Mathematical and Computer Modelling 49 (2009) 286–294

Contents lists available at ScienceDirect

Mathematical and Computer Modelling

journal homepage: www.elsevier.com/locate/mcm

Modeling job shop scheduling with batches and setup times by timed Petri nets ☆

Hehua Zhang [a,b,c,*], Ming Gu [b,c]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China

[b] School of Software, Tsinghua University, Beijing 100084, PR China

[c] Key Laboratory for Information System Security, Ministry of Education of China, Beijing, PR China

ARTICLE INFO

Article history:
Received 13 July 2007
Received in revised form 13 February 2008
Accepted 13 March 2008

Keywords:
Petri nets
Job shop
Scheduling
Batch
Setup times

ABSTRACT

Batch and setup times are two important factors in practical job shop scheduling. This paper proposes a method to model job shop scheduling problems including batches and anticipatory sequence-dependent setup times by timed Petri nets. The general modeling method is formally presented. The free choice property of the model is proved. A case study extracted from practical scheduling is given to show the feasibility of the modeling method. Comparison with some previous work shows that our model is more compact and effective in finding the best solution.

1. Introduction

Batch and setup times appear frequently in real-life scheduling applications. A batch represents the number of jobs that are processed at one time. A setup is a certain machine time before processing an operation. In prior research work, setup times are not considered or treated as part of the processing time. However, in many cases, setup times cannot be neglected and different setup times are required depending on the sequences [1,2]. It is necessary to explicitly consider setup times in scheduling.

Much recent work paid attention to batch and setup times in scheduling fields. [3–6] gave a comprehensive survey of the current research work on batch and setup times. So far, there is no breakthrough in job shop scheduling which considers both batch and sequence-dependent setup times simultaneously.

As a powerful and flexible modeling technique, Petri nets are well suited to model the complex constraints in real-life scheduling problems. There is some research work using Petri net-based methods to model and solve scheduling problems [7–11]. However, few considered setup times and batches. [12] gave formal mapping rules from general scheduling problems to timed Petri net models, but it did not consider batch or setup times. [13] considered sequence-dependent setup times but batches have not been taken into account. In addition, its model did not permit anticipatory setups [5], where the machine setup can be started before the corresponding jobs or batches become available on the machine. [14] considered many practical situations including batches, setup times and transportation times. However, the batch term in it means the number of jobs belonging to the same *family* [5]. The real batch in the model is one. On the other hand, it did not consider anticipatory setups either.

☆ This research is sponsored by 973 Program (2004CB719406), NSFC Program (60553002, 60635020) and 863 Program (2007AA04Z1B2) of China.

* Corresponding author at: Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China.
E-mail address: zhang-hh04@mails.tsinghua.edu.cn (H. Zhang).

0895-7177/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2008.03.010

As non-anticipatory setup times may hinder maximal concurrency and thus adversely affect the solution quality in many occasions, we give our method to model job shop scheduling problems by timed Petri nets. Both batch and anticipatory sequence-dependent setup times are considered. The proposed model is proved to be free choice and it bears many advantages through comparing with previous work.

The rest of the paper is organized as follows. Section 2 defines the scheduling problem and timed Petri net. In Section 3, the general modeling method is proposed. We prove the free choice property of the generated model in Section 4. Section 5 presents a case study and comparisons. Finally, we conclude the paper in Section 6.

2. Preliminaries

The scheduling problem is often described by a three-field $\alpha|\beta|\gamma$. The α field describes the machine environment. The β field specifies details of processing characteristics and constraints that may be relevant. The γ field describes the objective to be minimized [15]. We assume that all the jobs arrive at the shop at time 0 and transportation time among machines is negligible. The problem considered in this paper is defined as follow.

The α field is represented by:

– J_m: a job shop with m machines.

The β field is denoted as:

– F: The n jobs can be partitioned into F families, $F \geq 1$. The jobs in the same family own the the same route.
– R_i: The predetermined route of the ith family. The same machines occurs at most once in a specific route.
– L_i: The number of jobs belonging to the ith family. The jobs which belong to the same family share the same predetermined route.
– B_i: The batch size of the jobs that can be processed at one time which belong to the same ith family.
– $ST_{k,i}$: The setup time for a machine k precedes processing family i. A certain setup time is only needed for machine k if the former processed batch does not belong to family i. The definition $ST_{k,i}$ is widely used in real-life scheduling problems. It is a special case of general sequence-dependent setup time $ST_{k,i,j}$, which represents the setup time required for processing family j on machine k immediately after family i has been processed on it.
– $P_{i,j}$: The processing time of a single job which belongs to ith family on machine j. In this way, the batch processing time of a batch belongs to ith family equals to $P_{i,j} * B_i$.

The γ field is characterized as:

– To minimize the makespan for all the jobs.

The timed Petri net and its enabling and firing rule are defined as follows.

Definition 1 (*Timed Petri Net*). A timed Petri net is a 6-tuple $\mathrm{TPN} = (P, T, I, O, M_0, D)$, where P is a finite set of *places*; T is a finite set of *transitions*, $P \cup T \neq \varnothing$, and $P \cap T = \varnothing$; $I : (P \times N) \to N$ is an *input function* that defines directed arcs from places to transitions, where N is a set of nonnegative integers; $O : (T \times P) \to N$ is an *output function* that defines directed arcs from transitions to places; $M_0 : P \to N$ is the *initial markings*; $D : T \to N$ is a function which defines the *time delay* of each transition.

A marking M is a function from the set of places P to the nonnegative integers N, $M : P \to N$, which means the assignment of tokens to the places of a Petri net. The time semantics of TPN applied here follows *holding durations* [16]. When a transition fires, the action of removing and created tokens is done instantaneously. However, the created tokens are not available to enable new transitions until they have been in their output place for the time specified by the *time delay* of the transition. Time is associated with tokens. Each token has a *timestamp* which models the time when the token becomes available for consumption.

Definition 2 (*Enabling Rule*). A transition t in a TPN is said to be enabled at time τ iff:

(1) Each input place p of t contains at least the number of tokens which is equal to the weight of the directed arc connecting p to t, i.e., $\forall p \in P : M(p) \geq I(t, p)$;
(2) τ is not smaller than any *timestamp* that is held by the token contained in the input places of t. (This ensures that all tokens contained in the input places of t are available.);
(3) τ is the smallest element of N that satisfies (1) and (2).

Definition 3 (*Firing Rule*). The firing rule of an enabled transition t is defined below:

(1) Transitions are *eager*, i.e., t must fire if it can be fired;
(2) The firing of t removes tokens from each input place p with the number equals the weight of the directed arc from p to t. It also deposits tokens to each output places p with the number equal to the weight of the directed arc from t to p, i.e., $M'(p) = M(p) - I(t, p) + O(t, p)$ for any p in P;
(3) The timestamp of each generated token equals the *firing time* plus the *delay* of t.