce complexity in the analysis of time Petri nets [50,51].

In terms of the models used, the nets considered by all the works mentioned above were general, whereas a subclass of nets is considered in our paper. Even though only a subclass of nets is considered, the subclass of nets considered in this paper can be used to model the production processes commonly used in the manufacturing sector. As the net structure considered in the works mentioned above is general, a variety of approaches such as the construction of a State Class Graph, Timed Aggregate Graph, Timed Extended Reachability Graph and Approximated Timed Reachability Graph were proposed to reduce complexity. However, all these methods suffer from state explosion problems as the complexity of constructing and exploring variants of reachability graphs grows exponentially with the problem size. As the models used in this paper belong to a subclass of nets, we developed a more efficient algorithm based on the proposed JSTN based method by exploiting the net structure. The proposed JSTN based method does not rely on the explicit construction of reachability graphs, State Class Graphs or Timed Extended Reachability Graphs or their variants. As a result, we can identify a lower bound on the complexity of constructing JSTN and solving NOP, which is of polynomial complexity.

The problems addressed in this study are different from the ones in the literature. Although there are many approaches to context-aware workflow systems in [57], there are only a few studies on the development of context-aware applications and services based on different variants of Petri nets, e.g., [58–60]. In particular, existing studies on modeling, analyzing and generating time-relevant contextual information for context-aware applications and systems are limited, e.g., [9,13,15,61]. Issues regarding the achievement of situation awareness, including deadline awareness and future states awareness in Cyber-Physical Production Systems is less explored. This paper attempts to bridge this gap by expanding the preliminary results reported in [62] through developing relevant theory and algorithms to address situation awareness problems of CPPS, and conducting experiments to study the feasibility of the proposed approach. The results of this study confirm the feasibility of the proposed method.

8. Conclusions

The complex workflows and heterogeneous entities in CPPS pose several challenges in the development of context-aware applications for CPPS. Due to contention and sharing of resources among different workflows and production processes in CPPS, a state that is seemingly not undesirable at one point in time may evolve to another state that is undesirable in the future. Therefore, development of context-aware applications for CPPS should consider not only the current state but also future states of the system. In other words, situation awareness is an important factor that must be considered in context-aware applications for CPPS. However, relevant studies are still lacking on situation awareness for CPPS. This situation awareness requirement raises an important research issue in CPPS. For this reason, we focus on the development of a theory for the analysis of situation awareness property for CPPS to pave the way for the development of context-aware applications for CPPS. In this paper, we study problems relevant to identifying situations of CPPS. These include the Deadline Awareness Problem and Future States Awareness Problem. The Deadline Awareness Problem aims to determine whether the deadline can be met in the future, whereas the Future States Awareness Problem aims to determine whether the undesirable states can be avoided in the process to evolve to the target state from the current state in CPPS. We develop the theory and relevant algorithms to address the above problems and support analysis of the situation of CPPS. We illustrate the application of the theory to an application scenario through the use of an example. To assess the computational efficiency of the proposed algorithms, we analyze the complexity and conduct experiments to verify the CPU time for applying the proposed algorithm to many test cases. The results show that the CPU time grows polynominally with the scale of the problem and are consistent with the expected lower bound of computational complexity. This study provides an alternative approach to the analysis of a class of timed Petri nets without relying on the exploration of graphs that are defined in the literature based on

the State Class of general time Petri nets, such as the State Class Graph, Timed Aggregate Graph, Timed Extended Reachability Graph and Approximated Timed Reachability Graph. This study sheds light on a promising approach to the development of efficient algorithms for analyzing subclasses of Petri nets by exploiting their structures. The development of relevant theory to analyze situation awareness properties in other problem domains is one interesting direction of future research.

Funding: This research was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 110-2410-H-324-001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository described in the article.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Lee, E.A. Cyber-physical systems-are computing foundations adequate? In Proceedings of the Position Paper for NSF Workshop Cyber-Physical Systems: Research Motivation, Techniques and Road Map, Austin, TX, USA, 16–17 October 2006.
2. Ivanov, R.; Weimer, J.; Lee, I. Towards Context-Aware Cyber-Physical Systems. In Proceedings of the 2018 IEEE Workshop on Monitoring and Testing of Cyber-Physical Systems (MT-CPS), Porto, Portugal, 10–13 April 2018; pp. 10–11.
3. Sahlab, N.; Jazdi, N.; Weyrich, M. An Approach for Context-Aware Cyber-Physical Automation Systems. *IFAC-PapersOnLine* **2021**, *54*, 171–176. [CrossRef]
4. Monostori, L. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* **2014**, *17*, 9–13.
5. Lee, J.; Bagheri, B.; Kao, H.-A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manuf. Lett.* **2015**, *3*, 18–23. [CrossRef]
6. Cardin, O. Classification of cyber-physical production systems applications: Proposition of an analysis framework. *Comput. Ind.* **2019**, *104*, 11–21. [CrossRef]
7. Hong, J.; Suh, E.; Kim, S.J. Context-aware systems: A literature review and classification. *Expert Syst. Appl.* **2009**, *36*, 8509–8522. [CrossRef]
8. Situation Awareness. Available online: https://en.wikipedia.org/wiki/Situation_awareness (accessed on 28 February 2022).
9. Hsieh, F.-S. A Dynamic Context-Aware Workflow Management Scheme for Cyber-Physical Systems Based on Multi-Agent System Architecture. *Appl. Sci.* **2021**, *11*, 2030. [CrossRef]
10. Petri, C.A. Kommunikation mit Automaten. Ph.D. Thesis, Institut für Instrumentelle Mathematik, Bonn, Germany, 1962.
11. Ramchandani, C. Analysis of Asvnchronous Concurrent Systems by Timed Petri Nets. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.
12. Merlin, P.M. A Study of the Recoverability of Computing Systems. Ph.D. Thesis, Department of Information and Computer Science, University of California, Irvine, CA, USA, 1974.
13. Hsieh, F.-S. Temporal Analysis of Influence of Resource Failures on Cyber-Physical Systems Based on Discrete Timed Petri Nets. *Appl. Sci.* **2021**, *11*, 6469. [CrossRef]
14. Lefebvre, D.; Daoui, C. Control Design for Bounded Partially Controlled TPNs Using Timed Extended Reachability Graphs and MDP. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 2273–2283. [CrossRef]
15. Hsieh, F.-S.; Lin, J.B. Development of context-aware workflow systems based on Petri Net Markup Language. *Comput. Stand. Interfaces* **2014**, *36*, 672–685. [CrossRef]
16. Hsieh, F.-S.; Lin, J.-B. Context-aware workflow management for virtual enterprises based on coordination of agents. *J. Intell. Manuf.* **2012**, *25*, 393–412. [CrossRef]
17. Jyotish, N.K.; Singh, L.K.; Kumar, C. A state-of-the-art review on performance measurement petri net models for safety critical systems of NPP. *Ann. Nucl. Energy* **2022**, *165*, 108635. [CrossRef]
18. Riahi, I.; Moussa, F. A formal approach for modeling context-aware Human-Computer System. *Comput. Electr. Eng.* **2015**, *44*, 241–261. [CrossRef]
19. Jyotish, N.K.; Singh, L.K.; Kumar, C.; Ahmed, F.D.; Majid, M.A. Towards agent-based petri net decision making modelling for cloud service composition: A literature survey. *J. Netw. Comput. Appl.* **2019**, *130*, 14–38.
20. Schilit, B.; Adams, N.; Want, R. Context-Aware Computing Applications. In Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, 8–9 December 1994; pp. 85–90.
21. Alegre, U.; Augusto, J.C.; Clark, T. Engineering context-aware systems and applications: A survey. *J. Syst. Softw.* **2016**, *117*, 55–83. [CrossRef]
22. Perera, C.; Zaslavsky, A.B.; Christen, P.; Georgakopoulos, D. Context Aware Computing for The Internet of Things: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 414–454. [CrossRef]

23. Dey, A.K. Understanding and Using Context. *Pers. Ubiquitous Comput.* **2015**, *1*, 4–7. [CrossRef]
24. Subbu, K.P.; Vasilakos, A.V. Big Data for Context Aware Computing–Perspectives and Challenges. *Big Data Res.* **2017**, *10*, 33–43. [CrossRef]
25. Capurso, N.; Mei, B.; Song, T.; Cheng, X.; Yu, J. A survey on key fields of context awareness for mobile devices. *J. Netw. Comput. Appl.* **2018**, *118*, 44–60. [CrossRef]
26. Kurniawan, I.F.; Asyhari, A.T.; He, F.; Liu, Y. Mobile computing and communications-driven fog-assisted disaster evacuation techniques for context-aware guidance support: A survey. *Comput. Commun.* **2021**, *179*, 195–216. [CrossRef]
27. Kulkarni, S.; Rodd, S.F. Context Aware Recommendation Systems: A review of the state of the art techniques. *Comput. Sci. Rev.* **2020**, *37*, 100255. [CrossRef]
28. Islam, M.S.U.; Kumar, A.; Hu, Y.-C. Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions. *J. Netw. Comput. Appl.* **2021**, *180*, 103008. [CrossRef]
29. Camargo-Henríquez, I.; Silva, A. An Activity Theory-Based Approach for Context Analysis, Design and Evolution. *Appl. Sci.* **2022**, *12*, 920. [CrossRef]
30. Awareness. Available online: https://en.wikipedia.org/wiki/Awareness (accessed on 28 February 2022).
31. Napoleone, A.; Macchi, M.; Pozzetti, A. A review on the characteristics of cyber-physical systems for the future smart factories. *J. Manuf. Syst.* **2020**, *54*, 305–335. [CrossRef]
32. Harrison, R.; Vera, D.; Ahmad, B. Engineering Methods and Tools for Cyber–Physical Automation Systems. *Proc. IEEE* **2016**, *104*, 973–985. [CrossRef]
33. Model Driven Architecture. Available online: https://www.omg.org/mda/index.htm (accessed on 28 February 2022).
34. The Unified Modeling Language. Available online: https://www.uml.org/ (accessed on 28 February 2022).
35. Thramboulidis, K.C. Using UML in control and automation: A model driven approach. In Proceedings of the 2nd IEEE International Conference on Industrial Informatics, Berlin, Germany, 24–26 June 2004; pp. 587–593.
36. The Systems Modeling Language. Available online: https://sysml.org/ (accessed on 28 February 2022).
37. Shah, A.A.; Kerzhner, A.A.; Schaefer, D.; Paredis, C.J. Multi-view modeling to support embedded systems engineering in SysML. In *Graph Transformations and Model-Driven Engineering*; Springer: New York, NY, USA, 2010; pp. 580–601.
38. Cao, Y.; Liu, Y.; Paredis, C.J. System-level model integration of design and simulation for mechatronic systems based on SysML. *Mechatronics* **2011**, *21*, 1063–1075. [CrossRef]
39. Petri Net. Available online: https://en.wikipedia.org/wiki/Petri_net (accessed on 28 February 2022).
40. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE.* **1989**, *77*, 541–580. [CrossRef]
41. Zuberek, W.M. Timed Petri nets definitions, properties, and applications. *Microelectron. Reliab.* **1991**, *31*, 627–644. [CrossRef]
42. Zuberek, W.M. Timed Petri nets and preliminary performance evaluation. In Proceedings of the 7th Annuual Symposium on Computer Architecture, La Baule, France, 6–8 May 1980; pp. 88–96.
43. Sifakis, J. Petri nets for performance evaluation. Measuring Modelling and Evaluating Computer Systems. In Proceedings of the 3rd International Symposium IFIP Working Group 7.3, Bonn, Germany, 3–5 October 1977; pp. 75–93.
44. Marsan, M.A.; Balbo, G.; Conte, G. A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.* **1984**, *2*, 198–199.
45. Molloy, M.K. Discrete Time Stochastic Petri Nets. *IEEE Trans. Softw. Eng.* **1985**, *11*, 417–423. [CrossRef]
46. Berthomieu, B.; Menasche, M. An Enumerative Approach for Analyzing Time Petri Nets. In Proceedings of the IFIP 9th World Computer Congress, Paris, France, 19–23 September 1983; pp. 41–46.
47. Berthomieu, B.; Diaz, M. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Softw. Eng.* **1991**, *17*, 259–273. [CrossRef]
48. Berthomieu, B.; Vernadat, F. State Class Constructions for Branching Analysis of Time Petri Nets. In *Tools and Algorithms for the Construction and Analysis of Systems*; TACAS 2003. Lecture Notes in Computer Science; Garavel, H., Hatcliff, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2619, pp. 442–457.
49. Klai, K.; Aber, N.; Petrucci, L. A New Approach to Abstract Reachability State Space of Time Petri Nets. In Proceedings of the 2013 20th International Symposium on Temporal Representation and Reasoning, Pensacola, FL, USA, 26–28 September 2013; pp. 117–124.
50. Lefebvre, D. A Timed Extended Reachability Graph for the simulation and analysis of bounded TPNs. In Proceedings of the European Simulation and Modelling Conference, Lisbon, Portugal, 25–27 October 2017; pp. 33–37.
51. Lefebvre, D. Approximated Timed Reachability Graphs for the robust control of discrete event systems. *Discret. Event Dyn. Syst.* **2019**, *29*, 31–56. [CrossRef]
52. Esparza, J.; Silva, M. Compositional synthesis of live and bounded free choice Petri nets. *Lect. Notes Comput. Sci.* **1991**, *527*, 172–187.
53. Murata, T.; Koh, J. Reduction and expansion of live and safe marked graphs. *IEEE Trans. Circuits Syst.* **1980**, *27*, 68–71. [CrossRef]
54. Marechal, A.; Buchs, D. Generalizing the Compositions of Petri Nets Modules, Application and Theory of Petri Nets and Concurrency. *Fundam. Inform.* **2015**, *137*, 87–116. [CrossRef]
55. Jeng, M.D.; DiCesare, F. A review of synthesis techniques for Petri nets with applications to automated manufacturing systems. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 301–312. [CrossRef]
56. CPLEX Optimizer. Available online: https://www.ibm.com/analytics/cplex-optimizer (accessed on 28 February 2022).

57. Boutamina, S.; Maamri, R. A survey on context-aware workflow systems. In Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication, Batna, Algeria, 23–25 November 2015; pp. 1–6.
58. Kwon, O.B. Modeling and generating context-aware agent-based applications with amended colored Petri nets. *Expert Syst. Appl.* **2004**, *27*, 609–621. [CrossRef]
59. Hu, Z.; Lu, T.; Zhao, Z. Context-aware service system modeling using timed CPN. In Proceedings of the 2013 10th International Conference on Service Systems and Service Management, Hong Kong, China, 17–19 July 2013; pp. 164–169.
60. Xing, Z.; Hong, Z.; Yulong, L. A Petri-Net Based Context-Aware Workflow System for Smart Home. In Proceedings of the 2012 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, Shanghai, China, 21–25 May 2012; pp. 2336–2342.
61. Han, S.; Youn, H.Y. Modeling and Analysis of Time-Critical Context-Aware Service Using Extended Interval Timed Colored Petri Nets. *IEEE Trans. Syst. Man Cybern.—Part A Syst. Hum.* **2012**, *42*, 630–640. [CrossRef]
62. Hsieh, F.S. Robust Supervisory Control for Cyber-Physical Systems based on Discrete Timed Petri nets. In Proceedings of the 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Virtual Conference, 26–29 January 2022.