Fast reactive scheduling to minimize tardiness penalty and energy cost under power consumption uncertainties

Cao Vinh Le, Chee Khiang Pang *

Department of Electrical and Computer Engineering, National University of Singapore, Singapore

A R T I C L E   I N F O

Article history:
Received 11 March 2013
Received in revised form 12 June 2013
Accepted 9 July 2013
Available online 23 July 2013

Keywords:
Discrete event control
Dynamic scheduling
Energy-efficient manufacturing
Power consumption uncertainties
Reactive scheduling

A B S T R A C T

Motivated by the need to deal with uncertainties in energy optimization of flexible manufacturing systems, this paper considers a dynamic scheduling problem which minimizes the sum of energy cost and tardiness penalty under power consumption uncertainties. An integrated control and scheduling framework is proposed including two modules, namely, an augmented discrete event control (ADEC) and a max-throughput-min-energy reactive scheduling model (MTME). The ADEC is in charge of inhibiting jobs which may lead to deadlocks, and sequencing active jobs and resources. The MTME ensures the fulfillment of the innate constraints and decides the local optimal schedule of active jobs and resources. Our proposed framework is applied to an industrial stamping system with power consumption uncertainties formulated using three different probability distributions. The obtained schedules are compared with three dispatching rules and two rescheduling approaches. Our experiment results verify that MTME outperforms three dispatching rules in terms of deviation from Pareto optimality and reduces interrupted time significantly as compared to rescheduling approaches. In addition, ADEC and MTME are programmed using the same matrix language, providing easy implementation for industrial practitioners.

1. Introduction

Flexible manufacturing systems (FMSs) are modern production facilities which possess high flexibility of resource allocation and part routing. A resource is capable of performing multiple jobs, and multiple resources can be used to perform the same job on a part (Abazari, Solimanpur, & Sattari, 2012; Chan, Bhagwat, & Wadhwa, 2008, 2012; Pang, Lewis, Lee, & Dong, 2011). If one monitors the energy consumption of FMS, it is not uncommon to see that different resources require different productive powers and processing times to perform the same job. This variation is due to a multitude of factors, whether predicted or unpredicted, including the resource type, its operating conditions, process parameters, and part type (Abdelaziz, Saidur, & Mekhilef, 2011). To reduce energy cost of FMS, it is crucial to develop effective scheduling algorithms which generate energy-efficient schedules complying with production constraints.

Owing to current looming economic situation and rising energy prices, the reduction of energy cost in FMS has been recently addressed with great efforts in both academia and industry (Du, Chen, Huang, & Yang, 2011; Fang & Lin, 2013). Current research literature on energy-efficient scheduling often deals with the static environments, where power consumption of resources is fixed and no uncertainties would influence job processing after a schedule is executed. Real manufacturing is, however, dynamic and subjected to a wide range of uncertainties. Uncertainties in manufacturing have been classified into two categories, namely, resource-related uncertainties such as machine breakdown, machine degradation, tool wears, and job-related uncertainties such as rush jobs, job cancellation, stochastic processing times (Vieira, Hermann, & Lin, 2003). As such, scheduling under uncertainties, also known as dynamic scheduling, has attracted much attention in recent years (He & Sun, 2012; Horng, Lin, & Yang, 2012; Xiong, Xing, & Chen, 2012). The FMS scheduling problem is non-deterministic polynomial-time hard (NP-hard) in computational complexity theory, but consideration of uncertainties further aggravates its complexity. The existent approaches for dynamic scheduling in FMS can be classified into three categories, namely, the reactive, the proactive, and the predictive–reactive. Each approach has its own pros and cons (Ouelhadj & Petrovic, 2009).

Predictive–reactive scheduling is a scheduling/rescheduling process, in which the baseline schedules are generated offline and the active schedules are revised online in response to real-time uncertainties. The most common predictive-reactive scheduling include completed rescheduling (CR) and partial rescheduling (PR) (Choi & Wang, 2012; Vieira et al., 2003). In theory, CR provides
the optimal schedules, but these schedules are rarely achievable in practice and require prohibitive computational time. In addition, it can result in instability and disruption in manufacturing flows, leading to tremendous production costs. In PR, only jobs and resources affected by the uncertainties are rescheduled. On the other hand, the reactive scheduling is characterized by its capability of real-time decision-making, in which no baseline schedules are generated offline, and decisions are quickly made online using real-time information. Dispatching rules are typical examples of reactive scheduling, in which jobs are selected by sorting them according to some predefined criteria. Dispatching rules are still the most preferred scheduling approaches in industry due to their ease of implementation, low computational cost, and guarantee of schedule stability and feasibility (Ko, Kim, & Baek, 2010; Mouelh-Chibani & Perreval, 2010; Chiang, 2013; Sule, 2007; Tay & Ho, 2008). The main weakness of reactive scheduling is that they cannot globally optimize the overall performance of generated schedules. Proactive scheduling focuses on building a predictive schedule which minimizes the effects of real-time uncertainties (Horng et al., 2012). Baseline schedules are generated offline and will not be revised online. The main difficulty of these approaches is modeling of uncertainties. Computational cost is also an issue, since the stochastic search space is usually huge.

In this paper, a FMS dynamic scheduling problem which minimizes the sum of energy cost and tardiness penalty is considered under power consumption uncertainties. Uncertainties in power consumption are realistic in a dynamic manufacturing environment, as power consumption was verified to be dependent on uncertain factors including machine conditions, tool conditions, and workloads (Abdelaziz et al., 2011). The minimization of energy cost and tardiness penalty is a practical problem which was considered by Fang and Lin (2013) under static environment. Such tradeoff happens when a resource requires shorter time but higher energy to perform a job as compared to others (Fang & Lin, 2013).

To solve the formulated dynamic scheduling problem, this paper proposes a matrix-based integrated control and scheduling framework for a class of FMSs with shared resources and flexible part routing. Such configuration can be encountered in many realistic manufacturing flowlines, job shops, and material handling systems. The proposed framework can be viewed as an aggregation of two interacting modules, an augmented discrete event control (ADEC) and a max-throughput-min-energy reactive scheduling model (MTME). The ADEC has been proposed recently (Le, Pang, Lewis, Gan, & Chan, 2011; Pang, Hudas, Mikulski, Le, & Lewis, submitted for publication), proving to be very efficient in modeling and controlling the large-scale discrete-event dynamics of typical manufacturing systems. In particular, it reduces the model complexity when modeling large-scale FMSs as compared to the traditional conjunctive supervisory tools, such as the discrete event control (DEC) (Bogdan, Lewis, Kovacic, & Mireles, 2006; Pang et al., 2011) and Petri Nets (PNs) (Huang, Shi, & Xu, 2012). The proposed MTME resembles a reactive scheduling approach, which dispatches the imminent jobs and resources quickly and online using real-time power consumption of resources. It includes two 0–1 linear programming submodels, the former maximizes the production throughput and the latter minimizes the energy cost at every dispatching epoch. Both ADEC and MTME are programmed using the same matrix language and function during operational control as a whole, which provide easy implementation for industrial practitioners.

Our proposed framework is tested on an industrial FMS at a stamping company in the Republic of Singapore. The stamping parts are various types of voice coil motor (VCM) yokes used in commercial hard disk drive (HDD) actuators. Power consumption of resources is continually monitored using Rudolf R-DPA96A digital power analyzers (RUDOLFS). RUDOLFS are interfaced with computers via LabVIEW® environment. The schedules obtained by our proposed framework are compared with three dispatching rules, CR, and PR approaches. The experiment results with different batch sizes verify that MTME outperforms the three dispatching rules for all test cases in terms of deviation from Pareto optimality. The PR outperforms MTME when the batch size is small (short schedules), but the reverse is observed when the batch size is larger than 60 parts (long schedules). In terms of mean interrupted time, MTME achieves less than 1 s for all test cases, while the PR and CR cause prohibitive interrupted time (instability) for the FMS.

The rest of the paper is organized as follows. Section 2 describes the FMS and introduces the ADEC model of FMS. Section 3 formulates the dynamic scheduling problem under power consumption uncertainties, while Section 4 provides the formulation of MTME based on the ADEC model. In Section 5, our framework is evaluated based on an industrial stamping system with experiment results and discussions. Finally, our conclusions and future work directions are summarized in Section 6.

2. Background

In an era of intensive competition, manufacturing systems have migrated from conventional fixed-hardware sequential or batch production with dedicated workstations to FMSs with shared resources and flexible part routing. In this section, FMSs are described. It is then proceeded to provide a general description of the ADEC model of FMS, introducing the most significant details and notations (Le et al., 2011; Pang et al., submitted for publication).

2.1. Description of FMS

The FMS class of systems, investigated herein, has the following properties (Bogdan et al., 2006): (a) each part type has a strictly defined sequence of jobs; (b) each job in the system requires one and only one resource; (c) there are choice jobs (jobs which can be performed by alternative resources) and shared resources (resources which can perform different jobs); (d) resource allocation and part routing are flexible; (e) there are no assembly jobs, and (f) jobs are not preemptive, i.e., once assigned, a resource cannot be removed from a job until it is completed.

A FMS consists of a set of resources, denoted by \( R = \{ r_1, r_2, \ldots, |R| \} \), to manufacture \(|I|\) types of parts, where \(|i|\) is a standard term to denote the cardinality of a set. Each resource can be a machine, a buffer, a robotic arm, an automated guided vehicle, and so on. In large-scale FMSs, \( r_j \) can denote a pool of similar parts (batch size) to be manufactured. Each \( r_j \) has a strictly predefined sequence of jobs \( \omega_q = \{ q^1, q^2, \ldots, q^{\omega_j} \} \), where \( q^i \) is the \( i \)th job in \( \omega_q \) and \( |\omega_q| \) is the length of \( \omega_q \). The set of jobs is denoted by \( V = \{ v^1, q = 1, 2, \ldots, |I| \} \). In FMSs with flexible part routing, choice jobs are ubiquitous. Therefore, \( V \) can be partitioned into two disjoint subsets, \( V = V_C \cup V_N \), where \( V_C \) and \( V_N \) denote the sets of choice and nonchoice jobs, respectively. Let \( R(V_C) \) be the set of resources which can perform \( v^j \). Obviously, \( |R(v^j)| > 1 \) if \( v^j \in V_C \) and \( |R(v^j)| = 1 \) if \( v^j \in V_N \). For each \( \pi_q \omega_q \) is associated with two fictitious jobs \( u^q \) and \( y^q \) called input buffer and output buffer jobs which represent the storage of raw and finished parts, respectively. \( u^q \) and \( y^q \) do not require any resources, thus \( R(u^q) = R(y^q) = \emptyset \).

2.2. ADEC Model of FMS

Let us consider a FMS with part type \( \pi_q \) characterized a job sequence \( \omega_q \) properly predefined and a set of available resources...
It is convenient to describe the production flow of \( p_\alpha \) using a finite set of linguistic IF–THEN rules denoted by \( X^q \), and \( \cup_{\eta=1}^{f_{\eta,q}} X^q = X \). Each rule \( x_i \in X^q \) has the form:

**IF** (job A is finished **AND** job B is finished **AND**...) **AND** (resource C is free **AND** resource D is free **AND**...) **AND** (resource M is free **OR** resource N is free **OR**...)** THEN** (start job E **AND** start job F **AND**...) **AND** (release resource C).

In the IF–part, the sets of preceding jobs, required resources, and part inputs needed to activate each rule are predefined. The THEN-part of each rule specifies the consequent jobs to be performed and the part outputs in the next dispatching epoch. As compared the conjunctive supervisory tools such as the DEC (Bogdan et al., 2006; Pang et al., 2011) and PNs (Huang et al., 2012), the ADEC contains additional add-on disjunctive rule bases (the OR operators) in the IF-part.

The ubiquity of choice jobs has a serious impact on the conjunctive supervisory tools. Specifically, the starting of a choice job which can be processed by \( p \) alternative resources must be described by \( p \) conjunctive IF–THEN rules (Bogdan et al., 2006). This exhibits an incompetency, which is called the rule explosion, to model a large-scale FMS which possesses a large number of choice jobs. This will be expatiated further in Section 2.2.3. In the ADEC, the rule explosion is overcome by proposing a novel add-on disjunctive reasoning into the rule base, where the IF–part now contains OR operators. As such, the starting of a choice job which is described by one and only one rule regardless of the number of disjunctive resources (Le et al., 2011; Pang et al., submitted for publication).

In the ADEC model, the system sets of rules \( X \), jobs \( V \), and resources \( R \) are represented in a compact form using Boolean matrices and vectors. The following Boolean vectors are defined: a job vector \( v \in V \), a resource vector \( r \in R \), and a rule vector \( x \in X \). For instance, if \( x \) is the set of jobs, resources, and rules, respectively, corresponding to their “1” elements. The set represented by \( a \) (for \( a \in \{ \mathbf{v}, \mathbf{r}, \mathbf{x} \} \)) is called the support of \( a \), denoted by \( \text{supp}(a) \); e.g., given \( v = [v_1, v_2, \ldots, v_q]^T \), \( v_1 = 1 \) if and only if \( v_1 \in \text{supp}(v) \).

### 2.2.1. ADEC Matrices and Vectors

Let us first focus on the ADEC matrices and vectors of a single part type \( p_\alpha \), and then obtain the global ADEC model of FMS with multiple part types. To map the set of preceding jobs to the set of rules, job sequence matrix \( F^1_\alpha \) is defined such that element \( f_{\eta,q}^\alpha = 1 \) if the completion of job \( v_\eta \) is required to activate rule \( x_q \). Analogously, job start matrix \( S^0_\alpha \) has element \( s_{\eta,q}^0 = 1 \) such that job \( v_\eta \) is started if rule \( x_q \) is activated. To map the set of conjunctive resources to the set of rules, conjunctive resource assignment matrix \( F^2_\alpha \) is defined such that element \( f_{\eta,q}^2 = 1 \) if the availability of resource \( r_j \) is required to activate rule \( x_q \). Finally, input buffer matrix \( F^3_\alpha \) is the input matrix which maps the set of input parts to the set of resources, having element \( f_{i,\eta}^3 = 1 \) if the presence of input \( v_{\eta,i} \) is required to activate rule \( x_i \). Output matrix \( S^1_\alpha \) has the \((i,j)\) element set to “1” if output \( v_{\eta,j} \) is released if rule \( x_i \) is activated.

The rule set of \( X^q \) is represented by vector \( x^q \) having element \( x_{i,q}^q \) stand for rule \( x_i \). If all antecedences (IF part) required for rule \( x_i \) are met, then \( x_{i,q}^q = 1 \) (true). \( v_i \) is the job completed vector having element \( v_{\eta,i}^q = 1 \) if job \( v_{\eta,i} \) is completed. \( r_j \) is the resource available vector having element \( r_{i,j}^q = 1 \) if resource \( r_j \) is available. \( u_i \) is the input vector having element \( u_{i,j}^q = 1 \) if part input \( v_{\eta,j} \) occurs. Entries of “1” in vectors \( v_i \) denote the starting jobs and in vectors \( u_j \) imply that finished parts are out.

Deadlock is avoided in ADEC using the deadlock resolution matrix \( F^5_\alpha \) and vector \( u_i^q \) (Bogdan et al., 2006). Matrix \( F^5_\alpha \) has as many columns as the number of jobs performed by shared resources, i.e., the number of columns of \( F^3_\alpha \) having multiple “1s.” Element \( f_{\eta,q}^5 = 1 \) if job \( v_\eta \) is a preceding job needed to activate rule \( x_q \). Then, element \( u_{i,j}^q = 1 \) determines the inhibition of logic state \( x_j \) (whether rule \( x_j \) can be activated). Depending on the way one selects the conflict resolution strategy to generate vector \( u_i^q \), deadlock can be avoided. On the other hand, possible assignments of available disjunctive resources to choice jobs are captured using the conjunctive resource assignment matrix \( F_{\alpha,\eta} \), which has entry \( f_{\eta,q}^\alpha = 1 \) if resource \( r_j \) can accomplish rule \( x_q \). \( F_{\alpha,\eta} \) essentially captures information about which available resources can be used for each rule, such that only one of the possible resources listed in row \( i \) of \( F_{\alpha,\eta} \) is required to activate rule \( x_q \). As such, \( F_{\alpha,\eta} \) maps the set of resources to the set of rules.

For example, let us consider part type \( p_\alpha \) of a flexible stamping system. \( p_\alpha \) has a job sequence \( v_{\eta,1} = v_{\eta,1}^1 \cdots v_{\eta,4} \), where \( v_{\eta,1}^1 \), \( v_{\eta,2}^1 \), and \( v_{\eta,4}^1 \) are stamping jobs, and \( v_{\eta,1}^2 \), \( v_{\eta,2}^2 \), and \( v_{\eta,3}^2 \) are routing jobs. \( v_{\eta,1}^3 \), \( v_{\eta,2}^3 \), and \( v_{\eta,4}^3 \) are choice jobs and \( v_{\eta,1}^3 \) is a nonchoice job. \( u^1 \) and \( y^1 \) are input buffer and output buffer, respectively. The resource set of this flexible stamping system includes eight stamping machines (M1–M8) and five material routing resources (B1–B4). The jobs and possible resource assignments of \( p_\alpha \) are reported in Table 1. The rule base which describes the processing of \( p_\alpha \) is reported in Table 2. The matrix representation of the reported rule base can be easily obtained. Namely, the job sequence matrix and the input matrix are

\[
F^1_\alpha = \begin{bmatrix}
v_{\eta,1}^1 & v_{\eta,2}^1 & v_{\eta,3}^1 & v_{\eta,4}^1 & v_{\eta,1}^2 & v_{\eta,2}^2 & v_{\eta,3}^2 \\
v_{\eta,1}^3 & v_{\eta,2}^3 & v_{\eta,3}^3 & v_{\eta,4}^3 \\
\end{bmatrix},
\]

\[
F^3_\alpha = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix},
\]

(A)
Analogously, the conjunctive and disjunctive resource requirement matrices are
\[ M_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_9 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_{10} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[ M_{12} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

and similarly matrices \( S_1, S_2, \ldots, S_6 \) can be derived. \( \mathbf{0} \) denotes a null matrix. The global vectors \( \mathbf{x}, \mathbf{v}, \mathbf{r}, \mathbf{u}, \mathbf{v}_c, \mathbf{r}_c, \mathbf{u}_c, \mathbf{v}_r, \mathbf{r}_r, \mathbf{u}_r \), etc., of the FMS are given by
\[ \mathbf{x} = \begin{bmatrix} x_1 \n x_2 \n x_3 \n x_4 \n x_5 \n x_6 \n x_7 \n x_8 \n x_9 \n x_{10} \n x_{11} \n x_{12} \end{bmatrix} \]
\[ \mathbf{v}_c = \begin{bmatrix} v_{c1} \n v_{c2} \n v_{c3} \n v_{c4} \n v_{c5} \n v_{c6} \n v_{c7} \n v_{c8} \n v_{c9} \n v_{c10} \n v_{c11} \n v_{c12} \end{bmatrix} \]
\[ \mathbf{r}_c = \begin{bmatrix} r_{c1} \n r_{c2} \n r_{c3} \n r_{c4} \n r_{c5} \n r_{c6} \n r_{c7} \n r_{c8} \n r_{c9} \n r_{c10} \n r_{c11} \n r_{c12} \end{bmatrix} \]
\[ \mathbf{u}_c = \begin{bmatrix} u_{c1} \n u_{c2} \n u_{c3} \n u_{c4} \n u_{c5} \n u_{c6} \n u_{c7} \n u_{c8} \n u_{c9} \n u_{c10} \n u_{c11} \n u_{c12} \end{bmatrix} \]

and similarly vectors \( \mathbf{u}, \mathbf{u}_d, \mathbf{v}, \mathbf{y} \) can be derived. It is worth noting that the job sequences of different part types are independent, each using its own jobs, so that \( \mathbf{F}_d \) is block diagonal. However, all the job sequences use the same pool of resources available in the FMS, and so have commensurate columns of their resource assignment matrices \( \mathbf{F}_d \) and \( \mathbf{F}_{id} \).

2.2.2. ADEC logical state equation

At each dispatching epoch, the ADEC receives the vectors \( \mathbf{v}, \mathbf{r}, \mathbf{u}, \mathbf{v}_c, \mathbf{r}_c, \mathbf{u}_c \). The ADEC’s main function at a supervisory level is to determine which rules can be activated, which jobs to be started, and the part outputs for which a release command to be sent to the FMS. These functions are processed by means of two different sets of logical equations, the former is used for checking the conditions for the activation of rules, and the latter is used for defining the consequent controller outputs. The updated value of the logical rule vector is computed with the following logical state equation
\[ x(k+1) = F_p \otimes \mathbf{x}_d(k) \oplus F_r \otimes \mathbf{r}_d(k) \oplus F_u \otimes \mathbf{u}_d(k) \]
\[ \oplus F_y \otimes \mathbf{y}_d(k) \]
where \( k \) denotes the dispatching epoch. The overbar in (6) denotes a vector negation. Given a natural number vector \( \mathbf{a} \), its negation is such that \( \overline{a} = 0 \) if \( a_i > 0 \), and \( \overline{a} = 1 \) otherwise. \( \otimes \) and \( \oplus \) denote the and/or multiplication and addition, respectively. \( C = A \oplus B \) is defined by \( c_q = (a_q \land b_q) \lor (a_q \land b_q) \lor \ldots \) and \( C = A \land B \) is defined by \( c_q = (a_q \lor b_q) \land (a_q \lor b_q) \land \ldots \) and \( \lor \) and \( \land \) are standard symbols for logical AND and OR operations, respectively. \( x(k+1) \) essentially provides the information of which rules can be activated without causing a deadlock in dispatching epoch \( k+1 \).

On the ground of the current value of \( x(k+1) \), the scheduling decisions are included by the MTME resulting in the final state vector \( x(k+1) \), which describes the set of eventually activated rules at dispatching epoch \( k+1 \). The computation of \( x(k+1) \) will be presented later in Section 3. Based on the value of \( x(k+1) \), the ADEC decides which jobs to be started and which outputs to be released by means of the following output equations
\[ v_i(k+1) = S_v \otimes x_p(k+1), \]
\[ y(k+1) = S_y \otimes x_p(k+1). \]

(6) and (7), (8) represent a conjunctive and disjunctive rule-based supervisory control for any class of discrete event systems. It is worth noting that all matrices and vectors are Boolean, making real-time computations easy even for large-scale FMSs.

2.2.3. Model complexity reduction for modeling large-scale FMS

The key benefit provided by using the ADEC is reduction of the model complexity for implementing large-scale FMSs. If one compares our framework with the conjunctive DEC, then usage of our framework requires less memory. It should be noted that one dimension of all DEC (ADEC) matrices is \( |X| \) (e.g., \( |X| \times |X| \) for \( F_r \), \( |X| \times |V| \) for \( F_v \), \( |X| \times |R| \) for \( F_r, F_v, \ldots \)), where \( |X|, |V|, \text{ and } |R| \) are the numbers of rules, jobs, and resources, respectively. Recall that the conjunctive DEC needs \( p \) rules to describe the starting of a choice job, which can be performed by \( p \) different resources. This drastically increases \( |X| \). In our framework, a new matrix \( F_{id} \) (dimensions of \( |X| \times |R| \) is included to keep \( |X| \) minimized. Since \( |X| \gg |R| \) in large-scale FMSs, the reduction in the model complexity can be significant.

It is also worth noting that the ADEC described in (6) is more general than a PN. In fact, the first two and last two terms are equivalent to a PN, while the middle term allows additional OR reasoning in the rule base. To further exemplify this, let consider the FMS recently presented in (Huang et al., 2012). Although, this system was considered as a place-timed PN, the timing is ignored in this paper.

In part type \( p_2 \) of this FMS, there are three machining jobs \( P_{22}, P_{24}, \text{ and } P_{26} \), two buffering jobs \( P_{23} \text{ and } P_{25} \), three resources \( R_1, R_2, \text{ and } R_3 \), an input \( P_1 \), and an output \( P_2 \). This part type is therefore characterized by a job sequence \( o_2 = \{P_{22}, P_{23}, P_{24}, \ldots, P_{26}\} \) and a set of resources \( R = \{R_1, R_2, R_3\} \). Job \( P_{22} \) is not a choice job, while job \( P_{24} \text{ and } P_{26} \) are. Choice job \( P_{24} \) can be done by either resource \( R_2 \text{ or } R_3 \), while choice job \( P_{26} \) can be done by either resource \( R_1 \text{ or } R_3 \). The PN model of this part type is presented in Fig. 1. It can be seen that the PN, which only contains...
AND reasoning, requires \( p \) branches (each branch contains two transitions and one place noted by dashed circles) to represent a choice job that is processed by \( p \) disjunctive resources.

For example, consider choice job \( P_{24} \). The branch containing place \( P_{24} \) presents the case where this choice job is processed by resource \( R2 \), while the token will flow to the branch containing place \( P_{23} \) if resource \( R3 \) is assigned instead. To switch between these branches (resource routing), controlled places are added to the PN accordingly. Next, a PN-equivalent ADEC model of this part type is presented in Fig. 2. It can be seen that the ADEC only needs one branch to represent a choice job regardless of the number of processable resources. The resource routing is decided by switching corresponding resources (noted by dashed circles) not branches. This significantly reduces the model complexity for modeling large-scale FMSs.

### 3. Dynamic scheduling under power consumption uncertainties

In this section, a mathematical model of power consumption uncertainties is presented and the dynamic scheduling problem is formally defined. We use a \( |\mathcal{R}| \times |\mathcal{A}_q| \) matrix \( \mathbf{D}^j \) to denote the job processing times of part \( \pi_q \), where element \( d_{ij}^q \) is the processing time of \( r_j \) to perform \( q_j^i \). \( d_{ij}^q = 0 \) always if \( r_j \) cannot perform \( q_j^i \). Analogously, the productive power of resources to manufacture \( \pi_q \) are denoted by a \( |\mathcal{R}| \times |\mathcal{A}_q| \) matrix \( \mathbf{A}^q(t) \), where element \( a_{ij}^q(t) \) is the productive power of \( r_j \) to perform \( q_j^i \) at operation time \( t \). \( a_{ij}^q(t) = 0 \) always if \( r_j \) cannot perform \( q_j^i \).

#### 3.1. Mathematical model of power consumption uncertainties

A practical mathematical model of power consumption uncertainties is selected purely for performance evaluation of MTME and the related works presented later in Section 5. In practice, power consumption uncertainties due to machine degradation may follow different kinds of mathematical models.

Most importantly, industrial practitioners need not to model power consumption uncertainties prior to using our proposed framework. As a reactive scheduling approach, MTME does not consider the uncertainties in generating schedules, but finding the effective ways to react to uncertainties at every dispatching epoch. With this reactive capability, our proposed framework can be adapted to any mathematical model of power consumption uncertainty. The reactive nature of our proposed framework also differs with predictive-reactive approaches such as CR and PR in the sense that our framework is triggered by job completions instead of changes in energy consumption.

The power consumption \( a_{ij}^q(t) \) is modeled by a step function, which is often defined in literature as \((\text{Bachman, Lawrence, \\& Edward, 2000})\)

\[
a_{ij}^q(t) = \sum_{k=0}^{n} \lambda_k \delta_{ik}(t),
\]

where \( n > 0 \) is the number of times that \( a_{ij}^q(t) \) increases, \( \lambda_k \) are real numbers, \( B_k \) are intervals, and \( \delta_{ik} \) is the indicator function of \( B \) defined by

\[
\delta_{ik}(t) = \begin{cases} 
1 & \text{if } t \in B_k \\
0 & \text{if } t \notin B_k
\end{cases}
\]

In this definition, the intervals \( B_k \) have the following two properties, namely, \( B_k \cap B_l = \emptyset \) for \( k \neq l \) and \( t_{z-1}^q B_k = [0, \infty) \). To incorporate the uncertainties into \( a_{ij}^q(t) \), \( B_k \) is defined by

\[
B_k = \begin{cases}
(0, b_0) & \text{if } k = 0, \\
\left[ \sum_{i=1}^{k-1} b_i / \sum_{i=1}^{k} b_i \right] & \text{if } k \neq \{0, n\}, \\
t_{n+1}^q / \sum_{i=1}^{k} b_i & \text{if } k = n,
\end{cases}
\]

where the values of \( \{b_z, 0 \leq z \leq n-1\} \) follow a probability distribution denoted by \( g \left( b; \mu^q_{ij}, \sigma^q_{ij} \right) \) with \( \mu^q_{ij} \) and \( \sigma^q_{ij} \) are the distribution mean and variance, respectively. Three probability distributions widely used in reliability engineering and life data analysis are investigated, namely, the Weibull, exponential, and truncated normal distributions (Horng et al., 2012). Obviously, \( a_{ij}^q(t) \) must be monotonically increasing, as such \( a_{ij}^q(0) = \lambda_0 \) is assumed for simplicity, i.e., \( a_{ij}^q \) constantly grows by 5% every time it increases. Lastly, it is worth noting \( a_{ij}^q(0) = \lambda_0 \) always.

#### 3.2. Problem description

The FMS scheduling problem that minimizes the sum of energy cost and tardiness penalty under power consumption uncertainties can be formulated as

\[
\begin{align*}
\min_{s \in \Theta} \quad & J(s) = \sum_{r_j \in \mathcal{R}} \sum_{q_j \in \mathcal{A}_q} \sum_{i=1}^{n} \sum_{m=1}^{q_j} \gamma_{ijm}^q a_{ijm}^q(t) dt_{ijm}^q + \sum_{q_j \in \mathcal{Q}} \gamma_{ij}^q \tau_{ij},
\end{align*}
\]

where \( \tau_{ij} \) and \( \gamma_{ij}^q \) denote tardiness and penalty of unit-time tardiness of \( \pi_q \), respectively. \( J(s) \) denotes the weighted sum of energy consumption and makespan to be minimized. \( \gamma_{ijm}^q \) are decision variables such that \( \gamma_{ijm}^q = 1 \) if job \( q_j^i \) on part \( m \) \( \{m = 1, 2, \ldots, q(\pi_q)\} \) is assigned to resource \( r_j \). \( \gamma_{ij}^q \) are defined variables, otherwise. Let \( \Theta \) denote the set of feasible schedules. A feasible schedule \( s \) should satisfy the following production constraints:

\[
\begin{align*}
\sum_{r_j \in \mathcal{R}} y_{ijm}^q &= 1, \\
\gamma_{ijm}^q &\geq t_{ijm}^q + \sum_{r_j \in \mathcal{R}} y_{ijm}^q \gamma_{ijm}^q, \\
y_{ijm}^q + y_{ijm}^q &\geq 1 + \left( \gamma_{ijm}^q + \gamma_{ijm}^q \right), \\
t_{ijm}^q &\geq t_{ijm}^q = \gamma_{ijm}^q \left( a_{ijm}^q d_{ijm}^q - M \right), \\
t_{ijm}^q &\geq \gamma_{ijm}^q \left( a_{ijm}^q d_{ijm}^q - M \right), \\
\tau_{ij} &= \gamma_{ijm}^q \left( a_{ijm}^q d_{ijm}^q - M \right), \\
\forall \pi_q \in \Pi, \forall q_j \in \mathcal{Q}, &\gamma_{ijm}^q \in \mathcal{Q}, \quad m = 1, 2, \ldots, q(\pi_q), \quad y_{ijm}^q \in \{0, 1\},
\end{align*}
\]

### Fig. 1. PN model of example part type.

### Fig. 2. PN-equivalent ADEC model of example part type.
where $t_{in}^m$ denotes the starting time of $t^m$ on part $m$, $\delta_{in}^m$ is a dummy variable, $M$ is a large number for big $M$ method, and $D^p$ denotes the due date of part type $p$. (13) ensures that each job needs only one machine at a time. (14) specifies the precedence constraints due to the order in which the jobs need to be done for each part. (15)-(17) guarantee that each resource can process at most one job at a time and jobs cannot be preempted once started. (15) functions as an indicator such that if $y_{im}^{p}=y_{im}^{p+1}=1$ then (17) and (18) will work in such a way that only one of them will hold. Finally, (18) defines the tardiness of each part type. It is worth noting that though the energy cost and tardiness penalty are weighted equally in this paper, decision makers can adjust the weights based on the specific economic situations.

4. Fast reactive scheduling in FMS

At a given dispatching epoch, the ADEC determines in (6) which rules can be activated without causing a deadlock. When multiple uninhibited rules are ready to be activated and multiple disjunctive resources are available to be assigned, the MTME must be adopted as tie-break rule to select the most effective schedules to execute, such that the predefined performance criteria are optimized. Based on the ADEC model of FMS, this section develops a fast reactive scheduling model that optimizes the throughput and energy cost of the FMS at every dispatching epoch. Throughput is adopted here as an objective function as throughput maximization was verified to effectively reduce the tardiness penalty of FMS (Prakash, Khilwani, Tiwari, & Cohen, 2008). For generality, it should be noted that any kind of reactive scheduling approaches, with any kind of optimization criteria and constraints, can be easily combined with the ADEC in such a way to be described as follows.

4.1. Solution overview

From a global viewpoint, it is convenient to view the framework architecture as an aggregation of two interacting modules, the ADEC and the MTME, both are programmed using the same matrix language. The framework provides a complete description of the discrete event dynamics of a FMS; and is used (1) as a means to track active job/resource statuses and sequence deadlock-free imminent jobs and outputs, and (2) to identify the optimal schedule of jobs and resources at each dispatching epoch. As shown in Fig. 3, its inputs are the acknowledgement messages from the FMS sensors for resource availability and job completion (vectors $v_s$ and $r_y$), the information about the arrival of new part inputs (vector $u_r$), and the deadlock avoidance control (vector $u_d$) computed by a deadlock avoidance policy (DAP). Using these information, the logic conditions of control rules (vector $x$) is computed by the ADEC. The interaction of the ADEC and the MTME includes two phases. First, the ADEC computes and passes the deadlock-free search space (choice sets) to the MTME (resource set $R_a$ and rule set $X_a$). Second, the MTME identifies the optimal assignment of rules and resources (matrices $F_r$ and $F_ad$) that optimizes (12) without violating the required production constraints. It is worth noting that the inputs to the MTME are also real-time power consumption and processing time of resources (matrices $X(t)$ and $D$) obtained by a real-time energy monitoring network. The framework’s outputs are vectors describing the conditions of the jobs to be start (vector $v_s$) and the part output to be released (vector $y$). All the mentioned tasks of the two modules are performed by means of matrix equations. The FMS sensors returns acknowledgements for job completion, for the subsequent release of the resource, and about the arrival of new part inputs. Interested practitioners are referred to (Bogdan et al., 2006) for the detailed hardware and software implementations of ADEC.

4.2. Choice set

Prior to formulating the MTME, it is needed to identify its search space or choice set. It can be seen that (6) determines the set of deadlock-free rules, denoted by $X_a(k+1)$, which can be activated at dispatching epoch $k+1$, where $X_a = \text{supp}(x)$. $X_a$ can always be partitioned into two disjoint subsets $X_a = X_a^c \cup X_a^{nc}$, where $X_a^c$ denotes the set of choice rules, i.e., rules which start choice jobs, and $X_a^{nc}$ denotes the set of nonchoice rules.

In the resource domain, denote by $R(k+1)$ a set of resources which accomplish the rule set $X_a(k+1)$. A resource vector that represents $R(k+1)$ is calculated by
\[ r_j^d (k+1) = x^j (k+1) \odot (F_r \odot F_a), \]

(20)

where \( R_r = \text{supp}(r_j) \). In addition, denote the set of available resources by \( R_a(k+1) \), where \( R_t = \text{supp}(r_t) \). Let \( R_r = R_r \cap R_t \). It can be clearly seen that the search space of MTME, which includes all possible schedules at dispatching epoch \( k+1 \), is defined by a resource set \( R_a(k+1) \) and a rule set \( X_r(k+1) \).

4.3. MTME

To compress the scheduling decisions into convenient matrix forms, at dispatching epoch \( k \) define \( F_a \)'s submatrix \( F_{ud}(k) \); such that in the case of multiple entries of "1" (choice jobs) in a row of \( F_{ra} \), submatrix \( F_{ud}(k) \) comprises at most one "1" referring to the resource selected to process the corresponding choice job, and in the case of multiple entries of "1" (shared resources) in a column of \( F_{ra} \), submatrix \( F_{ud}(k) \) comprises at most one "1" referring to the rule selected to be fired. Analogously, define \( F_r \)'s submatrix \( F_{rd}(k) \); such that in the case of multiple entries of "1" in a column of \( F_{ra} \), submatrix \( F_{rd}(k) \) comprises at most one "1". As such, \( F_{ud}(k) \) and \( F_{rd}(k) \) are assembled in the ADEC logical state equation by

\[
X_r(k+1) = F_r \odot \tau_r(k) \odot (F_r \odot F_{ud}(k)) \odot r_t(k) \odot F_a \odot u(k)
\]

\[
+ (F_{rd} \odot F_{rd}(k)) \odot r_r(k) \odot F_{td} \odot u(k),
\]

(21)

where \( \odot \) denotes the Hadamard product (piecewise multiplication) with \( C = A \odot B \) is defined by \( c_{ij} = a_{ij} \times b_{ij} \), \( X_r(k) \), \( X_a \) are sets of rules that include the scheduling decisions and will be eventually activated at dispatching epoch \( k+1 \).

The MTME at dispatching epoch \( k \) is presented as follows, where index \( k \) is dropped in all mathematical notations for brevity, i.e., \( f_{qj}^d(k) \) is simplified to \( f_{qj}^d \), where \( f_{qj}^d \) is the element of \( F_{ud}(k) \). The MTME comprises of two 0–1 linear programming submodels. The former computes the maximum throughput achievable; and the latter, among solutions of the former, decides the one with least energy cost. The first submodel is given by

\[
\text{maximize } \delta = \sum_{q,j} \sum_{r_i \in X_r} \sum_{r_j \in X_r} f_{qj}^d + f_{qj}^u,
\]

(22)

subject to

\[
\sum_{r_j \in X_r} f_{qj}^d \leq 1, \forall r_i \in X_r,
\]

(23)

\[
\sum_{j} f_{qj}^d + \sum_{j} f_{qj}^u \leq 1, \forall r_j \in R_r,
\]

(24)

\[
f_{qj}^d, f_{qj}^u \in \{0,1\}, \forall \langle q, r_j \rangle \in X_a \times R_r,
\]

(25)

where (22) is the cost function of throughput to be maximized, (23) essentially constrains the solution to select one and only one resource for each rule of \( X_a \), while (24) avoids shared-resource conflicts (if any). (25) is a mapping constraint which implies how resources and rules are indexed. With a solution of the first submodel, it is now proceeded to express the second submodel by

\[
\text{minimize } \sum_{q,j} \sum_{r_i \in X_r} \sum_{r_j \in X_r} q_j^o d_j^r (f_{qj}^d + f_{qj}^u),
\]

(26)

subject to

\[
\sum_{q,j} \sum_{r_i \in X_r} \sum_{r_j \in X_r} (f_{qj}^d + f_{qj}^u) = c \delta_{\text{max}},
\]

(27)

where (26) is the cost function of energy cost to be minimized. (27) depicts the constraint of the minimum throughput must be achieved, where \( c \in R_{c ECC} \) is a weight parameter. As the energy cost and tardiness penalty are weighted equally in (12), \( c = 0.5 \) is chosen herein. \( c \) can be adjusted depending on how the dynamic scheduling problem is formulated. \( q_j^o (t) \) are real-time productive power measurements from the meters. Finally, \( f_{qj}^{\text{up}}(k) \) and \( f_{qj}^{\text{eu}}(k) \) denote the elements of \( F_{ud}(k) \) and \( F_{rd}(k) \), respectively.

The MTME is formulated as a standard 0–1 (binary) linear programming, which is classified as NP-hard in computational complexity theory. Advanced algorithms for solving 0–1 linear programming include cutting-plane method, branch and bound (B&B), branch and cut, branch and price, etc., each method has its own pros and cons. In this paper, a specialized B&B algorithm known as Balas additive algorithm is chosen as solution method, which is widely available in commercial solvers (Christou, 2012).

In general, B&B algorithms have exponential worst-case complexity on the problem size, but the average-case complexity is significantly lower. Analysis of average-case complexity of B&B algorithms often requires assumptions to be made, and the obtained results also vary on a case by case basis. The readers are referred to (Papadimitriou & Steiglitz, 1998) and the references therein for details on average-case complexity of B&B algorithms.

At every dispatching epoch, the MTME generates online schedules in a local Pareto optimal way, and the global Pareto front can be calculated in certain scenarios. Unlike deterministic multi-objective optimization problems, whose Pareto optimal solutions are commonly generated using evolutionary algorithm (EA) or genetic algorithm (GA), the Pareto optimality in our problem is difficult to be achieved. The unknown statistics of power consumption uncertainties impedes us to define a robustness measure for robust Pareto optimality. In addition, since our proposed framework functions as an online scheduler, it is not possible to use advanced algorithms such as EA and GA, which require long computation times and induce disruptions for the production flow of FSMs.

5. Industrial case study

An industrial case study is carried out to verify the usability of our proposed integrated control and reactive scheduling framework. An application related to flexible stamping system is selected for the experiment. The energy data are monitored at a stamping company in the Republic of Singapore. This stamping system can be characterized by the type of FSMs described in this paper. Each stamping part type has a predetermined sequence of jobs, with some jobs can be processed by more than one resource, and some resources can perform more than one job. At this stamping system, the scheduling task is primarily decided based on human decisions.

5.1. Energy analysis of stamping process

Stamping includes a variety of sheet-metal forming processes, e.g., punching, coining, blanking, piercing, and bending, etc. During operation, the stamping die is placed into a reciprocating stamping press. As the press moves up, the top die moves with it, which allows the material to feed. When the press moves down, the die closes and performs the stamping operation. With each stroke of the press, a completed part is removed from the die (Kalpakjian & Schmid, 2006).

In this application, input parts are raw metal sheets, while output parts are various types of VCM yokes used in commercial HDD actuators. A typical VCM comprises of a coil rotatable about a predetermined axis; a pair of yokes opposing each other with a predetermined distance; and a permanent magnet between the pair of actuators. A typical VCM comprises of a coil rotatable about a predetermined axis; a pair of yokes opposing each other with a predetermined distance; and a permanent magnet between the pair of actuators.
This stamping system comprises of eight stamping machines and five material handling resources, which are only named by (M1–M8) and (B1–B5), respectively. Energy consumption is continually monitored using RUDOLFs. To interface RUDOLFs with computers or handheld devices, a graphic user interface (GUI) has been developed in LabVIEW®.

The stamping machines are of different working conditions as well as energy consumption profiles, and their performances and efficiencies are summarized in Table 3. It can be seen that there is a wide range in average stamping power, even for different machines of the same model. This is due to a multitude of factors, e.g., tooling, machine loading, machine degradation, machine age, etc. The entire stamping cycle can be divided into three main states, namely, productive, idle, and off as shown Fig. 5. In idle and off states, the power data are observed to be relatively constant. In productive state, many spikes are generated, and each spike is observed every time the stamping press moves down to perform stamping operations.

Using the measured power data, the productive power matrix $A(t)$ and the processing time matrix $D$ can be constructed as follows. Let $\text{Power}_{m}$, where $m$ is the number of samples, be the power data shown in Fig. 5, which is measured on resource $r_i$ when performing job $v_i^j$. As such, one has

$$a_{ij}^p = \frac{\sum_{m=0}^{i} \text{Power}_{m}}{t_j - t_i},$$

$$d_{ij}^p = \frac{t_j - t_i}{f_i},$$

where $t_j$ and $t_i$ denote the instances that the state is changed from idle to productive and productive to idle, respectively. $f_i$ is the sampling frequency of RUDOLFs.

5.2. ADEC model of stamping system

A part of this stamping system is now modeled using the ADEC. There are totally two part types $\pi_1$ and $\pi_2$, where part type $\pi_1$ was already described in Section 2.2.1 as an example. Part type $\pi_2$ has a job sequence $\omega_2 = \nu_2^M \nu_2^F \nu_2^B \cdot \nu_2^B$, where $\nu_2^M$, $\nu_2^F$, and $\nu_2^B$ are stamping jobs, and $\nu_2^M$ and $\nu_2^B$ are routing jobs. $\nu_2^F$ and $\nu_2^B$ are choice jobs and $\nu_2^M$ is a nonchoice job.

For each choice job, there is an associated routing resource that routes parts. For instance, choice job $\nu_2^F$ can be processed by M2 or M3, and is associated with routing resource B1; choice job $\nu_2^M$ can be processed by M3, M4, or M5, and uses routing resource B2 for part routing; and choice job $\nu_2^B$ can be processed by M7 or M8, and is routed by routing resource B5, etc. All routing resources are nonshared. All stamping machines are shared resources, except for M6–M8. Capacities of all resources are one.

Since energy data of material routing resources are not available in our application, they are assumed to be identical. This assumption does not affect the scheduling decisions, as all routing resources are nonshared without routing choices. From (28) and (29), the initial productive power matrices $A(0)(\text{kW})$ and the processing time matrices $D(t)$ of the stamping system are obtained as

![Fig. 4. An example of VCM yokes.](Image)

![Fig. 5. Typical power profile of stamping process.](Image)

### Table 3
Machine performance and efficiency.

<table>
<thead>
<tr>
<th>Machine ID</th>
<th>Rated tonnage (tonnes)</th>
<th>Motor rated power (tonnes)</th>
<th>Actual max load (tonnes)</th>
<th>Average stamping power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>200</td>
<td>22</td>
<td>168</td>
<td>11.96</td>
</tr>
<tr>
<td>M2</td>
<td>300</td>
<td>37</td>
<td>238</td>
<td>4.45</td>
</tr>
<tr>
<td>M3</td>
<td>300</td>
<td>37</td>
<td>250</td>
<td>7.60</td>
</tr>
<tr>
<td>M4</td>
<td>300</td>
<td>37</td>
<td>183</td>
<td>6.19</td>
</tr>
<tr>
<td>M5</td>
<td>300</td>
<td>37</td>
<td>176</td>
<td>5.37</td>
</tr>
<tr>
<td>M6</td>
<td>300</td>
<td>37</td>
<td>198</td>
<td>6.46</td>
</tr>
<tr>
<td>M7</td>
<td>300</td>
<td>37</td>
<td>202</td>
<td>7.84</td>
</tr>
<tr>
<td>M8</td>
<td>300</td>
<td>37</td>
<td>–</td>
<td>12.23</td>
</tr>
</tbody>
</table>

### Table 4
Part type $\pi_2$ jobs and possible resource assignments.

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu_1^M$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\nu_1^F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\nu_1^B$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\nu_2^M$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\nu_2^F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\nu_2^B$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 5
Part type $\pi_2$ rule base.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>$x_1^j$</td>
<td>IF $x_1^j$ is ready AND B4 is free THEN start $v_1^j$</td>
</tr>
<tr>
<td>Rule 2</td>
<td>$x_2^j$</td>
<td>IF $x_2^j$ is done AND (M1 is free OR M6 is free) THEN start $v_2^j$</td>
</tr>
<tr>
<td>Rule 3</td>
<td>$x_3^j$</td>
<td>IF $x_3^j$ is done AND (M5 is free THEN start $v_3^j$</td>
</tr>
<tr>
<td>Rule 4</td>
<td>$x_4^j$</td>
<td>IF $x_4^j$ is done AND (M7 is free OR M8 is free) THEN start $v_4^j$</td>
</tr>
<tr>
<td>Rule 5</td>
<td>$x_5^j$</td>
<td>IF $x_5^j$ is done AND (M5 is free THEN start $v_5^j$</td>
</tr>
<tr>
<td>Rule 6</td>
<td>$x_6^j$</td>
<td>IF $x_6^j$ is done THEN release $v_{out}$</td>
</tr>
</tbody>
</table>
shown in (30)–(33). The jobs and possible resource assignments of part type \( p_2 \) are reported in Table 4. The IF–THEN rule bases of \( p_1 \) and \( p_2 \) are reported in Tables 2 and 5, respectively.

\[
A^{1(0)} = \begin{bmatrix}
10.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4.3 & 7.4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 7.7 & 6.2 & 5.2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4.9 & 6.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(30)

\[
A^{2(0)} = \begin{bmatrix}
4.76 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3.89 & 3.48 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4.26 & 5.68 & 4.37 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 5.21 & 0 & 4.46 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(31)

The rule bases of two part types are now represented by means of Boolean matrices \( F^1, F^2, F^{1d}, F^{2d} \) as shown in 1, 2, 3 and 34, 35, 36, respectively. It is noted that the contents of matrices \( S^1 \) and \( S^2 \) are omitted for brevity.

\[
F_p = \begin{bmatrix}
v_1^p & v_2^p & v_3^p & v_4^p & v_5^p \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

(34)

\[
F^{1d} = \begin{bmatrix}
x_1^{1d} & x_2^{1d} & x_3^{1d} & x_4^{1d} & x_5^{1d} \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(35)

\[
F^{2d} = \begin{bmatrix}
x_1^{2d} & x_2^{2d} & x_3^{2d} & x_4^{2d} & x_5^{2d} \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(36)

Concatenating these mission matrices, the overall system's matrix description is obtained by

\[
F_v = \begin{bmatrix}
F_1 v_0 & F_2 v_0 & F^{1d} v_0 & F^{2d} v_0 & F_u v_0 & 0 \\
0 & F_1 v_0 & F_2 v_0 & F^{1d} v_0 & F^{2d} v_0 & 0 \\
0 & 0 & F_1 v_0 & F_2 v_0 & F^{1d} v_0 & F^{2d} v_0 \\
0 & 0 & 0 & F_1 v_0 & F_2 v_0 & F^{1d} v_0 \\
0 & 0 & 0 & 0 & F_1 v_0 & F_2 v_0 \\
0 & 0 & 0 & 0 & 0 & F_1 v_0 \\
\end{bmatrix}
\]

(37)

5.3. Experiment results

In this section, we will show the test results of applying the proposed framework and demonstrate the solution quality by comparing with the schedules obtained by three dispatching rules, CR, and PR approaches. The considered dispatching rules include shortest processing time first (SPT), least energy cost first (LEC), and first come first served (FCFS) (Sule, 2007). The SPT rule sequences the jobs so that the job which takes the shortest time to process is first to be performed. The LEC rule gives the priority to the job which has the least energy cost to be scheduled first. The FCFS rule sequences the jobs starting with the current time period and working forward. In predictive-reactive approaches, the baseline schedule and the reschedules are generated using particle swarm optimization (Fang & Lin, 2013). In CR approach, a totally new schedule is regenerated, while only the jobs and resources which are affected by the power consumption uncertainties are rescheduled in PR approach. Since jobs are not preemptive, the reschedules are only applied if all ongoing jobs from the previous schedules have been completed. The rescheduling is triggered if any \( a_{ij}^q(t) \) increases. On the contrary, MTME and dispatching rules are triggered if any job is completed and there are sets of jobs and resources to be dispatched. These approaches are used to solve (12) and the obtained objective values are compared under three probability distributions of \( g_b \), \( l_{ij}^q \), \( r_{ij}^q \), namely, the Weibull, exponential, and truncated normal distributions. The distribution mean and variance are reported in Table 6.

<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>70,140</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>40,80</td>
<td>50,100</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>80,160</td>
<td>60,120</td>
<td>70,140</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>50,100</td>
<td>-</td>
<td>70,140</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50,100</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>60,120</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>80,160</td>
<td>90,180</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>70,140</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[ \text{dev} = \frac{J - J_{CR}}{J_{CR}} \times 100\%, \]  

(38)

where \( J_{CR} \) is the optimal cost obtained by the CR. The mean interrupted time is computed as follows

\[ T_{\text{mean}} = \frac{\sum_{i=1}^{n} T_i(s)}{n}, \]  

(39)

where \( T_i \) is the computational time to generate the \( i \)th reschedule, and \( n \) denotes the number of reschedules. For all scheduling approaches, the results are obtained after 20 test runs. This experiment is carried on a digital computer equipped with Intel Core i7 processor and 32 gigabyte random-access memory. All computations are done using MATLAB. 

The deviation from Pareto optimality of the schedules generated by three dispatching rules, CR, PR, and MTME in the Weibull distribution, truncated normal distribution, and exponential distribution are reported in Figs. 6–8, respectively, while test results of mean interrupted time are provided in Table 7. For simplicity, \( \phi(\pi_1) = \phi(\pi_2) \) and \( w_1 = w_2 = 4.75 \) are assumed, and suitable due dates are selected for each test cases. \( n = 40 \) is set, i.e., each power consumption will increase for 40 times during the entire production.

In terms of deviation from Pareto optimality, MTME outperforms the three dispatching rules for all test cases. The PR outperforms MTME when the batch size is small (short schedules), but the reverse is observed when the batch size is larger than 60 parts (long schedules). In terms of mean interrupted time, MTME achieves less than 1 s for all distributions and batch sizes, while the PR and CR cause prohibitive interrupted time.

5.4. Scalability of MTME for scheduling large-scale FMS

In this section, the usability of our proposed MTME is verified with different sizes of FMS, i.e., different numbers of jobs and resources. Let rewrite the MTME in a standard 0–1 linear programming model as follows.

\[ \text{maximize } w^Tz. \]  

(40)

subject to \( Gz \leq h \),

\[ z_i \in \{0, 1\}, \forall z_i, \]  

(41)

where \( z \) represents the vector of variables, \( w \) and \( h \) are vectors of coefficients, and \( G \) is a matrix of coefficients. Computational results for each 0–1 submodel are reported in Table 8, where \( \lambda \) is the number of variables, \( \gamma \) is the number of constraints, and \( \sigma \) is the density (the ratio of the number of non-zero elements to the total number of elements) of matrix \( G \). Obviously, the values of \( \lambda, \gamma, \) and \( \sigma \) depend on the numbers of jobs and resources in the FMS.

It can be seen that in most problem sizes, each submodel takes less than 2 s of computational time. In addition, the ADEC and the MTME are programmed in the same language of Boolean matrices and vectors, which allows fast deployment of scheduling decisions.
6. Conclusion

In this paper, an integrated control and reactive scheduling framework is proposed for improving energy efficiencies in FMS subjected to power consumption uncertainties. Our proposed framework was rigorously justified with mathematical formulation and its effectiveness was evaluated based on an industrial stamping system, where the stamping parts are various types of VCM yokes used in commercial HDD actuators. The obtained schedules are compared with three dispatching rules and two rescheduling approaches. The experiment results verify that MTME outperforms three dispatching rules in terms of deviation from Pareto optimality and reduces interrupted time significantly as compared to rescheduling approaches. Our future work includes implementation of the proposed framework on an industrial plant.

Acknowledgement

This work was supported in part by Singapore MOE AcRF Tier 1 Grants R-263-000-A44-112 and R-263-000-A52-112.

References


