Automaton-based timed supervisory control for operational planning and scheduling under multiple job deadlines

Lin, Liyong; Shehabinia, Ahmad Reza; Brandin, Bertil; Su, Rong.

2014

Lin, L., Shehabinia, A. R., Su, R., & Brandin, B. (2014). Automaton-based timed supervisory control for operational planning and scheduling under multiple job deadlines. 2014 IEEE 53rd Annual Conference on Decision and Control (CDC), 5943-5948.

https://hdl.handle.net/10356/81363

https://doi.org/10.1109/CDC.2014.7040319

Automaton-based Timed Supervisory Control for Operational Planning and Scheduling under Multiple Job Deadlines

Liyong Lin, Ahmad Reza Shehabinia, Rong Su, Bertil Brandin

***Abstract*—In this paper we model an operational planning and scheduling problem under multiple job deadlines in a time-weighted automaton framework. We first present a method to determine whether all given job specifications and deadlines can be met by computing a supremal controllable job satisfaction sublanguage. When this supremal sublanguage is not empty, we compute one of its controllable sublanguages that ensures the minimum total job earliness by adding proper delays. When this supremal sublangauge is empty, we will determine the minimal sets of job deadlines that need to be relaxed.**

Index Terms – time-weighted automaton, controllability, scheduling, earliness, timed supervisory control

I. Introduction

Job deadline satisfaction problems have been extensively studied in the operations research community due to their important applications in manufacturing and logistics. There is an enormous number of publications on this subject. For example, [1] provides a thorough account on job shop scheduling problems with deadlines. [2] gives a survey of scheduling with controllable processing times, in order to meet deadlines in an optimal manner, e.g., reducing job earliness. When some deadlines are deemed infeasible to be fulfilled, relaxations can be considered, usually in an optimal manner. For example, in [3] the authors consider a single machine scheduling problem with common due dates, and the performance is measured by the minimization of the sum of earliness and tardiness penalties of the jobs. In [4] the authors analyze a problem in which the cost function is the weighted sum of the quadratic earliness and tardiness of jobs and of quadratic deviations between pre-defined nominal unitary processing times and the actual ones. In operations research usually mathematical and constraint programming models are used.

Recently, researchers in the supervisory control community have been formulating the job deadline satisfaction problems in automaton (or language) based models, motivated by the belief that supervisory control theory [5] [6] can ensure optimal performance while guaranteeing operational safety and liveness. In [7] the authors adopt the Brandin-Wonham timed control paradigm [8] to deal with supervisory control for job deadlines in cluster tools applications, whose modeling formalism follows the Ostroff's semantics for timed transition models [9]. In [10] the authors adopt the timed automaton modeling formalism [11] and use reachability analysis techniques to determine deadline satisfaction in an industrial case study. In [12] a new time optimal supervisory control paradigm is introduced, in which a plant is modeled as a time-weighted automaton and a logic requirement is modeled as an unweighted automaton. The heap-of-pieces theory [13] is used to describe execution time of each trajectory in a quasi-concurrent setup, i.e., executions of different transitions in the trajectory can be overlapped but their starting moments must be sequentially ordered. The goal in [12] is to achieve minimum makespan.

In this paper we adopt the time-weighted automaton modeling formalism introduced in [12] to address the job deadline satisfaction problems. The basic setup consists of a plant modeled as a time-weighted automaton, a logic requirement language that takes care of general logic constraints about operational safety and progress, and a collection of job requirements associated with relevant job deadlines. We first tackle the problem of determining the least restrictive controllable sublanguage (denoted as S for easy reference) of the plant that satisfies all requirements and deadlines. When S exists, our second problem is to determine an optimal strategy to add delays to relevant transitions of the plant so that there exists the least restrictive controllable sublanguage that satisfies all requirements and deadlines with minimum job earliness. By solving a constraint optimization problem we show that the set of all optimal delays can be computed. When S does not exist due to infeasibility of fulfilling some deadlines, our next problem is to determine how to relax some deadlines and in order to minimize the impact of deadline relaxation, we present an algorithm to determine the minimal sets of deadlines which need to be relaxed.

We organize the paper as follows. Firstly, we review some basic concepts of time-weighted automaton formalism introduced in [12] in Section II. Then we present the baseline problem of synthesizing the supremal controllable job-satisfaction sublanguages in Section III. After that, we bring in the problem of delay addition for minimum job earliness in Section IV, and the problem of relaxing job deadlines in Section V. After providing a simple illustration example in Section VI, conclusions are drawn in Section VII.

This work is supported by Singapore Ministry of Education Tier 1 Academic Research Fund RG84/13.

Liyong Lin, Ahmad Reza Shehabinia, Rong Su are affiliated with School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. Emails: {llin5,ahmadrez1}@e.ntu.edu.sg, rsu@ntu.edu.sg.

Bertil Brandin is affiliated with ABB Pte. Ltd, 2 Ayer Rajah Crescent, Singapore 139935. Email: bertil.brandin@sg.abb.com.

II. Time-weighted automaton

We assume that the reader is familiar with concepts and operations in supervisory control theory [5] and follow the notations in [14]. Given two strings $s, t \in \Sigma^*$, we write $s \leq t$ to denote s being a *prefix substring* of t. Given a language $L \subseteq \Sigma^*$ we use $\overline{L}$ to denote its *prefix closure*. Given two languages $L, L' \subseteq \Sigma^*$, let LL' denote the concatenation of L and L'. Given an arbitrary set S we use $|S|$ to denote its cardinality. We write $P : \Sigma^* \to \Sigma'^*$ for the *natural projection* with respect to (Σ, Σ'), where $\Sigma' \subseteq \Sigma$. The projection over a language $P(L)$ and the inverse image of P, denoted as P^{-1}, are defined accordingly. Given $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, we write $L_1 || L_2 = P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$ for its *synchronous product*.

Let $\mathbb{R}$, $\mathbb{R}^+$ and $\mathbb{N}$ be the collections of reals, positive reals and natural numbers respectively. A *finite-state time-weighted automaton* is a 3-tuple $(G = (X, \Sigma, \xi, x_0, X_m), f, h)$, where G is a finite-state automaton, the (partial) weight function $f : X \times \Sigma \to \mathbb{R}^+$ assigns to each transition of G a finite positive real, which denotes the duration required for the corresponding transition to be completed. The mutual exclusion relation $h \subseteq \Sigma \times \Sigma$ is reflexive and symmetric. A pair $(\sigma, \sigma') \in h$ if one event is under execution, the other event cannot be fired. For notational simplicity, we write $(\sigma, \sigma') \in h$ to denote both $(\sigma, \sigma') \in h$ and $(\sigma', \sigma) \in h$. We use $\xi(x, \sigma)!$ to denote that the transition $\xi(x, \sigma)$ is defined. Let $L(G) := \{s \in \Sigma^* \mid \xi(x_0, s)!\}$ be the *closed* behavior of G and $L_m(G) := \{s \in L(G) \mid \xi(x_0, s) \in X_m\}$ be the *marked* behavior of G. Let $\phi(\Sigma)$ and $\varphi(\Sigma)$ denote respectively the set of finite-state automata, and the set of finite-state time-weighted automata, whose alphabets are Σ.

Definition 1: [12] Given a time-weighted automaton $(G, f, h) \in \varphi(\Sigma)$, let $s \in L(G)$. Suppose $s = \sigma_1 \cdots \sigma_n$ for $n \in \mathbb{N}$. Let $s_q := \sigma_1 \cdots \sigma_q \leq s$. If $q = 0$ then $s_q := \epsilon$.

1) A *time-stamp* of s with respect to (G, f, h) is a nondecreasing list of nonnegative reals $\rho = (t_k^s \in \mathbb{R}^+ \cup \{0\} \mid k = 1, \cdots, n)$, where for all $q, v \in \{1, \cdots, n\}$,
$$q < v \wedge (\sigma_q, \sigma_v) \in h \Rightarrow t_q^s + f(\xi(x_0, s_{q-1}), \sigma_q) \leq t_v^s.$$
Each t_k^s denotes a starting moment of firing σ_k in s.
2) Let $\Theta_{G,f,h}(s)$ be the set of all time-stamps of s, and
$$\upsilon_{G,f,h}(s) := \min_{\rho \in \Theta_{G,f,h}(s)} \max\{t_1^s + f(x_0, \sigma_1), \cdots, t_n^s + f(\xi(x_0, s_{n-1}), \sigma_n)\}$$
is called *execution time* of s with respect to (G, f, h). As a convention, $\upsilon_{G,f,h}(\epsilon) := 0$. $\square$

We present an approach based on the heaps-of-pieces theory to calculate the string execution time $\upsilon_{G,f,h}(s)$.

Given $(G = (X, \Sigma, \xi, x_0, X_m), f, h)$, we use $\varpi : L(G) \to (X \times \Sigma)^*$ to map each string in $L(G)$ to the (unique) trajectory (or path) in G, where $\varpi(\epsilon) := \epsilon$. Let $\mathcal{T} := \{(x, \sigma) \in X \times \Sigma \mid \xi(x, \sigma)!\}$ be the set of all transitions in G. By [12] we know that the mutual exclusion relation h induces a resource set $\mathcal{R}$ and a map $R : \mathcal{T} \to 2^{\mathcal{R}}$, which maps each transition $\tau \in \mathcal{T}$ to its set of resources. Let $n_h = |\mathcal{R}|$. Then there exists a morphism $\hat{\mathcal{M}}_{G,f,h} : \mathcal{T}^* \to \mathbb{R}_{max}^{n_h, n_h}$, where $\mathbb{R}_{max}^{n_h, n_h}$ is the collection of all matrices whose dimensions are $n_h \times n_h$ such that $\hat{\mathcal{M}}_{G,f,h}(\epsilon)$ is defined as the unit matrix $\mathbf{I}_{n_h \times n_h}$, i.e., all diagonal entries are 0 and all other entries are $-\infty$, and for each $\tau \in \mathcal{T}$,
$$\hat{\mathcal{M}}_{G,f,h}(\tau)_{qv} := \begin{cases} 0 & \text{if } q = v,\ q \notin R(\tau), \text{ or } q \in R(\tau),\ v \notin R(\tau) \\ f(\tau) & \text{if } q, v \in R(\tau) \\ -\infty & \text{otherwise} \end{cases}$$
In [12] it has been shown that
$$\upsilon_{G,f,h}(s) = \mathbf{1}_{n_h}^t \hat{\mathcal{M}}_{G,f,h}(\varpi(s)) \mathbf{1}_{n_h}, \qquad (1)$$
where $\mathbf{1}_{n_h} \in \mathbb{R}_{max}^{n_h}$ is the n_h-dimensional column vector, whose entries are all 0, and $\mathbf{1}_{n_h}^t$ is the transpose of $\mathbf{1}_{n_h}$. When the context is clear, we use $\upsilon_f(s)$ to denote $\upsilon_{G,f,h}(s)$.

Let $\Sigma = \Sigma_c \dot{\bigcup} \Sigma_{uc}$, where Σ_c and Σ_{uc} denote respectively the sets of *controllable* events and *uncontrollable* events.

Definition 2: [5] Given $G \in \phi(\Sigma)$, $K \subseteq L_m(G)$ is *controllable* with respect to G if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. $\square$

III. Supremal controllable job satisfaction sublanguage

In this section, we propose an algorithm to compute the supremal controllable sublanguage that satisfies the job requirements and deadlines. To that end, we formally introduce the notion of job requirements and their deadlines below.

Let $\mathcal{E}_T = \{(E_i, d_i) \in 2^{\Sigma_i^*} \times \mathbb{R}^+ \mid i \in I\}$, where E_i is a regular language, called *job requirement*, over $\Sigma_i \subseteq \Sigma$ and $d_i \in \mathbb{R}^+$ is its *deadline* (or *due date*). Here I is a finite index set and each tuple $(E_i, d_i) \in \mathcal{E}_T$ is interpreted as follows: each job requirement E_i has to be satisfied within its pre-specified due date d_i. In this work, the completion of any *process* $s \in E_i$ is said to satisfy or complete the job requirement E_i. Since an operational sequence generated by the plant may involve events that contribute to the satisfaction of different jobs, we also say an operation sequence $s \in \Sigma^*$ completes job E_i if $P_i(s) \in E_i$, where $P_i : \Sigma^* \to \Sigma_i^*$ is the natural projection. We define the job E_i execution time by s (under f) as
$$t_{E_i,f}(s) = \min_{s' \leq s \wedge P_i(s') = P_i(s)} \upsilon_f(s') = \max_{s' \leq s \wedge s' \in \Sigma^* \Sigma_i} \upsilon_f(s'),$$
i.e., the job E_i execution time by s is the duration from the beginning of s to the last event of s in Σ_i. For example, if $\Sigma = \{a, b, c, d\}$, $\Sigma_i = \{a, c\}$, $s = abcd$, then $t_{E_i,f}(s) = \upsilon_f(abc)$ is the job E_i execution time by s. Moreover, if s completes E_i, i.e., $P_i(s) \in E_i$, then $t_{E_i,f}(s)$ is said to be the *job E_i completion time by s (under f)*. We assume $\Sigma = \bigcup_{i \in I} \Sigma_i$ and allow $\Sigma_i \cap \Sigma_j \neq \varnothing$ for $i \neq j$. Hence each event executed by the plant contributes to the completion of some jobs and we allow the same event to contribute to the completion of several different jobs. Let $W_f(\mathcal{E}_T) := \{s \in \Sigma^* \mid \forall i \in I, P_i(s) \in E_i \wedge t_{E_i,f}(s) \leq d_i\}$ be the (possibly empty) collection of all strings which satisfy all job requirements and the corresponding job deadlines. $W_f(\mathcal{E}_T)$ is a finite language since the transition duration of each event is positive and $\Sigma = \bigcup_{i \in I} \Sigma_i$. In addition to the job requirements, there is a *general logic requirement* $E \subseteq \Sigma^*$ used to specify the safety properties and the progress properties. We now bring in the notion of a *controllable job satisfaction sublanguage*.

Given a time-weighted plant $(G, f, h) \in \varphi(\Sigma)$, the set of jobs and deadlines $\mathcal{E}_T = \{(E_i, d_i) \in 2^{\Sigma_i^*} \times \mathbb{R}^+ \mid i \in I\}$ and a general logic requirement $E \subseteq \Sigma^*$, let

$$\mathcal{C}(G, f, h, E, \mathcal{E}_T) := \{K \subseteq L_m(G) \cap E \cap W_f(\mathcal{E}_T) \mid \\ K \text{ is controllable w.r.t. } G\}$$

be the collection of all controllable sublanguages[*] of $L_m(G)$ satisfying all requirements and the job deadlines, i.e., the collection of *controllable job satisfaction sublanguages* of (G, f, h) under E and $\mathcal{E}_T$. There exists a unique element $K_* \in \mathcal{C}(G, f, h, E, \mathcal{E}_T)$ such that

$$\forall K \in \mathcal{C}(G, f, h, E, \mathcal{E}_T), K \subseteq K_*$$

We call K_* the *supremal controllable job satisfaction sublanguage of* (G, f, h) *under* E *and* $\mathcal{E}_T$, denoted as sup$\mathcal{C}(G, f, h, E, \mathcal{E}_T)$. The first problem is stated below.

Problem 1: Given a plant (G, f, h), requirements E and $\mathcal{E}_T$, compute sup$\mathcal{C}(G, f, h, E, \mathcal{E}_T)$. □

A solution to Problem 1 is given by the following algorithm.

Algorithm 1 Compution of sup$\mathcal{C}(G, f, h, E, \mathcal{E}_T)$

```
1) Input: A centralized model $(G, f, h) \in \varphi(\Sigma)$, requirements $E \subseteq \Sigma^*$ and $\mathcal{E}_T$ with $I = \{1, 2, \ldots, k\}$.
2) Construct a trim tree-structured automaton for $H = L_m(G) \cap E \cap W_f(\mathcal{E}_T)$ by constructing a tree-structured automaton for $L_m(G) \cap E \cap \|_{i \in I} E_i$ in breadth-first order with the following modifications:
    a) Each state $q$ is maintained with a $(k+2)$-tuple $(\hat{\mathcal{M}}_{G,f,h}(s), \upsilon_f(s), t_1(q), t_2(q), \ldots, t_k(q))$, where $s$ is the unique string labeling the path from the root state $q_0$ to state $q$. Initially, $t_1(q_0) = t_2(q_0) = \ldots = t_k(q_0) = 0$.
    b) For the immediate successor state $q'$ of $q$ via $\sigma$, let $\hat{\mathcal{M}}_{G,f,h}(s\sigma) := \hat{\mathcal{M}}_{G,f,h}(s)\hat{\mathcal{M}}_{G,f,h}(\sigma)$ and $\upsilon_f(s\sigma) := \mathbf{1}_{n_h}^t \hat{\mathcal{M}}_{G,f,h}(s\sigma)\mathbf{1}_{n_h}$. If $\sigma \in \Sigma_i$, let $t_i(q') := \upsilon_f(s\sigma)$, else let $t_i(q') := t_i(q)$. If $\exists i \in I, t_i(q) \geq d_i$, the subtree rooted at state $q$ is cut.
    c) Repeat step b) for each reached but unexplored state $q$.
3) Output: The supremal sublanguage $K_* \subseteq H$ that is controllable w.r.t. $G$.
```

Here $t_i(q)$ is interpreted as the job E_i execution time $t_{E_i,f}(s)$, where s is the unique string labeling the path from the root state q_0 to state q. Since H is a finite language, step 2) of the above algorithm clearly terminates. We immediately have the following result.

Theorem 1: Let K_* be the output of Algorithm 1. Then $K_* = \text{sup}\mathcal{C}(G, f, h, E, \mathcal{E}_T)$. □

[*]We consider the framework of marking non-blocking supervisory control [14]. The work could be easily adapted to dealing with the non-marking non-blocking supervisory control framework.

IV. Supremal minimum-earliness controllable job satisfaction sublanguage

In Problem 1, if sup$\mathcal{C}(G, f, h, E, \mathcal{E}_T) \neq \varnothing$, it is desirable to synthesize a non-empty controllable sublanguage with minimum job earliness, by adding proper delays to the occurrences of some controllable transitions. To formalize this idea we need to introduce the concept of *earliness of job* E_i *completion (under* f*)*.

Definition 3: Given a string $s \in \Sigma^*$ and a job requirement (E_i, d_i) such that $P_i(s) \in E_i$ and $t_{E_i,f}(s) \leq d_i$. The *earliness* of completing E_i by s is defined as $e_{i,f}(s) = d_i - t_{E_i,f}(s)$. Accordingly, the *earliness* of completing $\mathcal{E}_T$ by s is defined as $e_f(s) = \sum_{i \in I} e_{i,f}(s)$, if $s \in W_f(\mathcal{E}_T)$. □

Definition 4: Given a language $K \subseteq W_f(\mathcal{E}_T)$, the *earliness* of K w.r.t. $\mathcal{E}_T$ is defined as $e_f(K) := \max_{s \in K} e_f(s)$. □

The earliness of the empty set is defined to be zero, i.e., $e_f(\varnothing) := 0$. If sup$\mathcal{C}(G, f, h, E, \mathcal{E}_T) \neq \varnothing$, it is always possible to compute a non-empty controllable sublanguage with minimum job earliness by adding delays. To introduce delays to the plant model we have the following construction. For a given (G, f, h), let $D : X \times \Sigma \to \mathbb{R}^+ \cup \{0\}$ be a partial weight function such that if $\xi(x, \sigma)!$, then $D(x, \sigma) = 0$ for $\sigma \in \Sigma_{uc}$ and $D(x, \sigma) \in \mathbb{R}^+ \cup \{0\}$ for $\sigma \in \Sigma_c$; D is undefined elsewhere. Let $\mathbf{DC}$ be the collection of delays that satisfy above constraint. For each $D \in \mathbf{DC}$, the firing duration of each transition $(x, \sigma) \in \mathcal{T}$ is extended to $(f + D)(x, \sigma) := f(x, \sigma) + D(x, \sigma)$, where $f + D : X \times \Sigma \to \mathbb{R}^+$ is an *extended (partial) weight function* parameterized by $D \in \mathbf{DC}$. Recall that, when there is no delay, to compute string execution time $\upsilon_{G,f,h}(s)$ we need to bring in an induced morphism $\hat{\mathcal{M}}_{G,f,h}$. After introducing the delay D, the induced morphism becomes $\hat{\mathcal{M}}_{G,f+D,h}$. Here for each $\tau \in \mathcal{T}$ we have

$$(\hat{\mathcal{M}}_{G,f+D,h}(\tau))_{qv} := \\ \begin{cases} 0 & \text{if } q = v,\ q \notin R(\tau), \text{ or } q \in R(\tau),\ v \notin R(\tau) \\ (f + D)(\tau) & \text{if } q, v \in R(\tau) \\ -\infty & \text{otherwise} \end{cases}$$

which can be used in computing the new string execution time $\upsilon_{G,f+D,h}(s)$. The computation of earliness is also affected accordingly and changed from e_f to e_{f+D} after adding the delay. The existence of the *supremal minimum-earliness controllable job satisfaction sublanguage* is ensured by the following proposition.

Proposition 1: Let $D \in \mathbf{DC}$ be a delay function. If sup$\mathcal{C}(G, f + D, h, E, \mathcal{E}_T) \neq \varnothing$, then there exists a non-empty set $\hat{K}_* \in \mathcal{C}(G, f + D, h, E, \mathcal{E}_T)$ such that for any non-empty $K \in \mathcal{C}(G, f + D, h, E, \mathcal{E}_T)$ the following holds,

1) $e_{f+D}(\hat{K}_*) \leq e_{f+D}(K)$,
2) $e_{f+D}(K) = e_{f+D}(\hat{K}_*) \Rightarrow K \subseteq \hat{K}_*$. □

We call $\hat{K}_*$ the *supremal minimum-earliness controllable job satisfaction sublanguage of* $(G, f + D, h)$ *under* E *and*

$\mathcal{E}_T$, denoted as $\sup\mathcal{CF}(G, f+D, h, E, \mathcal{E}_T)$. In the remaining of this paper, when we write $\sup\mathcal{CF}(G, f+D, h, E, \mathcal{E}_T)$, we implicitly assume $\sup\mathcal{C}(G, f+D, h, E, \mathcal{E}_T) \neq \varnothing$ and thus $\sup\mathcal{CF}(G, f+D, h, E, \mathcal{E}_T)$ exists. The problem of ensuring minimum earliness of job completions by adding proper delays is formalized below.

Problem 2: If Problem 1 has a non-empty solution, i.e., $\sup\mathcal{C}(G, f, h, E, \mathcal{E}_T) \neq \varnothing$, then compute a delay function $D^* \in \mathbf{DC}$ such that $e_{f+D^*}(\sup\mathcal{CF}(G, f+D^*, h, E, \mathcal{E}_T)) = \min_{D\in\mathbf{DC}} e_{f+D}(\sup\mathcal{CF}(G, f+D, h, E, \mathcal{E}_T))$. □

The difficulty of this problem lies in the fact that the underlying set $\mathcal{C}(G, f+D, h, E, \mathcal{E}_T)$ of controllable job satisfaction sublanguage depends on D. To solve this problem, we introduce the following concept.

Definition 5: A non-empty controllable sublanguage K_1 of K, where $K \subseteq L_m(G)$, with respect to G, if it exists, is said to be minimal if for any non-empty controllable sublanguage $K_2 \subseteq K$, $K_2 \subseteq K_1$ implies $K_2 = K_1$. □

Given a finite language $K \subseteq L_m(G)$, let $\mathbf{CL}(K)$ denote the collection of all non-empty minimal controllable sublanguages contained in K with respect to G. We need the following definitions and operations. A state x in a finite state automaton is said to be a controllable (respectively, an uncontrollable) state if all the transitions out of x are controllable (respectively, uncontrollable). It is said to be a mixed state if there are both controllable and uncontrollable transitions out of state x. The controllable transitions out of each mixed state are removed and each mixed state is thus transformed into an uncontrollable state. It is easy to see that this preprocessing stage does not affect the computation of $\mathbf{CL}(K)$. Given a set of languages $\mathcal{L}$, we define a new set of languages $a\mathcal{L} := \{aL \mid L \in \mathcal{L}\} \subseteq 2^{\Sigma^*}$ for each $a \in \Sigma$. For two sets of languages $\mathcal{L}_1, \mathcal{L}_2$, we define another set of languages $\mathcal{L}_1 \sqcup \mathcal{L}_2 = \{L_1 \cup L_2 \mid L_1 \in \mathcal{L}_1, L_2 \in \mathcal{L}_2\} \subseteq 2^{\Sigma^*}$. We propose the following algorithm to compute $\mathbf{CL}(K)$ when K itself is controllable with respect to G.

Algorithm 2 Algorithm for Computing $\mathbf{CL}(K)$

```
1) Input: A trim tree-structured automaton $K \in \phi(\Sigma)$
2) Assign each leave node $q$ of $K$ with $\mathcal{L}(q) = \{\{\epsilon\}\}$.
3) For each node $q$ whose children have been assigned a set of languages, do the following until the root node is reached:
    a) if $q$ is a controllable state but not a final state, such that $\{(q, a_1, q_1), (q, a_2, q_2), \ldots, (q, a_k, q_k)\}$ are all its branches, then $\mathcal{L}(q) := \bigcup_{i\in[1,k]} a_i\mathcal{L}(q_i)$;
    b) if $q$ is both a controllable state and a final state, then $\mathcal{L}(q) := \bigcup_{i\in[1,k]} a_i\mathcal{L}(q_i) \cup \{\{\epsilon\}\}$;
    c) if $q$ is an uncontrollable state, then $\mathcal{L}(q) := \bigsqcup_{i\in[1,k]} a_i\mathcal{L}(q_i)$.
4) Output $\mathcal{L}(q_0)$, where $q_0$ is the root of $K$.
```

Proposition 2: When K is controllable with respect to G, $\mathbf{CL}(K) = \mathcal{L}(q_0)$. □

With the above technical preparation, a solution to Problem 2 is provided in Algorithm 3. It is likely that two different delays lead to the same earliness associated with two different supremal minimum-earliness controllable job satisfaction sublanguages. Algorithm 3 indeed computes the set of all such optimal delays. This can be seen from the following lemma, which is used to prove the theorem.

Lemma 1: Let $s \in L_m(G)$ be a string. For each $j \in [1, n_h]$, there exist $c_j \in \{0,1\}^h$ and $c_{j,0} \in \mathbb{R}^+ \cup \{0\}$, where $h = |\mathcal{T}|$, such that $t_{E_i, f+D}(s) = \max_{j\in[1,n_h]}(c_j^T D + c_{j,0})$, where D is a h-tuple vector corresponding to the delay function. □

Algorithm 3 Algorithm for Computing Optimal Delays

```
1) Input: $K = \sup\mathcal{C}(G, f, h, E, \mathcal{E}_T)$ from Problem 1.
2) Compute $\mathbf{CL}(K)$ using Algorithm 2.
3) For each $L \in \mathbf{CL}(K)$, solve the following optimization problem:
       $\min_{D\in\mathbf{DC}} e_{f+D}(L) = \min_{D\in\mathbf{DC}} \max_{s\in L} e_{f+D}(s)$
   subject to
       $\bigwedge_{s\in L} \bigwedge_{i\in I} t_{E_i, f+D}(s) \leq d_i$
   Let $\mathbf{DC}^*(L)$ denote the collection of optimal delays for $L$ and $e^*(L)$ denote the minimum earliness that is achievable by $L$ with delays.
4) Let $\mathbf{OL} := \{L \in \mathbf{CL}(K) \mid \forall L' \in \mathbf{CL}(K), e^*(L) \leq e^*(L')\}$. Set $\mathbf{DC}^* = \bigcup_{L\in\mathbf{OL}} \mathbf{DC}^*(L)$.
5) Output: $\mathbf{DC}^*$.
```

Theorem 2: Given a plant (G, f, h) and requirements E and $\mathcal{E}_T$, if $\sup\mathcal{C}(G, f, h, E, \mathcal{E}_T) \neq \varnothing$, then Algorithm 3 solves Problem 2. Indeed, it computes the set of all optimal delays. □

For each $D \in \mathbf{DC}^*$, we could also compute the corresponding supremal minimum-earliness controllable job satisfaction sublanguage $\sup\mathcal{CF}(G, f+D, h, E, \mathcal{E}_T)$ by Proposition 1.

V. Relaxations of Job Deadlines

If $\sup\mathcal{C}(G, f, h, E, \mathcal{E}_T) = \varnothing$, then there is no controllable sublanguage of $L_m(G)$ that satisfies all jobs and the imposed deadlines. If the supremal controllable sublanguage of $L_m(G) \cap E \cap (||_{i\in I} E_i)$ is non-empty, then it means that some job deadlines need to be relaxed in order to have a non-empty controllable job satisfaction sublanguage. Thus it is practically important to know the maximal sets of job deadlines that can be met. Dually, we would like to compute the minimal sets of job deadlines that need to be relaxed. Formally, the following problem is considered:

Problem 3: Given a plant (G, f, h) and requirements E, $\mathcal{E}_T$, determine all the subsets of job requirements $I_1 \subseteq I$ such that

1) $\sup\mathcal{C}(G, f, h, E\cap(||_{i\in I_1} E_i), \{(E_i, d_i) \mid i \in I-I_1\}) \neq \varnothing$
2) $\forall I_2 \subseteq I_1, (\sup\mathcal{C}(G, f, h, E \cap (||_{i\in I_2} E_i), \{(E_i, d_i) \mid i \in I - I_2\}) \neq \varnothing \Rightarrow I_1 = I_2)$. □

Note that $\sup\mathcal{C}(G, f, h, E \cap (||_{i\in I_1} E_i), \{(E_i, d_i) \mid i \in I - I_1\})$, denoted as $\sup\mathcal{CR}(I_1)$ for convenience, is in general an infinite language, since the new job requirements $\mathcal{E}'_T = \{(E_i, d_i) \mid i \in I - I_1\}$ does not satisfy $\Sigma = \bigcup_{i\in I-I_1} \Sigma_i$. The tree-structured automaton construction method does not work in this case. Let $W_f(\mathcal{E}'_T) = \{s \in \Sigma^* \mid \forall i \in I - I_1, P_i(s) \in E_i \wedge t_{E_i,f}(s) \leq d_i\}$. We propose the following algorithm, which is quite similar to Algorithm 1, to compute $\sup\mathcal{CR}(I_1)$.

Algorithm 4 Computation of $\sup\mathcal{CR}(I_1)$

```
1) Input: A centralized model $(G, f, h) \in \varphi(\Sigma)$, requirements $E \subseteq \Sigma^*$, $\{E_i \mid i \in I_1\}$ and $\mathcal{E}'_T$.
2) Compute the minimal finite automaton $G_L$ for $L_m(G) \cap E \cap \|_{i\in I} E_i$.
3) Construct a finite automaton $A_T$ over $\Sigma$ by constructing a tree-structured automaton for $\Sigma^*$ with the following modifications, such that each string accepted by $A_T$ satisfies the job requirements $I - I_1 = \{1, 2, \ldots, k\}$:
    a) Each state $q$ is maintained with a $(k+2)$-tuple $(\hat{\mathcal{M}}_{G,f,h}(s), \upsilon_f(s), t_1(q), t_2(q), \ldots, t_k(q))$, where $s$ is the unique string labeling the path from the root state $q_0$ to state $q$. Initially, $t_1(q_0) = t_2(q_0) = \ldots = t_k(q_0) = 0$.
    b) For the immediate successor state $q'$ of $q$ via $\sigma$, let $\hat{\mathcal{M}}_{G,f,h}(s\sigma) := \hat{\mathcal{M}}_{G,f,h}(s)\hat{\mathcal{M}}_{G,f,h}(\sigma)$ and $\upsilon_f(s\sigma) := \mathbf{1}^t_{n_h}\hat{\mathcal{M}}_{G,f,h}(s\sigma)\mathbf{1}_{n_h}$. If $\sigma \in \Sigma_i$, let $t_i(q') := \upsilon_f(s\sigma)$, else let $t_i(q') := t_i(q)$. If $\forall i \in I - I_1, \upsilon_f(s\sigma) \geq d_i$, loop at state $q'$ for each $\sigma \in \Sigma - \bigcup_{i\in I-I_1} \Sigma_i$. If $\exists i \in I - I_1, t_i(q) \geq d_i$, the subtree rooted at state $q$ is cut.
    c) Repeat step b) for each reached but unexplored state $q$.
4) Compute $H := L_m(G_L \| A_T)$.
5) Output: The supremal sublanguage $K_* \subseteq H$ that is controllable w.r.t. $G$.
```

Proposition 3: The step 2) of Algorithm 4 terminates and the output K_* satisfies $K_* = \sup\mathcal{CR}(I_1)$. □

Problem 3 is now solved by examining all the elements of the lattice $(2^I, \cup, \cap, I, \emptyset)$, starting from the bottom element $\emptyset$ to the top element I. Once an element, i.e, a subset of I, is found to be a solution, there is no need to examine all the greater elements.

VI. Simple job-shop example

To illustrate above setup, a simplified job shop depicted in Fig. 1 is used.

It consists of one I/O (Input/Output) buffer for feeding raw materials and dropping out those processed ones. There are two machines (M1 and M2) that process the raw materials,

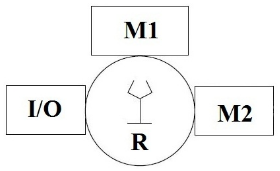

Fig. 1. Two-Machine Job-Shop Environment

and one robot that performs tasks of loading and unloading of parts. In this example, we deal with two types of product (A and B) and accordingly, based on product type, machines' actions are differentiated. The timed-weighted model[†] for each component is shown in Fig. 2.

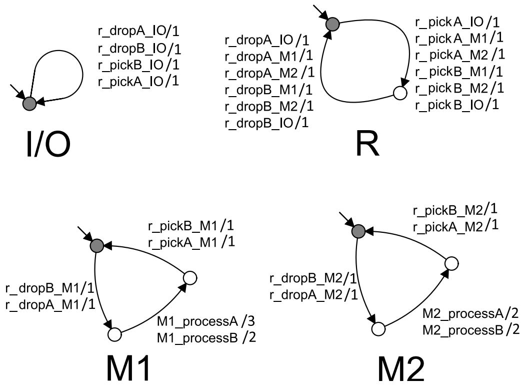

Fig. 2. Component Model

We assume that all events of $\Sigma = \Sigma_{M_1} \cup \Sigma_{M_2} \cup \Sigma_R$ are controllable, and the mutual exclusion relation is $h = \Sigma_{M_1} \times \Sigma_{M_1} \cup \Sigma_{M_2} \times \Sigma_{M_2} \cup \Sigma_R \times \Sigma_R$. In this example, to have the complete products, both parts are required to go through $I/O \to M_1 \to M_2 \to I/O$. Let $\mathcal{E}_T = \{(E_1, 20), (E_2, 12)\}$, where E_1 and E_2 are job requirements for product type A and type B respectively. The logic requirement and job requirements are depicted in Fig. 3. Upon applying Algorithm 1, we can compute the supremal controllable job satisfaction sublanguage, which is depicted in Fig. 4. We can easily see that the result of Problem 1 consists of 5 non-empty minimal controllable sublanguages - each one is a singleton. The completion time of E_1 and E_2 for different strings are as follows: (we order the strings from left to right)

For $i \in [1, 4], t_{J_1,f}(s_i) = 16, t_{J_2,f}(s_i) = 10$ and for string $s_5, t_{J_1,f}(s_5) = 17, t_{J_2,f}(s_5) = 11$.

Clearly $e_f(s_i) = 6$ for all $i \in [1, 4]$ and $e_f(s_5) = 4$, thus $\{s_5\}$ is the supremal minimum-earliness controllable job satisfaction sublanguage. If delay functions are considered, it is easy to see that the achievable minimum earliness is zero. Indeed, it suffices to check that if we choose D^* such that D^*(r_dropB_IO) $= 1$, D^*(r_dropA_IO) $= 3$, and for all $\sigma \in \Sigma - \{$r_dropB_IO, r_dropA_IO$\}, D^*(\sigma) = 0$, then $e_{f+D^*}(\sup\mathcal{CF}(G, f + D^*, h, E, \mathcal{E}_T)) = e_{f+D^*}(\{s_5\}) = 0$.

[†]In this example, the simple automaton model works because the logic behavior will be further constrained by the job requirement in Fig.3.

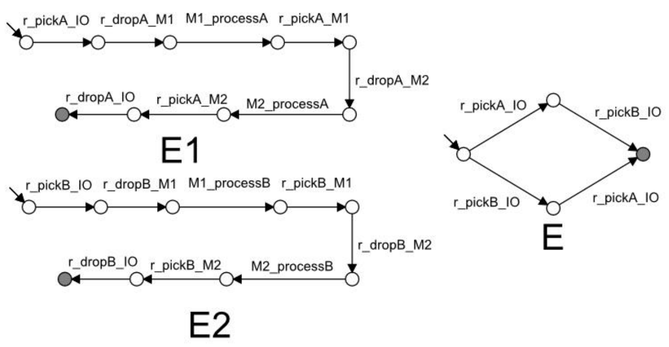

Fig. 3. Logic Requirement and Job Requirements

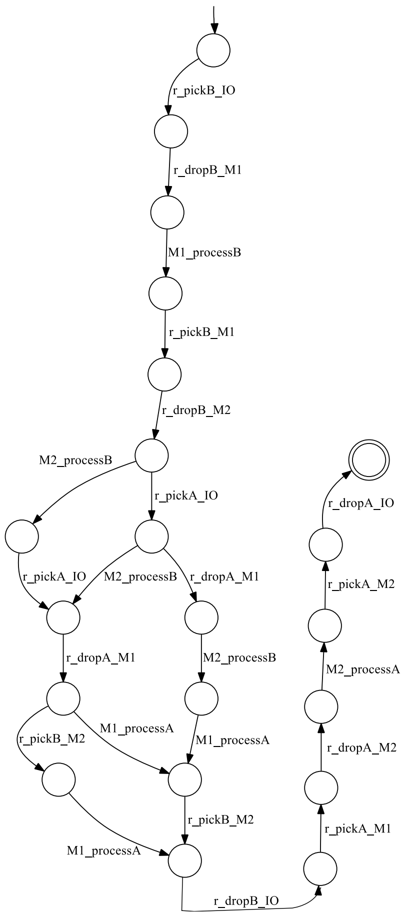

Fig. 4. Supremal Controllable Job Satisfaction Sublanguage

Here, we use events instead of transitions to specify D as there is a clear one to one map between them in this example.

VII. Conclusions

In this paper we address the problem of timed control for multiple job deadlines. We have first introduced the concept of supremal controllable job satisfaction sublanguage, and provided algorithms to compute such a supremal sublanguage. Considering that in practical applications large job earliness is usually undesirable, we have introduced the concept of supremal controllable minimum-earliness job satisfaction sublanguage, which ensures minimum job earliness by adding delays in properly chosen transition firings. In case that there does not exist a non-empty controllable job satisfaction sublanguage, we have proposed an algorithm to compute the minimal sets of deadlines that need to be relaxed.

Acknowledgment: The support for this work from Ministry of Education Academic Tier 1 Research Grant (No. M4011221 RG84/13) is gratefully acknowledged.

References

[1] M. L. Pinedo. Scheduling: Theory, Algorithms, and Systems (3rd edition). *Springer*, 2008.
[2] D. Shabtay and G. Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666, 2007.
[3] C. M. Hino, D. P. Ronconi and A. B. Mendes. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, 160(1):190–201, 2005.
[4] D. Giglio, R. Minciardi, S. Sacone and S. Siri. Optimal control of production processes with variable execution times. *Discrete Event Dynamic Systems*, 19(3):423-448, 2009.
[5] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.
[6] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.
[7] J. E. R. Cury, C. Martinez and M. H. de Queiroz. Scheduling cluster tools with supervisory control theory. *Intelligent Manufacturing Systems*, 11(1): 312–317, 2013.
[8] B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Trans. Autom. Control*, 39(2):329-342, 1994.
[9] J. S. Ostroff. Temporal Logic for Real-Time Systems. *Advanced Software Development Series, ed. J. Kramer. Research Studies Press Limited (distributed by John Wiley and Sons)*, England, 1989.
[10] G. Behrmann, E. Brinksma, M. Hendriks and A. Mader. Production scheduling by reachability analysis - a case study. In *Proc. 19th IEEE International Parallel and Distributed Processing Symposium*, pages 140a–140a, 2005.
[11] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183-235, 1994.
[12] R. Su, J. H. van Schuppen and J. E. Rooda. The synthesis of time optimal supervisors by using heaps-of-pieces. *IEEE Trans. Automatic Control*, 57(1):105-118, 2012.
[13] G. X. Viennot. Heaps of pieces, I: Basic definitions and combinatorial lemmas. *Combinatoire Énumérative*, Labelle and Leroux, Eds., no. 1234 in Lecture Notes in mathematics, pages 321–350, 1997.
[14] W. M. Wonham. *Supervisory Control of Discrete-Event Systems*. Systems Control Group, Dept. of ECE, University of Toronto. URL: www.control.utoronto.ca/DES, 2013.