# Approximate dynamic programming for an energy-efficient parallel machine scheduling problem

Mojtaba Heydar[a,b,*], Elham Mardaneh[a], Ryan Loxton[a]

[a] *School of Electrical Engineering, Computing, and Mathematical Sciences, Curtin University, Bentley, WA 6102, Australia*

[b] *ARC Training Center for Transforming Maintenance through Data Science, Curtin University, Bentley, WA 6102, Australia*

**Abstract**

In this paper, we propose an approximate dynamic programming approach for an energy-efficient unrelated parallel machine scheduling problem. In this scheduling problem, jobs arrive at the system randomly, and each job's ready and processing times become available when an order is placed. Therefore, we consider the online version of the problem. Our objective is to minimize a combination of makespan and the total energy costs. The energy costs include cost of energy consumption of machines for switching on, processing, and idleness. We propose a binary program to solve the optimization problem at each stage of the approximate dynamic program. We compare the results of the approximate programming approach against an integer linear programming formulation of the offline version of the scheduling problem and an existing heuristic method suitable for scheduling problem with ready times. The results show that the approximate dynamic programming algorithm outperforms the two off-line methods in terms of solution quality and computational time.

*Keywords:* Scheduling, approximate dynamic programming, energy-efficient production planning, makespan, time-of-use tariff

## 1. Introduction

In recent years, many countries have introduced new regulations to enforce sustainability recognition and implementation in manufacturing industries to reduce resource consumption and carbon footprint. Energy efficiency in production systems from the operational standpoint and new advanced equipment have been pursued as two strategies to make the reduction possible. However, in recent years, more efforts have been devoted to operational energy efficiency in the form of production planning and scheduling through time-of-use (TOU) tariff schemes set by energy suppliers (Anghinolfi et al. 2020, Cao et al. 2020).

Since the implementation of TOU tariff by utility suppliers, many companies, especially energy-intensive industries, have applied energy-aware production scheduling policies to reduce the electric-

*Corresponding author

*Email addresses:* mojtaba.heydar@curtin.edu.au (Mojtaba Heydar), elham.mardaneh@curtin.edu.au (Elham Mardaneh), r.loxton@curtin.edu.au (Ryan Loxton)

ity cost by shifting from peak hours operations to medium-pick or off-pick hours operations (Hadera et al. 2015, Zhang et al. 2014). This creates a new interesting and challenging scheduling problem in which there is a cost associated with each period (this cost is also called green metric or green performance measure (Anghinolfi et al. 2020)). These problems belong to the class of bi-criteria scheduling problems with machine assignment costs, and there is a trade-off between the two criteria (Leung et al. 2012). Therefore, to measure the quality of each feasible schedule, a cost-oriented objective should be considered in addition to the conventional performance measures such as completion time and tardiness. Wan and Qi (2010) proved that a single machine scheduling problem with time-dependent scheduling cost is an NP-hard problem.

This paper deals with a class of online scheduling problem. In this class, unlike the off-line problems, the required information is not available at the time of making decisions. For example, the jobs to be processed, their arrival and processing times, and machine availability time may not be known (Albers 2009). In this paper, we extend the work of Nezami et al. (2017) by considering random arrival time and processing time of jobs, and present an approximate dynamic programming (ADP) approach to solve the online bi-objective unrelated parallel machine scheduling problem. The objectives are minimizing the makespan and total energy cost, which is defined based on the TOU concept for measuring energy consumption. It is assumed that jobs arrive at the system randomly. Therefore, the ready times of jobs are not known in advance, but it is assumed that they follow a known probability distribution.

The main contributions of this paper are twofold: this is the first study that considers an online energy-efficient unrelated parallel machine scheduling problem; and this is the first study in the literature that presents an approximate dynamic programming algorithm for this online scheduling problem. The remainder of this paper is organized as follows. A survey of previous works is given in Section 2. The online scheduling problem is defined and formulated in Section 3. Section 4 presents the approximate dynamic programming algorithm designed for solving the problem at hand. The computational analysis is presented in Section 5. Finally, concluding remarks are given in Section 6.

## 2. Literature review

Energy-efficient scheduling has been the subject of many studies in the past few years, and the number is still growing, indicating the importance and popularity of the subject. An excellent survey on the energy-efficient scheduling problem was written by Gahm et al. (2016). They considered studies from 1990 to 2015 and introduced energetic coverage, energy supply, and energy demand as three dimensions for classifying the literature. In another survey, two approaches were defined for classifying energy-efficient scheduling problems known as input and output-oriented approach (Giret et al. 2015). According to this classification, the input-oriented approaches mostly deal with performance measures and energy minimization objectives, whereas output-oriented approaches deal with environmental waste and pollution minimization along with performance measures (Nezami et al. 2017). This study, however, does not consider the output-related objectives such as carbon emission

minimization. Several studies that applied the concept of energy-efficient scheduling in real-world industries involved tempered glass process (Liu et al. 2020), glass production plant (Wang et al. 2016, 2020b), steel production plant (Cao et al. 2020, Hadera et al. 2015), steelmaking-refining-continuous casting (Tan and Liu 2014), moulding industry (Paolucci et al. 2017), and scrap steel melting process (Gajic et al. 2017, Yang et al. 2018).

In recent years, several studies have addressed the energy-efficient scheduling problem in different machine settings. Single machine scheduling problem was the first class of energy-efficient scheduling problems that were studied. Cheng et al. (2016) developed a bi-objective (makespan and total electricity costs) mixed-integer linear program for a single-machine batch scheduling problem based on the TOU concept and developed an $\epsilon$-constraint method for generating Pareto optimal solutions. Wang et al. (2016) extended their problem by considering two new decision variables, which are processing in machine low-temperature mode and processing in machine high-temperature mode. They approximated the Pareto fronts by two decomposition-based heuristics. Shrouf et al. (2014) considered processing, idle, and shutdown as three machine modes in their formulation, which was solved by a genetic algorithm. Aghelinejad et al. (2018) solved the non-preemptive case of Shrouf et al. (2014) by a genetic algorithm where scheduling at the machine and job levels were considered. Cheng et al. (2017a) also considered idle and processing modes in their MILP formulation and used an $\epsilon$-constraint-based heuristic to handle instances of larger size. Fang et al. (2016) developed an approximation algorithm for the uniform and speed-scaling single machine problem. A weighted sum of the total energy costs and total completion time of all jobs on all machines combined was studied by Mouzon et al. (2007) and Rubaiee and Yildirim (2019).

Energy-efficient scheduling problem in the flow-shop setting was studied extensively (for example, see Mansouri and Aktas (2016), Wang et al. (2020b), Zhang et al. (2014), Zheng et al. (2020)). Gong et al. (2020) studied the energy-efficient flexible flow shop with worker flexibility as it has a significant impact on the performance of the production system. Therefore, they considered makespan, worker or human factor cost, and energy indicator for their proposed MILP model. Then they developed a hybrid evolutionary algorithm for solving large-scale instances of the problem. Other variants of the energy-efficient flow-shop scheduling problem which have been studied include flexible flow-shop with sequence-dependent setup times (Jiang and Wang 2019, Mansouri et al. 2016), distributed no-idle flow-shop (Chen et al. 2019), permutation flow-shop (Lu et al. 2017), and hybrid flow-shop (Li et al. 2019, Luo et al. 2013).

Energy-efficient job-shop scheduling problem and its variants have been studied by many researchers over the past few years. A dynamic energy-efficient job-shop scheduling problem was addressed by Nouiri et al. (2018). They proposed a particle swarm optimization approach to minimize makespan and energy efficiency simultaneously. Golpîra et al. (2018) studied a robust energy-efficient production scheduling problem in which supply and demand are uncertain and developed risk-based robust mixed-integer linear program. Zhang et al. (2017) proposed a dynamic game theory approach to minimize makespan, machine load, and energy consumption. Meta-heuristic approaches such as genetic algorithms (Liu et al. 2016, May et al. 2015, Salido et al. 2016, Zhang and

Chiong 2016), evolutionary algorithms (Wang et al. 2020a), memetic algorithms (Abedi et al. 2020), and shuffled frog-leaping algorithms (Lei et al. 2017) were also used for solving the energy-efficient job shop scheduling problem. Other variants of job-shop scheduling that received attention from researchers include job shop scheduling with transportation constraint (Dai et al. 2019), scheduling and lot-sizing (Giglio et al. 2017), flexible job shop scheduling (Dai et al. 2019, Meng et al. 2019, Mokhtari and Hasani 2017), and job shop scheduling with variable spindle speed (Yin et al. 2017).

The subject of this research, energy-efficient parallel machine scheduling problem, has received attention over the past few years. For a detailed literature review on the parallel machine environment see Safarzadeh and Niaki (2019) and the references therein. The off-line unrelated parallel machine problem solved by genetic algorithm was studied by Moon et al. (2013). Ding et al. (2015) proposed a MILP for the parallel machine scheduling problem based on TOU with makespan and total energy cost as the objectives. They developed a column generation algorithm to deal with the complexity. Their model was then improved by Cheng et al. (2017b). Che et al. (2017) developed a mixed-integer linear program for the off-line uniform parallel machine scheduling problem based on the concept of TOU and proposed a two-stage heuristic to handle the large-scale cases. Nezami et al. (2017) developed a mixed-integer nonlinear program for unrelated parallel machine scheduling with total completion time, total energy cost, and total power consumption as three objective functions. Anghinolfi et al. (2020) developed a time-indexed MILP based on the concept of TOU for an off-line identical parallel machine scheduling problem, where no-preemption is allowed, with minimization of makespan and total energy cost. Then, they introduced a heuristic procedure called Split-Greedy Heuristic, which is a constructive algorithm enhanced with a local search. Safarzadeh and Niaki (2019) studied an off-line uniform parallel machine problem with minimization of makespan and green production costs of all machines combined. Instead of using TOU, they introduced three cost factors called green processing cost, green working cost of machines per unit time, and green working cost of machines per unit of job processing. Using $\epsilon$-constraint technique, an improvement of the heuristic algorithm of Ji et al. (2013) was used to generate Pareto optimal solutions. The current study and the work of Nezami et al. (2017), to the best of our knowledge, are the only two studies that consider machines' modes (states) such as processing, idle, on/off, various processing speeds, and TOU policies simultaneously.

The subject of this study is related to the applications of approximate dynamic programming. For some recent applications of approximate dynamic programming see (Al-Kanj et al. 2020, Godfrey and Powell 2002, Heydar et al. 2021, Jenkins et al. 2021, Koch and Klein 2020, Saure et al. 2012, Yuan and Tang 2017). Ronconi and Powell (2010) is the only study that applies the approximate dynamic programming approach to the online single machine tardiness scheduling problem. The current study differs from their work in several ways. First, this study considers the parallel machine scheduling problem. Second, this study considers an energy-efficient scheduling problem where the performance measure is makespan. Third, we consider different modes of machines in this problem. Fourth, we use a double-pass approximate dynamic programming algorithm. Finally, we consider a 0-1 program at each stage of the problem to find the best decision where the objective

includes cost-per-stage and future costs.

## 3. The scheduling problem

In this section, we present the online energy-efficient unrelated parallel machine scheduling problem formally. We mostly use the conventions defined in Nezami et al. (2017). It is assumed that there are $\mathcal{M} = \{1, 2, \ldots . M\}$ unrelated parallel machines available to process a set of available jobs denoted by $\mathcal{J}$. This implies that all jobs should be processed eventually. Each job arrives at the system randomly and its job ready time (arrival time) is denoted by $r_j \geq 0$. Each job $j$ has an integer random machine-dependent processing time denoted by $p_{mj} > 0$. If machine $m$ is not suitable to process job $j$, then we have $p_{mj} = \infty$. The processing time becomes available once the job appears at the system. We also assume that the planning horizon is discretized into time slots $t \in T$, where $T := \{0, 1, \ldots, T^H\}$, with equal lengths of $L$. The newly arrived jobs are ready at the beginning of each time interval. Moreover, without loss of generality, it is assumed that the processing times are divisible by the interval length. No preemption is allowed. Finally, all machines are off at the beginning of the planning horizon, and each machine can be switched on only once over the planning horizon. The objective is to assign the arriving jobs to the machines in order to minimize makespan and total energy costs.

We base the formulation of the problem on the following assumptions. We consider the TOU, meaning that energy prices vary in peak/off-peak periods. The energy price for each time slot $t \in T$ is denoted by $E_t$. We also assume that each machine can process only one job during each time slot $t \in T$. There is an electricity consumption rate for each machine's mode during each time slot $t$. The electricity consumption for processing and idle mode are $\mu_t^m$ and $\kappa_t^m$, respectively. Similarly, the parameter $\lambda_t^m$ denotes the required electricity to switch on machine $m$ at the beginning of time slot $t$. Finally, we assume that the time it takes to start a machine is negligible, however, during the period that the machine is switched on the average energy consumption increases. All problem notations are given in Table 1.

This online scheduling problem can be modeled as a Markov Decision Process, in which all aspects of a stochastic optimization problem are available (see Ronconi and Powell (2010) and Powell (2007, Chapter 5)). These aspects are state variable, decision variables, an objective function, transition function, and exogenous information. In the following, we explain these aspects of the online scheduling problem at hand.

The state of the system at time slot $t \in T$, denoted by $S_t$, is defined by set $S_t = \{J_t, A_t, \Upsilon_t\}$. In this set, $J_t$ is the set of unprocessed jobs at the beginning of time slot $t$ and $A_t = \{(r_j, p_{mj}) | j \in J_t, m \in M_t\}$, where $M_t$ is the set of available machines at time slot $t$. The state of machines, $\Upsilon_t$, is defined by $(y_t^m, z_t^m, \rho_t^m, a_t^m)$, where $y_t^m \in \{0, 1\}$, $z_t^m \in \{0, 1\}$, and $\rho_t^m \in \{0, 1\}$ denote whether machine $m$ is in idle mode, just switched on, and in processing mode at time slot $t$. In this convention, $a_t^m$ shows the time when machine $m$ is available to process which can be any time at or after $t$. For instance, let us assume that job $j$ with processing time $p_{mj} = 2$ is scheduled on machine $m$ at $t - 1$, then machine $m$ will be available at time $a_t^m = t + 1$. In this case, machine $m$ will be

Table 1: Notations used in the paper

| Sets | |
|---|---|
| $\mathcal{J}$ | Set of all jobs |
| $J_t$ | Set of unprocessed job at the beginning of time $t$, $J_t \subset \mathcal{J}$ |
| $J_t^C$ | Set of available jobs that can be scheduled at time $t$, $J_t^c \subseteq J_t$ |
| $J_t^{Ca}$ | Set of jobs scheduled at time $t$, $J_t^{Ca} \subseteq J_t^C$ |
| $T$ | The discretized planning horizon such that $\|T\| = T^H$ |
| $\mathcal{M}$ | Set of all machines |
| **Parameters** | |
| $r_j$ | Arrival (ready) time of job $j \in \mathcal{J}$ |
| $p_{mj}$ | Processing time of job $j$ on machine $m$ |
| $\kappa_t^m$ | Power consumption of machine $m$ at time $t$ in idle mode |
| $\lambda_t^m$ | Power consumption of machine $m$ at time $t$ when switches on |
| $\mu_t^m$ | Power consumption of machine $m$ at time $t$ in processing mode |
| $E_t$ | Price of energy at time $t$ |
| $L$ | The length of each interval $t \in \mathcal{T}$ |
| $\zeta_i$ | The weight of objective $i$ |
| $S_t$ | State of the system at time $t$ |
| $A_t$ | The release time and processing time of unprocessed jobs at time $t$ |
| $M_t$ | Set of available machine at time $t$ |
| $W_t$ | The exogenous information that arrive at time $t-1$ for time $t$ |
| $\Upsilon_t$ | State of machines at the beginning of time $t$ |
| $a_t^m$ | Shows the time that machine $m$ becomes available |
| $\rho_t^m$ | A binary parameter indicating whether machine $m$ is processing (=1) at time $t$ or not (=0) |
| $y_t^m$ | A binary parameter indicating whether machine $m$ is idle (=1) at time $t$ or not (=0) |
| $z_t^m$ | A binary parameter indicating whether machine $m$ is switched on (=1) at time $t$ or not (=0) |
| **Decisions** | |
| $x_{tj}^m$ | $\begin{cases} 1 & \text{if machine } m \text{ starts processing on job } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$ |
| $\omega_{tj}^m$ | $\begin{cases} 1 & \text{if machine } m \text{ is processing job } j \text{ during time } t \\ 0 & \text{otherwise} \end{cases}$ |

occupied during time slots $t$ and $t+1$, i.e. $\rho_t^m = \rho_{t+1}^m = 1 = 1 - y_t^m = 1 - y_{t+1}^m \Longrightarrow y_t^m = y_{t+1}^m = 0$. The following remark defines the constraints of each machine during each time slot:

*Remark* 1. During each time slot $t$, a particular machine $m$ is either in processing mode or idle mode, mathematically, $\rho_t^m + y_t^m \leq 1$.

The inequality of Remark 1 implies that if machine $m$ is off during time slot $t$, then $\rho_t^m = y_t^m = 0$. On the other hand, if machine $m$ is on during time slot $t$, then it is either processing, $\rho_t^m = 1 = 1 - y_t^m$, or idle, $\rho_t^m = 0 = 1 - y_t^m$. Furthermore, the following condition between two consecutive time slots holds

$$\rho_{t-1}^m + y_{t-1}^m = \rho_t^m + y_t^m - z_t^m, \tag{1}$$

with the initial condition $\rho_0^m + y_0^m - z_0^m = 0$. This initial condition is derived from the assumption that all machines are off at the beginning of the planning horizon. The condition in Eq. (1) states that if machine $m$ is in processing or idle mode for the first time at time $t$, it should be switched

on first.

Denote by $J_t^C$ the subset of available jobs that can be scheduled at the beginning of time slot $t$. The set of decisions is defined on set $J_t^C$ and represented as follows

$$x_{tj}^m = \begin{cases} 1 & \text{job } j \in J_t^C \text{ is scheduled on available machine } m \\ & \text{at time } t; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For any other jobs not in $J_t^C$ or unavailable machines, i.e. $a_t^m > t$, we have $x_{tj}^m = 0$. At the beginning of each time slot $t$, candidate jobs are assigned to the available machines. If the number of available machines is less than the number of candidate jobs, i.e. $|M_t| < |J_t^C|$, only a subset of jobs can be scheduled and the rest will be considered in the next time slot. This subset is denoted by $J_t^{Ca}$. The decision given by (2) is defined for the candidate jobs in $J_t^C$, and the available machines at time slot $t$. According to our definition of state of machines, a machine is available only if $a_t^m \leq t$, meaning that we can consider a machine only if it was freed before or just at the beginning of time slot $t$. By using decision variable $x_{tj}^m$, we decide whether or not to assign job j to machine $m$ at time slot $t$. If $x_{tj}^m = 1$, then states of the machine should be updated as $\{\rho_t^m = 1, y_t^m = 0\}$, for $t, t+1, t+2, \ldots, t+p_{mj}-1$, and $a_t^m = t + p_{mj}$, for $t+1, t+2, \ldots, t+p_{mj}$.

Using this kind of decision variables, we define a policy $\pi \in \Pi$ (the family of policies) that satisfy some assignment constraints. The set of these decisions under policy $\pi$ is defined by action space $X_t^\pi$.

The contribution of the objective function at time slot $t$ for makespan, $C_{\max,t}$, is calculated by

$$C_t^1(S_t, X_t^\pi) = C_{\max,t} = \max_{\substack{m \in \mathcal{M} \\ j \in |J_t^{Ca}|}} \left\{ \max\left\{r_j, a_t^m\right\} + p_{mj} x_{tj}^m \right\}. \tag{3}$$

Since $r_j \leq t$ for all $j \in J_t^{Ca}$, and for all available machines we have $a_t^m = t$, therefore, job $j$ can only start at time $t$ and the first term in Eq. (3) is $\max\{r_j, a_t^m\} = t$. As a result, Eq. (3) is simplified to

$$C_t^1(S_t, X_t^\pi) = C_{\max,t} = \max_{\substack{m \in \mathcal{M} \\ j \in |J_t^{Ca}|}} \left\{ t + p_{mj} x_{tj}^m \right\},$$

which is equivalent to the constraints (45) in the off-line model of the appendix.

Similarly, we define per stage contribution of the objective function with respect to total energy cost, $TEC^t$, as follows

$$C_t^2(S_t, X_t^\pi) = EC_t = E_t \sum_{m=1}^{M} \left( \mu_t^m \sum_{j=1}^{|J_t^{Ca}|} \omega_{tj}^m + \kappa_t^m y_t^m + \lambda_t^m z_t^m \right),$$

where, $\omega_{tj}^m \in \{0, 1\}$ is an auxiliary variable indicating if job $j$ is being processed by machine $m$ at time slot $t$. The three terms in the above equation are energy cost associated with processing

mode, idle mode, and starting machine $m$ at time slot $t$, and $E_t$ is the price of energy during time slot $t$. The processing state of machine $m$ due to machine capacity during each time slot $t$ will be updated as

$$\sum_{j=1}^{|J_t^C|} \omega_{tj}^m = \rho_t^m. \tag{4}$$

According to this condition, if machine $m$ is in processing mode, it should process only one job from the candidate list. By applying condition (4), the total energy cost can be written as follow:

$$C_t^2(S_t, X_t^\pi) = EC_t = E_t \sum_{m=1}^{M} \left( \mu_t^m \rho_t^m + \kappa_t^m y_t^m + \lambda_t^m z_t^m \right), \tag{5}$$

The total cost at $t$ is then defined by

$$C_t(S_t, X_t^\pi) = \sum_{i=1}^{2} \zeta_i C_t^i(S_t, X_t^\pi), \tag{6}$$

where $\zeta_i$ is the weight of the objective $i = \{1, 2\}$. It should be pointed out that the two criteria in Eq. (6) have different dimensions. Therefore, the weights $\zeta_1$ and $\zeta_2$ should be determined or normalized in such way that all terms of the resulting single objective problem have the same dimensions. The weighted sum of Eq. (6) can be written as $C_t^1(S_t, X_t^\pi) + (\frac{\zeta_2}{\zeta_1}) C_t^2(S_t, X_t^\pi)$ which means that the relativity of weights are important (Cohon 1978).

Since we consider the online version of the energy-efficient unrelated parallel scheduling problem, the exogenous information is the arrival of the new jobs to the system for each time slot $t$. In any stochastic system a transition from state $S_t$ to state $S_{t+1}$ is defined by the following function

$$S_{t+1} = S^M(S_t, X_t^\pi, R_{t+1}), \tag{7}$$

where $R_{t+1}$ denotes the exogenous information that becomes available after time $t$ and before time $t+1$. In this problem the $S^M(\cdot)$ is defined as follows

$$\begin{cases} J_{t+1} = (J_t \setminus J_t^{Ca}) \cup J_{t+1}^\omega, \\ A_{t+1} = \{(r_j, p_{jm}) | j \in J_{t+1}, m \in M_{t+1}\}, \\ a_{t+1}^m = \max(a_t^m + \sum_{j \in J_t^{Ca}} p_{mj} x_{tj}^m, t+1), \forall m \in M_{t+1}, \end{cases} \tag{8}$$

In this transition function, $J_{t+1}^\omega$ is the set of jobs that become known to us between $t$ and $t+1$, with their associated information such as ready time, $r_j$, and processing times, $p_{mj}$. Using $a_{t+1}^m$, we are able to update the state of machine $m \in M_{t+1}$ during the time slot $t$.

The last step in the modeling of the stochastic decision problem is to define the objective function. Our objective is to find the best policy $\pi$ to minimize the expected value of a linear

Table 2: Data for example 1

| $j$ | Time information received | $r_j$ | $p_{1j}$ | $p_{2j}$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 4 | 6 |
| 2 | 0 | 1 | 1 | 3 |
| 3 | 1 | 3 | 2 | 3 |
| 4 | 1 | 3 | 3 | 4 |
| 5 | 3 | 3 | 4 | 3 |

combination of the makespan and total energy cost as follows

$$\min_{\pi \in \Pi} \mathbb{E}\Big\{ \sum_{t=1}^{T^H} C_t(S_t, X_t^\pi) | S_0 \Big\}, \tag{9}$$

where $\mathbb{E}$ denotes the expectation value. Using Bellman's equation, we can write the optimization problem as follows

$$V_t(S_t) = \min_{x_t \in X_t^\pi} \Big( C_t(S_t, x_t) + \mathbb{E}\Big\{ V_{t+1}(S_{t+1}) | S_t \Big\} \Big), \tag{10}$$

where $V_t(S_t)$ is the minimum value of the value function at state $S_t$. The optimization problem (10) can be solved using the backward dynamic programming. Reasons that make the computing of the exact value of Eq. (10) intractable include: (1) the size of state space $S_t$, (2) the size of the decision vector $x_t$, and (3) the calculation of the expectation of the future cost due to the size of decision vector $x_t$ and exogenous information $R_t$. The three reasons are known as the curse of dimentionality (e.g., see Powell (2007, Chapter 4) for details on three curses of dimensionality). One approach to resolve this curse of dimensionality issue is to use approximate dynamic programming (ADP). Before introducing the proposed approximate dynamic programming approach, we provide a small example.

**Example 1.** *Consider a parallel machine environment with $|\mathcal{M}| = 2$ machines. There are $|\mathcal{J}| = 5$ jobs that we want to schedule over $T = 20$ time slots with $E_t = 1$. The jobs information are shown in Table 2. The second column of this table shows the time when exogenous information is available. For example, at time $t = 0$, information for jobs $j = 1$ and $j = 2$ are received but only $j = 1$ is available for processing. We also assume that $\mu_t^m = \kappa_t^m = \lambda_t^m = 1, \forall m \in \mathcal{M}, \forall t \in T$. At time $t = 0$, only information of jobs $j = 1, 2$ are available, hence we need to apply a sequential decision making approach. One simple way is to use a myopic approach. In this approach we can make decision as jobs arrive at the system without considering the impact of our decision on the future cost. In this example, job $j = 1$ is assigned to machine $m = 1$. It results in $C_{max,1} = 4$ and $EC_1 = 2$. At time $t = 1$, the second job, $j = 2$, arrives and it is assigned to machine $m = 2$, since this machine is the only available machine. The objective values are $C_{max,2} = 4$ and $EC_2 = 3$, respectively. At times $t = 2$, no job arrives, and three jobs, $j = 3, 4, 5$, arrive at time $t = 3$. At this time slot we have to wait until $t = 4$, so that both machines become available. At time $t = 4$, from $3 \times 3! = 18$ different possible schedules, by scheduling jobs $j = 3, 4$ on machine $m = 1$ and job*

Table 3: The value of objective functions for each time slot $t$

| | $t$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10-20 |
| $C_{max,t}$ | | 4 | 4 | 4 | 4 | 7 | 7 | 9 | 9 | 9 | - |
| $EC_t$ | | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

*$j = 5$ on machine $m = 2$, the values of makespan for each time slot will be $C_{max,5} = C_{max,6} = 7$, $C_{max,7} = C_{max,8} = C_{max,9} = 9$. Similarly, the energy costs for each time slot can be calculated. It should be noted that in this schedule, the first machine is idle for time slots $t = 8, 9$, and for $t \geq 10$ both machines are idle. The sequential decision process using this myopic policy is depicted in Figure 1. After scheduling all five jobs the problem maskespan and total energy cost are 9 and 41, respectively. The weighted sum of the objective value is $0.5 \times (9 + 41) = 25$. The values of makespan and energy costs are given in Table 3. It should be noted that the schedule obtained using the myopic policy is one of the 480 possibilities depending on the number of machines used. These number of schedules are known only if all information is available at the beginning of the planning horizon, which is the case for the off-line deterministic problems. Given this fact, other possibilities such as delaying the processing of jobs will increase the number of schedules, hence the complexity of the problem at hand. For the sake of comprehensiveness, we solve this example with the integer program of the Appendix A. Again, we assume that both objectives are equally important ($\zeta_1 = \zeta_2 = 0.5$). As depicted in Figure 2, the schedule starts as last as possible to reduce the total energy cost. This schedule allows the machines to be turned on towards the end of planning horizon, which eliminates the idle cost of machines. The overall objective of offline model is 17.5 with $C_{\max} = 20$ and $EC = 15$. Although better, this value can be obtained only if all information (exogenous and endogenous) are known prior to the decision making process; otherwise, this schedule might be infeasible if any new job shows up.*

## 4. Approximate Dynamic Programming

Approximate dynamic programming has been successfully applied to many stochastic optimization problems (e.g., see (Alkaabneh et al. 2020, Silva and de Souza 2020)). ADP is an algorithmic strategy that moves forward through time to approximate the future expected value of the value function. Its origin dates back to the birth of dynamic programming when Bellman introduced the concept of the curse of dimensionality (Ronconi and Powell 2010). There are two approximation approaches for solving dynamic programs, namely optimization-based and simulation-based (Saure et al. 2012). The simulation-based ADP, which is the focus of this paper, uses the concept of a post-decision state to cope with the expectation in the Bellman's equation (see (Powell 2007, 2016, 2019)). Our solution approach to the online unrelated parallel machine scheduling problem is based on the double-pass value iteration algorithm within the framework of approximate dynamic programming. In particular, we combine the approximate dynamic programming and mathematical

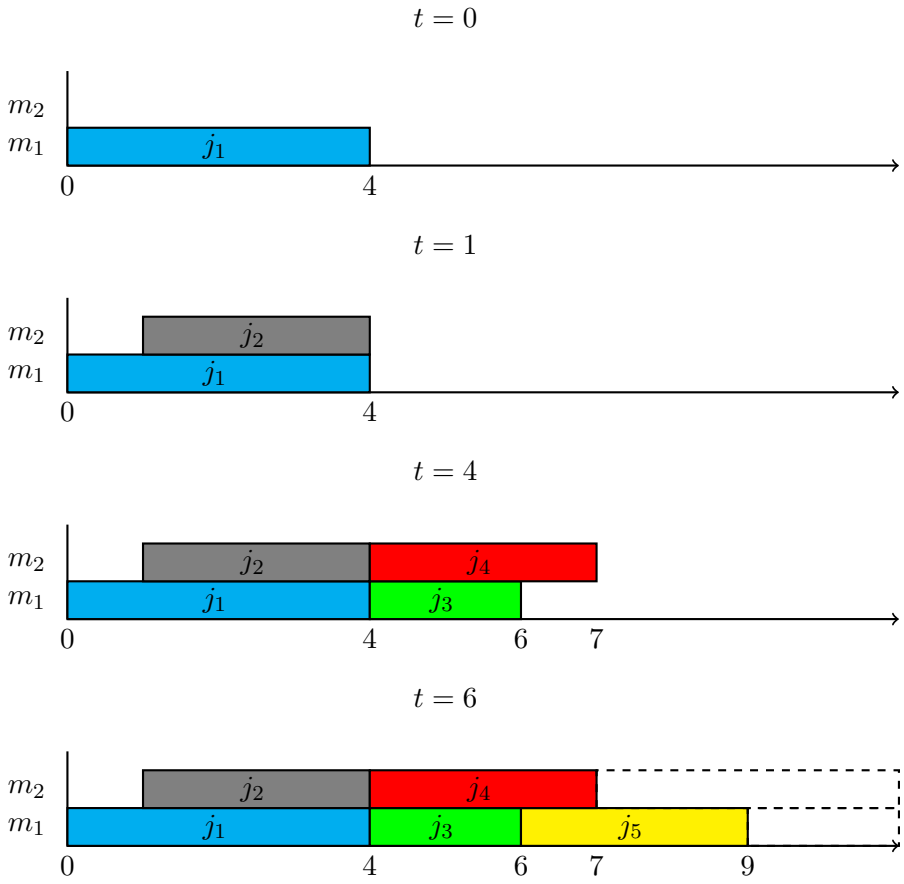


Figure 1: The sequential decisions made in Example 1

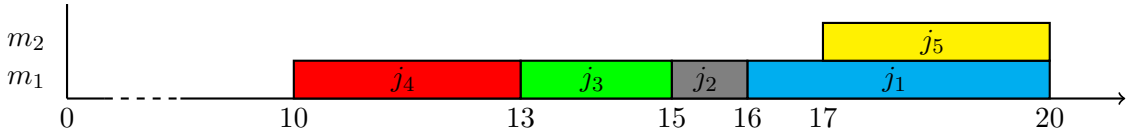


Figure 2: The optimal schedule of Example 1 obtained by the IP of Appendix A

programming to handle the three curses of dimnesionality. In this section, we present the details of our approach.

### *4.1. Post-decision state*

We use the concept of post-decision state to reduce the size of the decision outcome space and intractability of expected future cost. The post-decision state, $S_t^x$, refers to the state of the system immediately after decision $x_t$ is made, but before the arrival of any new exogenous information $R_{t+1}$. In other words, the post-decision state is a state between states $S_t$ and $S_{t+1}$. The post-decision state for this problem is defined by

$$S_t^x = \{J_t^x, A_t^x, M_t^x\}, \tag{11}$$

where $J_t^x$, $A_t^x$, and $M_t^x$ are updated as follows

$$\begin{cases} J_t^x = (J_t \setminus J_t^C), \\ A_t^x = \{(r_j, p_{jm}) | j \in J_t^x, m \in M_t^x\}, \\ M_t^x = \{m \in \mathcal{M} | x_{tj}^m = 1\}. \end{cases} \tag{12}$$

Here, $M_t^x$ denotes the post-decision states for machines with the updated states after making decision $x_t$. After this post-decision state the system will move forward to the next pre-decision state $S_{t+1}$ which is defined as follows

$$\begin{cases} J_{t+1} = J_t^x \cup J_{t+1}^\omega, \\ A_{t+1} = A_t^x \cup \{(r_j, p_{jm}) | j \in J_{t+1}^\omega, m \in M_{t+1}\}, \\ M_{t+1} = M_t \setminus M_t^x \cup \{m \in M : a_{t+1}^m = t+1\}. \end{cases} \tag{13}$$

Using the concept of post-decision state, we are able to write the Bellman's equation as follows

$$V_t(S_t) = \min_{x_t \in X_t^\pi} \Big( C_t(S_t, x_t) + V_t^x(S_t^x) \Big), \tag{14}$$

$$V_t^x(S_t^x) = \mathbb{E}\Big\{ V_{t+1}(S_{t+1}) | S_t^x \Big\}, \tag{15}$$

where $V_t^x(S_t^x)$ is the value function approximation for being in post-decision state $S_t^x$. The expectation value in Eq. (15) still makes the calculation of the value function intractable. Therefore, it is necessary to estimate it based on the simulation-based approach which it benefits from statistical learning algorithms (Hulshof et al. 2016, Ronconi and Powell 2010). This estimation converts the stochastic optimization problem into a deterministic one which is easier to solve because it

removes the expected future cost. Next, we explain one method for approximating the expectation in Eq (15).

*4.2. Value function approximation*

To reduce the size of space-action space for a particular decision, we need to approximate the value of post-decision state $V_t^x(S_t^x)$. We use a statistical learning algorithm that approximates the value function over $N$ iterations. Let $\overline{V}_t^n(S_t^x)$ denotes an estimation for $V_t^x(S_t^x)$ at iteration $n$ of the algorithm, then equation (15) can be expressed as

$$\overline{V}_t^n(S_x^t) = \mathbb{E}\Big\{V_{t+1}(S_{t+1})|S_t^x\Big\}.$$

Using this new definition and the original definition of Bellman's equation given by Eq. (10), we can write approximate dynamic programming equation as

$$\hat{v}_t^n = \min_{x_t \in X_t^\pi} \Big(C_t(S_t, x_t) + \overline{V}_t^{n-1}(S_t^x)\Big), \tag{16}$$

and

$$x_t^n = \arg\min_{x_t \in X_t^\pi} \Big(C_t(S_t, x_t) + \overline{V}_t^{n-1}(S_t^x)\Big), \tag{17}$$

where $x_t^n$ is the value of decision vector $x_t$ that optimizes value function Eq. (16) at iteration $n$ of the algorithm (Mes and Rivera 2017). At iteration $n$, we have to solve Eq (17) to obtain the best decision that minimizes the value $\hat{v}_t^n$ for state $S_t$. Finally, using Eq. (16) we can update the estimate for the value function at time $t-1$ and iteration $n$ by

$$\overline{V}_{t-1}^n \longleftarrow U^n\big(\overline{V}_{t-1}^{n-1}, S_{t-1}^{x,n}, \hat{v}_t^n\big). \tag{18}$$

The update function $U^n(\cdot)$ depends on the value approximation at time $t-1$ from iteration $n-1$, $\overline{V}_{t_1}^{n-1}$, post-decision state at time slot $t-1$ at iteration $n$ denoted by $S_{t-1}^{x,n}$, and the objective value estimation (16). Now the main challenge is to calculate $\overline{V}_{t-1}^n$ in Eq. (18).

There are various algorithms to estimate the value function $\overline{V}_{t-1}^n$. In this paper, we use the concept of basis function. In the basis function, some particular features of a system that can capture required information of the system, which has impacts on the value function, are considered. Then, the value function calculation using an affine combinations of basis functions is defined by

$$\overline{V}_t^n(S_t^x) = \sum_{f \in \mathcal{F}} \eta_f^n \phi_f(S_t^x), \tag{19}$$

where $\mathcal{F}$ is the set of features, $\eta_f^n$ is a weight for feature $f \in \mathcal{F}$, and $\phi_f(S_t^x)$ is the value of the basis function for a feature $f$ given the post-decision state $S_t^x$. In this approximation of the value function, the weight $\eta_f^n$ is updated recursively at each iteration $n$ of the algorithm.

There are different approaches for calculating and updating $\eta_f^n$, among which the recursive least

square method is the most effective approach (Hulshof et al. 2016, Powell 2007). Depending on the data, there are two methods for the recursive least square methods. If the data are stationary, the method puts equal weights on each data; while in the non-stationary data, the more recent data receives higher weights (Hulshof et al. 2016)(for details of this approach see Powell (2007)). At each iteration $n$ of the algorithm, the weights $\eta_f^n$ are updated recursively using following equation

$$\eta_f^n = \eta_f^{n-1} - \frac{1}{\gamma^n} B^{n-1} \phi_f(S_t^x) \left( \overline{V}_{t-1}^{n-1}(S_{t-1}^x) - \hat{v}_t^n \right), \quad \forall f \in \mathcal{F}, \tag{20}$$

where $B^{n-1}$ is a $|\mathcal{F}| \times |\mathcal{F}|$ matrix which is updated recursively by

$$B^n = \frac{1}{\alpha^n} \left( B^{n-1} - \frac{1}{\gamma^n} \left( B^{n-1} \phi(S_t^x) \big(\phi(S_t^x)\big)^T B^{n-1} \right) \right). \tag{21}$$

The parameter $\gamma^n$ is defined and updated by

$$\gamma^n = \alpha^n + \phi(S_t^x)^T B^{n-1} \phi(S_t^x). \tag{22}$$

At iteration $n = 1$, we set $B^0 = \epsilon I$, where $I$ is the identity matrix and $\epsilon$ is a very small constant. The parameter $\alpha^n$, $n = 1, 2, \ldots, N$, shows that the observations are either stationary or non-stationary and is defined by

$$\alpha^n = \begin{cases} 1 & \text{stationary;} \\ \\ 1 - \frac{\delta}{n} & \text{non-stationary.} \end{cases} \tag{23}$$

The parameter $\delta$, which tunes our approximation, should be determined in advance. In our model, we assume that our data are non-stationary; therefore, we set $\alpha^n = 1 - \frac{\delta}{n}$. The approximate dynamic programming algorithm, known as the double-pass algorithm, is given in Algorithm 1 (Heydar et al. 2021, Hulshof et al. 2016, Powell 2007). In this algorithm, we can arbitrarily assign values to parameters $\sigma$, $\epsilon$, and $\eta_t^0$. At each iteration of the algorithm, we solve the decision problem to obtain the post-decision states, and then using the exogenous information and the transition function $S^M(\cdot)$, the next pre-state will be obtained. In this study, we use probability distribution to generate the required information. The exogenous information for the problem at hand is the number of jobs along with their ready time and processing time on different machines. It should be noted that the exogenous information can be obtained from the historical data or the true distribution if it is known. Once we have cost-per-stage values $C_t(S_t, x_t)$ for all $t$, we backward through time from the end of the planning horizon, $T^H$, and calculate $\hat{v}_t^n$. This will enable us to update weights using Eqs. (20)-(23), and calculate the value approximation $\overline{V}_t^n(S_t^x)$ by Eq. (19). At the end of the algorithm, when the weights reach their steady-state values, the optimization method can be applied to the problem that we want to solve.

As mentioned earlier, we use the basis functions to estimate the value function of the post-decision state. The basis function has an impact on the quality of the ADP algorithm. The basis

function is a set of features that captures the required information of the system. For the problem in this study, we define the indicator function $\mathbf{1}_{\{.\}}$ to capture the state of machine $m$ at time $t$. Then, in each state, we count the number of machines in each processing, idle, and switched on mode. Also, we keep track of the number of jobs already assigned from the set of candidate jobs, denoted by $J_t^{Ca}$, and the number of unassigned jobs. We use a linear approximation for value function in Eq. (16) as follows

$$\begin{aligned}\overline{V}_t^n(S_t^x) :=& \eta_{t,0}^n + \eta_{t,1}^n |J_t^{Ca}| + \sum_{m\in\mathcal{M}} \eta_{t,2}^n \mathbf{1}_{\{y_t^m=1\}} \\ &+ \sum_{m\in\mathcal{M}} \eta_{t,3}^n \mathbf{1}_{\{z_t^m=1\}} + \sum_{m\in\mathcal{M}} \eta_{t,4}^n \mathbf{1}_{\{\rho_t^m=1\}} + \eta_{t,5}^n (|J_t| - |J_t^{Ca}|). \end{aligned} \tag{24}$$

In this equation, $\eta_{t,0}^n$ is a constant and $|J_t^{Ca}| = \min\{|J_t^C|, |M_t|\}$ is the set of jobs that are assigned to machines at time slot $t$. The function $\mathbf{1}_{\{y_t^m=1\}}$ returns 1 if machine $m$ is idle at time $t$. Similarly, $\mathbf{1}_{\{\rho_t^m=1\}}$ and $\mathbf{1}_{\{z_t^m=1\}}$ are defined for processing and switched-on modes, respectively. The last term in Eq. (24) keeps track of the number of unassigned job at time slot $t$. Using this basis function, we can approximate the current and future costs that are imposed on the value function. In the next section, this new value function will be used to solve the decision problem.

### *4.3. Optimal decision vector*

Making the best decision in each pre-decision state $S_t$ is the most important and challenging part of the ADP as the dimensions of the problem increase. Therefore, it is of great importance to design powerful solution techniques. For this purpose, we develop a 0-1 programming model with the approximated value function given by Eq. (16) as an objective function. Then, we use CPLEX 12.8 to solve this 0-1 program for each time slot $t$ of the system. In this formulation, in addition to decision variable $x_{tj}^m$, we define other binary decision variables for state variables $\rho_t^m$, $y_t^m$, and $z_t^m$. This new mathematical model is defined by

$$\min \quad \zeta_1 C_{\max,t} + \zeta_2 TEC_t - L x_{tj}^m + \overline{V}_t^n(S_t^x) \tag{25}$$

subject to

$$\sum_{m\in M_t} x_{tj}^m \le 1, \quad \forall j \in J_t^C, \tag{26}$$

$$\sum_{j\in J_t^C} x_{tj}^m \le \rho_t^m, \quad \forall m \in M_t, \tag{27}$$

$$y_t^m + \rho_t^m \ge \ddot{y}_{t-1}^m + \ddot{\rho}_{t-1}^m, \quad \forall m \in M_t, \tag{28}$$

$$\rho_t^m + y_t^m - z_t^m \le \ddot{\rho}_{t-1}^m + \ddot{y}_{t-1}^m + \ddot{z}_{t-1}^m, \quad \forall m \in M_t, \tag{29}$$

$$y_t^m + \rho_t^m \le 1, \quad \forall m \in M_t, \tag{30}$$

$$z_t^m + \sum_{\theta=0}^{\theta=t-1} \ddot{z}_\theta^m \le 1, \quad \forall m \in M_t, \tag{31}$$

**Algorithm 1** The double-pass approximate dynamic programming algorithm

Initialize the value function approximation $\overline{V}_t^0(S_t^x)$, which mean initialize $\eta_t^0$ and $B_t^0 = \epsilon I$.
Set $n = 1$.
Initialize $S_0$.
**while** $n \leq N$ **do**
  **for** $t = 0, 1, \ldots, T^H$ **do**
    Solve

$$x_t^n = \arg\min_{x_t \in X_t^\pi} \Big( C_t(S_t, x_t) + \overline{V}_t^{n-1}(S_t^x) \Big).$$

    **if** $t < T^H$ **then**
      Generate the next state using $S_{t+1} = S^M(S_t, X_t, W_{t+1})$.
    **end if**
  **end for**
  Let $\hat{v}_{T^H+1}^n = 0$.
  **for** $t = T^H, T^H - 1, \ldots, 1$ **do**
    Compute

$$\hat{v}_t^n = C_t(S_t, x_t) + \hat{v}_{t+1}^n,$$

    Update

$$\overline{V}_{t-1}^n \longleftarrow U^n\big(\overline{V}_{t-1}^{n-1}, S_{t-1}^x, \hat{v}_t^n\big),$$

    and then calculate

$$\overline{V}_{t-1}^n(S_{t-1}^x) = \sum_{f \in \mathcal{F}} \phi_f(S_{t-1}^x)\eta_f^n.$$

  **end for**
**end while**
Return $\overline{V}_t^N(S_t^x), \quad \forall t \in T^H$.

$$t + \sum_{m \in M_t} p_{mj} x_{tj}^m \leq C_{\max,t}, \quad \forall j \in J_t^C. \tag{32}$$

The values of $\ddot{x}_{(t-1),j}^m$, $\ddot{\rho}_{t-1}^m$, $\ddot{y}_{t-1}^m$, and $\ddot{z}_{t-1}^m$ are known from previous stage with initial values of $\ddot{x}_{0,j}^m = \ddot{\rho}_0^m = \ddot{y}_0^m = \ddot{z}_0^m = 0$. In this formulation, constraints (26)- (27) are the job assignment and machine capacity constraints. Writing constraint (26) in inequality form refrains the model from infeasibility in cases where number of jobs to process is greater than number of available machines, $|J_t| > |M_t|$. On the other hand, having constraint (26) in its current form may render some machines useless. To resolve this, we added each decision variable $x_{tj}^m$ with a very small coefficient ($L$) to the objective function. This guarantees that $x_t \neq \emptyset$ at time slot $t$. Constraint (28) updates the state of machine $m$, and constraint (29) determines whether or not machine $m$ must be switched on. Constraint (30) guarantees that the switched-on machine $m$ can be either in its idle mode or processing mode. This constraint is equivalent to Remark 1. Constraint (31) ensures that each machine can be switched on only one time over the planning horizon. Constraint (32) links the completion time of each job to the makespan. This decision problem possesses the Markov property.

It should be noted that it is not required to solve the 0-1 program at some time slots. If $J_t = \emptyset$ or $\rho_t^m = 1, \forall m \in M$, then the only decision is *do nothing*. In this situation, the value function can be easily computed using the system information and Eq. (25).

Now let us explain how the strategy functions. At each time slot $t$, we solve the 0-1 program for the available jobs with $r_j \leq t$ and their known processing times. These available jobs are those

which were in the system before $t-1$ or became availabe between $t-1$ and $t$. Using the solution of the 0-1 program, the contribution of the cost from time slot $t$ to time slot $t+1$ is calculated. The impact of the current decision on the time slots after $t+1$ is then updated and calculated using the value function approximation $\overline{V}_t$. At time slot $t+1$ new jobs may arrive, where $r_j \geq t+1$, and the set of available jobs will be updated. At this point, we find an optimal schedule using 0-1 program. This new schedule with its respective optimal objective value will be used to update the parameters of value function at time slot $t$, which can be used in the next iteration of the ADP algorithm.

Now we present a small illustrative example to better explain the optimization problem and algorithm presented in this study.

**Example 2.** *We consider an instance with $|\mathcal{M}| = 3$ and $T = 13$. In each time $t$, random jobs arrived according to a discrete distribution $DU(0,3)$ inclusive. We assume that system is empty at beginning of the process. To solve this problem using the proposed approximate dynamic programming, we start with an initial value of $\eta_f^n = \mathbb{J}_f$, where $\mathbb{J}$ is an all-one matrix of size $f$, and generate $N = 3000$ iterations or sample paths. Each sample path refers to a sequence of realization of the information over planning horizon. In this example, for each sample path $n$, starting from time $t = 0$ (current time) forward, number of arrivals is revealed using discrete uniform distribution. Once new information, such as job completion time and/or arrival time, is revealed, $S_t$, the (pre-)state of the system at the beginning of time $t$ is obtained. Then, using the IP model of Section 4.3, a decision is made. This converts the system from pre-decision state to the post-decision state $S_t^x$. Then by arriving new information (denoted by $R_{t+1}$), the system moves to the next state $S_{t+1}$. This process continues until the end of planning horizon, when the forward pass is completed (lines 5-10 of Algorithm 1). Through this forward pass, we make the best decision and calculate the per-stage cost for all $t \in T$. Then, the backward pass starts by setting $\hat{v}_{T+1} = 0$, and solve $\hat{v}_t^n = C_t(S_t, x_t) + \hat{v}_{t+1}^n$. Then, using Eqs. (20)- (23), the parameter $\eta_f^n$ is updated which is used for updating the value function approximation at time $t-1$, $\overline{V}_{t-1}^n$ (lines 11-16 of Algorithm 1). When the algorithm finalizes, $\eta_f^{3000}$ converges to its approximation and $\overline{V}_0^{3000}$ is calculated using $\sum_{f=1}^{6} \eta_f^{3000} \phi_f(S_0^x)$. The results of each iteration through double-pass algorithm are given Table 4.*

## 5. Computational study

In this section, we present the performance of the ADP algorithm, as depicted in Algorithm 1. One way of measuring the performance of the ADP algorithm is to compare its results with a deterministic model of the stochastic problem (Godfrey and Powell 2002, Powell 2007, 2008). For this purpose, we compare the results of the ADP with the deterministic counterpart of the problem at hand which is presented in Appendix A as an integer linear program (referred to as IP hereafter). Since the IP is incapable to solve large instances, we also compare the ADP algorithm with a simulation-based heuristic algorithm for the energy efficient scheduling problem (Paolucci et al. 2017). Their algorithm, to the best of our knowledge, is the only algorithm that can handle jobs

Table 4: Set of sample paths for Example 2

| $n$ | $\overline{v}_0$ | $t=0$ | $t=1$ | $t=2$ | $t=3$ | $t=4$ | $t=5$ | $t=6$ | $t=7$ | $t=8$ | $t=9$ | $t=10$ | $t=11$ | $t=12$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\eta_0$ | $\eta_1$ | $\eta_2$ | $\eta_3$ | $\eta_4$ | $\eta_5$ | $\eta_6$ | $\eta_7$ | $\eta_8$ | $\eta_9$ | $\eta_{10}$ | $\eta_{11}$ | $\eta_{12}$ |
| 1 | 14.9 | $\begin{pmatrix}14.9\\42.9\\1\\1\\1\\1\end{pmatrix}$ | $\begin{pmatrix}17.5\\1\\-18.5\\1\\20.5\\7.5\end{pmatrix}$ | $\begin{pmatrix}5.3\\1\\-12\\1\\14\\9.7\end{pmatrix}$ | $\begin{pmatrix}4.3\\7.6\\1\\1\\1\\1\end{pmatrix}$ | $\begin{pmatrix}7.2\\1\\-17.5\\1\\19.5\\7.2\end{pmatrix}$ | $\begin{pmatrix}3.8\\1\\-7.4\\1\\9.4\\6.6\end{pmatrix}$ | $\begin{pmatrix}3.4\\3.4\\1\\1\\1\\8.3\end{pmatrix}$ | $\begin{pmatrix}4.3\\1\\-8.8\\1\\10.8\\10.8\end{pmatrix}$ | $\begin{pmatrix}2.6\\2.6\\1\\1\\1\\4.2\end{pmatrix}$ | $\begin{pmatrix}4.1\\4.1\\1\\1\\1\\4.1\end{pmatrix}$ | $\begin{pmatrix}3.4\\3.4\\3.4\\1\\-1.4\\1\end{pmatrix}$ | $\begin{pmatrix}2.1\\1\\-1.2\\1\\3.2\\1\end{pmatrix}$ | $\begin{pmatrix}1.1\\1\\0.7\\1\\1.3\\1\end{pmatrix}$ |
| 2 | 15.3 | $\begin{pmatrix}15.3\\43.5\\1.3\\1\\0.7\\1\end{pmatrix}$ | $\begin{pmatrix}7.9\\1\\-19.2\\1\\21.2\\7.5\end{pmatrix}$ | $\begin{pmatrix}5.5\\1.4\\-12\\1\\14\\9.9\end{pmatrix}$ | $\begin{pmatrix}4.5\\7.6\\0.4\\1\\1.6\\1.2\end{pmatrix}$ | $\begin{pmatrix}7.1\\0.9\\-17.5\\1\\19.5\\7.1\end{pmatrix}$ | $\begin{pmatrix}3.9\\1\\-7.8\\1\\9.8\\7.3\end{pmatrix}$ | $\begin{pmatrix}3.5\\3.4\\0.7\\1\\1.3\\8.9\end{pmatrix}$ | $\begin{pmatrix}4.2\\0.9\\-8.8\\1\\10.8\\10.5\end{pmatrix}$ | $\begin{pmatrix}2.7\\2.8\\1\\1\\1\\4.5\end{pmatrix}$ | $\begin{pmatrix}4.1\\4\\0.8\\1\\1.2\\4.2\end{pmatrix}$ | $\begin{pmatrix}3.5\\3.4\\3.1\\1\\-1.1\\1.3\end{pmatrix}$ | $\begin{pmatrix}2.2\\1\\-1.3\\1\\3.3\\1.1\end{pmatrix}$ | $\begin{pmatrix}1.2\\1\\0.7\\1\\1.3\\1\end{pmatrix}$ |
| 3 | 15.4 | $\begin{pmatrix}15.4\\43.7\\1.6\\1\\0.4\\1\end{pmatrix}$ | $\begin{pmatrix}8.1\\1\\-19.5\\1\\21.5\\7.5\end{pmatrix}$ | $\begin{pmatrix}5.6\\1.4\\-12.1\\1\\14.1\\9.9\end{pmatrix}$ | $\begin{pmatrix}4.7\\7.8\\0.5\\1\\1.5\\1.2\end{pmatrix}$ | $\begin{pmatrix}7.3\\0.9\\-17.8\\1\\19.8\\7.1\end{pmatrix}$ | $\begin{pmatrix}3.9\\1\\-7.9\\1\\9.9\\7.3\end{pmatrix}$ | $\begin{pmatrix}3.6\\3.7\\0.7\\1\\1.3\\8.9\end{pmatrix}$ | $\begin{pmatrix}4.3\\0.9\\-8.9\\1\\10.9\\10.5\end{pmatrix}$ | $\begin{pmatrix}2.7\\2.8\\0.9\\1\\1.1\\4.5\end{pmatrix}$ | $\begin{pmatrix}4.2\\4\\0.7\\1\\1.3\\4.2\end{pmatrix}$ | $\begin{pmatrix}3.5\\3.4\\2.9\\1\\-0.9\\1.3\end{pmatrix}$ | $\begin{pmatrix}2.2\\1\\-1.4\\1\\3.4\\1.1\end{pmatrix}$ | $\begin{pmatrix}1.2\\1\\0.7\\1\\1.3\\1\end{pmatrix}$ |
| 4 | 15.5 | $\begin{pmatrix}15.5\\43.9\\1.7\\1\\0.3\\1\end{pmatrix}$ | $\begin{pmatrix}8.3\\1.2\\-19.5\\1\\21.5\\7.5\end{pmatrix}$ | $\begin{pmatrix}5.7\\1.5\\-12.1\\1\\14.1\\9.9\end{pmatrix}$ | $\begin{pmatrix}4.7\\7.8\\0.5\\1\\1.5\\1.2\end{pmatrix}$ | $\begin{pmatrix}7.4\\0.9\\-18\\1\\20\\7.1\end{pmatrix}$ | $\begin{pmatrix}3.9\\1\\-7.9\\1\\9.9\\7.3\end{pmatrix}$ | $\begin{pmatrix}3.7\\3.8\\0.7\\1\\1.3\\8.9\end{pmatrix}$ | $\begin{pmatrix}4.3\\0.9\\-8\\1\\10.9\\10.5\end{pmatrix}$ | $\begin{pmatrix}2.8\\2.8\\0.8\\1\\1.2\\4.5\end{pmatrix}$ | $\begin{pmatrix}4.2\\4\\0.7\\1\\1.3\\4.2\end{pmatrix}$ | $\begin{pmatrix}3.6\\3.4\\2.9\\1\\-0.9\\1.3\end{pmatrix}$ | $\begin{pmatrix}2.2\\1\\-1.4\\1\\3.4\\1.1\end{pmatrix}$ | $\begin{pmatrix}1.1\\1\\0.7\\1\\1.3\\1\end{pmatrix}$ |
| 5 | 15.7 | $\begin{pmatrix}15.7\\44\\2.1\\1\\-0.1\\1\end{pmatrix}$ | $\begin{pmatrix}8.5\\1.6\\-19.5\\1\\21.5\\7.7\end{pmatrix}$ | $\begin{pmatrix}5.8\\1.5\\-12.4\\1\\14.4\\10.4\end{pmatrix}$ | $\begin{pmatrix}4.8\\7.8\\0.2\\1\\1.8\\1.7\end{pmatrix}$ | $\begin{pmatrix}7.3\\0.9\\-17.8\\1\\19.8\\6.6\end{pmatrix}$ | $\begin{pmatrix}3.9\\0.9\\-7.9\\1\\9.9\\7.2\end{pmatrix}$ | $\begin{pmatrix}3.7\\3.8\\0.7\\1\\1.3\\8.9\end{pmatrix}$ | $\begin{pmatrix}4.3\\0.9\\-8.9\\1\\10.9\\10.5\end{pmatrix}$ | $\begin{pmatrix}2.8\\2.8\\0.8\\1\\1.2\\4.5\end{pmatrix}$ | $\begin{pmatrix}4.2\\4\\0.7\\1\\1.3\\4.2\end{pmatrix}$ | $\begin{pmatrix}3.6\\3.4\\2.9\\1\\-0.9\\1.3\end{pmatrix}$ | $\begin{pmatrix}2.2\\1\\-1.4\\1\\3.4\\1.1\end{pmatrix}$ | $\begin{pmatrix}1.1\\1\\0.7\\1\\1.3\\1\end{pmatrix}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| 2995 | 143.7 | $\begin{pmatrix}143.7\\263.9\\54.6\\1\\-52.6\\1\end{pmatrix}$ | $\begin{pmatrix}-6.4\\117.1\\218.7\\1\\-216.7\\-99.9\end{pmatrix}$ | $\begin{pmatrix}54\\82.7\\15.9\\1\\-13.9\\25.7\end{pmatrix}$ | $\begin{pmatrix}44.1\\57.7\\-7.9\\1\\9.9\\7.9\end{pmatrix}$ | $\begin{pmatrix}31.1\\29.7\\-23.3\\1\\25.3\\-7.3\end{pmatrix}$ | $\begin{pmatrix}22.9\\22.5\\-15.9\\1\\17.9\\2.7\end{pmatrix}$ | $\begin{pmatrix}17.6\\17.1\\-13.8\\1\\15.8\\4.3\end{pmatrix}$ | $\begin{pmatrix}12.8\\15.3\\-12.5\\1\\14.5\\4.1\end{pmatrix}$ | $\begin{pmatrix}10.3\\12.4\\-9.9\\1\\11.9\\5.2\end{pmatrix}$ | $\begin{pmatrix}5.5\\5.9\\-12.1\\1\\14.1\\5.2\end{pmatrix}$ | $\begin{pmatrix}8.3\\11.2\\-4.9\\1\\6.9\\3.3\end{pmatrix}$ | $\begin{pmatrix}-0.6\\2.9\\-7.1\\1\\9.1\\2.1\end{pmatrix}$ | $\begin{pmatrix}7.4\\11.2\\4.8\\1\\-2.8\\1.5\end{pmatrix}$ |
| 2996 | 143.7 | $\begin{pmatrix}143.7\\263.9\\54.6\\1\\-52.6\\1\end{pmatrix}$ | $\begin{pmatrix}-6.6\\117.1\\219.3\\1\\-217.3\\-100.3\end{pmatrix}$ | $\begin{pmatrix}54.1\\82.7\\15.5\\1\\-13.5\\26.4\end{pmatrix}$ | $\begin{pmatrix}44.1\\57.7\\-7.9\\1\\9.9\\7.8\end{pmatrix}$ | $\begin{pmatrix}31.1\\29.7\\-23.3\\1\\25.3\\-7.4\end{pmatrix}$ | $\begin{pmatrix}22.9\\22.5\\-15.8\\1\\17.8\\2.2\end{pmatrix}$ | $\begin{pmatrix}17.5\\17\\-13.8\\1\\15.8\\3.7\end{pmatrix}$ | $\begin{pmatrix}12.8\\15.3\\-12.5\\1\\14.5\\4.2\end{pmatrix}$ | $\begin{pmatrix}10.2\\12.4\\-9.9\\1\\11.9\\4.6\end{pmatrix}$ | $\begin{pmatrix}5.5\\5.9\\-12.1\\1\\14.1\\5.1\end{pmatrix}$ | $\begin{pmatrix}8.3\\11.2\\-4.9\\1\\6.9\\3.1\end{pmatrix}$ | $\begin{pmatrix}-0.5\\2.9\\-7.1\\1\\9.1\\2.2\end{pmatrix}$ | $\begin{pmatrix}7.3\\11.2\\4.8\\1\\-2.8\\1.3\end{pmatrix}$ |
| 2997 | 143.7 | $\begin{pmatrix}143.7\\264\\54.6\\1\\-52.6\\1\end{pmatrix}$ | $\begin{pmatrix}-6.8\\116.9\\219.3\\1\\-217.3\\-100.5\end{pmatrix}$ | $\begin{pmatrix}54.1\\82.7\\15.5\\1\\-13.5\\26.3\end{pmatrix}$ | $\begin{pmatrix}44.1\\57.7\\-7.9\\1\\9.9\\7.9\end{pmatrix}$ | $\begin{pmatrix}31.2\\29.8\\-23.3\\1\\25.3\\-7.1\end{pmatrix}$ | $\begin{pmatrix}22.9\\22.5\\-15.9\\1\\17.9\\2.4\end{pmatrix}$ | $\begin{pmatrix}17.5\\17\\-13.8\\1\\15.8\\3.6\end{pmatrix}$ | $\begin{pmatrix}12.8\\15.3\\-12.5\\1\\14.5\\4.1\end{pmatrix}$ | $\begin{pmatrix}10.3\\12.4\\-10.1\\1\\12.1\\4.7\end{pmatrix}$ | $\begin{pmatrix}5.5\\5.9\\-12.1\\1\\14.1\\5\end{pmatrix}$ | $\begin{pmatrix}8.3\\11.2\\-4.9\\1\\6.9\\3.1\end{pmatrix}$ | $\begin{pmatrix}-0.5\\2.9\\-7.1\\1\\9.1\\2.2\end{pmatrix}$ | $\begin{pmatrix}7.4\\11.2\\4.8\\1\\-2.8\\1.3\end{pmatrix}$ |
| 2998 | 143.7 | $\begin{pmatrix}143.7\\264\\54.6\\1\\-52.6\\1\end{pmatrix}$ | $\begin{pmatrix}-6.6\\116.9\\219.3\\1\\-217.3\\-100.5\end{pmatrix}$ | $\begin{pmatrix}54.2\\82.9\\15.6\\1\\-13.6\\26.3\end{pmatrix}$ | $\begin{pmatrix}44.1\\57.7\\-7.9\\1\\9.9\\7.9\end{pmatrix}$ | $\begin{pmatrix}31.2\\29.9\\-23.3\\1\\25.3\\-7.1\end{pmatrix}$ | $\begin{pmatrix}23\\22.5\\-16.1\\1\\18.1\\2.5\end{pmatrix}$ | $\begin{pmatrix}17.5\\17\\-13.8\\1\\15.8\\3.5\end{pmatrix}$ | $\begin{pmatrix}12.9\\15.3\\-12.7\\1\\14.7\\4.3\end{pmatrix}$ | $\begin{pmatrix}10.3\\12.4\\-10.1\\1\\12.1\\4.7\end{pmatrix}$ | $\begin{pmatrix}5.5\\5.9\\-12.1\\1\\14.1\\5.1\end{pmatrix}$ | $\begin{pmatrix}8.4\\11.2\\-5\\1\\7\\3.1\end{pmatrix}$ | $\begin{pmatrix}-0.5\\2.9\\-7.1\\1\\9.1\\2.2\end{pmatrix}$ | $\begin{pmatrix}7.4\\11.2\\4.8\\1\\-2.8\\1.3\end{pmatrix}$ |
| 2999 | 143.7 | $\begin{pmatrix}143.7\\264.1\\54.6\\1\\-52.6\\1\end{pmatrix}$ | $\begin{pmatrix}-6.7\\116.8\\219.3\\1\\-217.3\\-100.6\end{pmatrix}$ | $\begin{pmatrix}54.1\\82.8\\15.6\\1\\-13.6\\26.2\end{pmatrix}$ | $\begin{pmatrix}44.1\\57.7\\-7.9\\1\\9.9\\7.9\end{pmatrix}$ | $\begin{pmatrix}31.2\\29.9\\-23.5\\1\\25.5\\-6.9\end{pmatrix}$ | $\begin{pmatrix}23\\22.5\\-16.1\\1\\18.1\\2.5\end{pmatrix}$ | $\begin{pmatrix}17.6\\17.1\\-13.8\\1\\15.8\\3.8\end{pmatrix}$ | $\begin{pmatrix}12.9\\15.3\\-12.8\\1\\14.8\\4.5\end{pmatrix}$ | $\begin{pmatrix}10.3\\12.4\\-10.1\\1\\12.1\\4.7\end{pmatrix}$ | $\begin{pmatrix}5.6\\5.9\\-12.1\\1\\14.1\\5.1\end{pmatrix}$ | $\begin{pmatrix}8.4\\11.2\\-5\\1\\7\\3.1\end{pmatrix}$ | $\begin{pmatrix}-0.5\\2.9\\-7.2\\1\\9.2\\2.2\end{pmatrix}$ | $\begin{pmatrix}7.4\\11.2\\4.8\\1\\-2.8\\1.4\end{pmatrix}$ |
| 3000 | 143.7 | $\begin{pmatrix}143.8\\264.1\\54.6\\1\\-52.6\\1\end{pmatrix}$ | $\begin{pmatrix}-6.9\\116.8\\219.9\\1\\-217.9\\-100.9\end{pmatrix}$ | $\begin{pmatrix}54.2\\82.8\\15.3\\1\\-13.3\\26.6\end{pmatrix}$ | $\begin{pmatrix}44.1\\57.7\\-7.9\\1\\9.9\\7.9\end{pmatrix}$ | $\begin{pmatrix}31.4\\29.9\\-23.8\\1\\25.8\\-6.4\end{pmatrix}$ | $\begin{pmatrix}22.9\\22.4\\-16.1\\1\\18.1\\2.3\end{pmatrix}$ | $\begin{pmatrix}17.6\\17.1\\-13.8\\1\\15.8\\3.9\end{pmatrix}$ | $\begin{pmatrix}12.9\\15.3\\-12.8\\1\\14.8\\4.6\end{pmatrix}$ | $\begin{pmatrix}10.3\\12.5\\-10.1\\1\\12.1\\4.8\end{pmatrix}$ | $\begin{pmatrix}5.6\\5.9\\-12.2\\1\\14.2\\5.2\end{pmatrix}$ | $\begin{pmatrix}8.4\\11.2\\-5\\1\\7\\3.1\end{pmatrix}$ | $\begin{pmatrix}-0.5\\2.9\\-7.1\\1\\9.1\\2.1\end{pmatrix}$ | $\begin{pmatrix}7.4\\11.2\\4.8\\1\\-2.8\\1.4\end{pmatrix}$ |

ready time ($r_j > 0$) when multiple objectives are involved. This heuristic algorithm is presented in Appendix B. We refer to this algorithm as *Paolucci-Anghinolfi-Tonelli* algorithm or the heuristic algorithm in this paper.

All algorithms were written in Python 3.6, and all tests were run on a desktop computer with a Windows 10 Enterprise Edition 64 bit, a i7 Core processor at 2.5 GHz, and 8 GB of RAM. For the optimization part of the algorithm as well as the IP, the solver of CPLEX 12.8 with its default parameters was called into the Python code. All text files containing the problem instances used in this study are available upon request.

Several classes of instances were considered. These classes were defined based on the different levels of factors such as the length of the planning horizon, the number of jobs arriving at the system per time slot, the number of machines, and the interval from which jobs processing time on each machine is generated. These levels are given in Table 5. As shown in the table, in our numerical study, we considered three levels for time horizon which are $T^H = \{50, 100, 150\}$ units of time, and three levels for machines which are $\mathcal{M} = \{6, 10, 15\}$. We also considered three levels for jobs. In the first level, it is assumed that the number of arrivals to the system between time slot $t$ and $t+1$ follows a discrete uniform distribution $DU[0, 6]$, for the second level, the number of arrival jobs follows a discrete uniform distribution $DU[0, 12]$, and in the third level, the number of arrival jobs follows a discrete uniform distribution $DU[0, 16]$. To avoid the infeasibility, we let the jobs arrive in the first $[0, \frac{T}{2}]$ time slots. Therefore, the average number of jobs in each sample path of each instance are 75, 300, and 600 for class one, two, and three, respectively. For processing times of jobs, $p_{mj}$, we considered two levels. In the first level, processing times follow a discrete uniform distribution $DU[1, 5]$, and in the second level, processing times follow another discrete uniform distribution $DU[1, 10]$.

In our numerical study, instances are categorized in classes which are presented by a quadruple $(a\mathcal{M},\ [b, c]\mathcal{J},\ d\mathcal{T},\ [e, f]p_{mj})$, where $a$ shows the number of machines, interval $[b, c]$ indicates the discrete uniform distribution from which the arriving jobs in each time $t$ are randomly generated, $d$ shows the number of time slots in the planning horizon, and $[e, f]$ shows the discrete uniform distribution for generating processing times. We considered three classes of the problem denoted by $C1 = (6\mathcal{M},\ [0, 6]\mathcal{J},\ 50T,\ [1, 10]p_{mj})$, $C2 = (10\mathcal{M},\ [0, 12]\mathcal{J},\ 100T,\ [1, 10]p_{mj})$, and $C3 = (15\mathcal{M},\ [0, 16]\mathcal{J},\ 150T,\ [1, 5]p_{mj})$. For each class, we generated 10 instances, and for each instance, we generated $N = 3000$ sample paths. Overall, the double-pass ADP, the IP, and Paolucci-Anghinolfi-Tonelli algorithms were applied to 30 instances. From deterministic point of view, this problem can be considered as small to medium to large instances. But, more importantly, from stochastic point of view, all of these instances are categorized as large size due to the number of states, hence difficult to solve.

The price of energy per time slot $t$ was randomly generated from $E_t \sim DU[1, 10]$. The other problem parameters for instance generation such as energy consumption rates $(\lambda_t, \mu_t, \kappa_t)$ were assumed to be random for all time slots $t \in T$ throughout the experimental study and follows $DU[1, 5]$. After running several instances with different values for double-pass ADP parameters

Table 5: Levels of factors used in experimental study

| Parameters | Levels and values |
|---|---|
| $T^H$ | 50, 100, 150 |
| $\|\mathcal{M}\|$ | 6, 10, 15 |
| $\|\mathcal{J}\|$ | $\sim DU[0, a]$, where $a = 6, 12, 16$ |
| $p_{mj}$ | $\sim DU[1, b]$, where $b = 5, 10$ |
| $E_t$ | $\sim DU[1, 10]$ |
| $\lambda_t$, $\mu_t$, $\kappa_t$ | $\sim DU[1, 5]$ |

Table 6: Parameter values selected for the ADP and Paolucci-Anghinolfi-Tonelli algorithm

| Parameter | Value |
|---|---|
| ADP | |
| $\delta$ | 0.99 |
| $\epsilon$ | 0.001 |
| $\alpha^1$ | 1 |
| $\zeta_1$ | 0.8 |
| $\zeta_2$ | 0.2 |
| $\gamma_t^1$ | 1, $\forall t \in T$ |
| $\eta_f^1$ | 1, $\forall f \in \mathcal{F}$ |
| $N$ | 3000 |
| Heuristic | |
| $H$ | 500 |
| $\pi_{\min}$ | 0.05 |
| $q_0$ | 0.9 |
| $\chi$ | 0.95 |

and Paolucci-Anghinolfi-Tonelli algorithm, we chose the values depicted in Table 6.

As mentioned, in order to evaluate the performance of the double-pass ADP, we compared it against a deterministic IP model of the energy-efficient unrelated parallel machine scheduling problem given in Appendix A, and another heuristic, called Paolucci-Anghinolfi-Tonelli algorithm, which is given in Appendix B. In the former, all information were known a priori (Godfrey and Powell 2002, Powell 2007, 2008), while in the latter, the jobs arrival process were simulated using discrete-even simulation. In our ADP approach, we considered that all jobs arrive at the system over a sample path, and solved the parallel machine scheduling problem by the MILP solver of CPLEX 12.8. We used the standard setting of CPLEX except that we limited the run-time to 100 seconds for each problem of the sample path.

Table 7 shows the results of 10 randomly generated instances of the first class, $C1$. For each instance of this table, we generated $N = 3000$ sample paths. These iterations are required until the approximated objective values converges to its steady state. It is worth mentioning that, in our case, each sample path is a parallel machine scheduling problem itself. In this table, total number of jobs arrived, percentage of jobs dropped, overall objective, makespan, energy cost and computational times for each instance are depicted. In this table, the total number of jobs and time are cumulative for all 3000 sample paths, and the percentage of dropped jobs is the average

Table 7: Results for applying the ADP on C1 ($N = 3000$, $\zeta_1 = 0.8$)

| Instance | Total jobs arrived | Dropped jobs (%) | Overall objective | Makespan | Energy cost | Time (seconds) |
|---|---|---|---|---|---|---|
| 1 | 225157 | 1.01 | 1095.33 | 46 | 5292.66 | 2781.61 |
| 2 | 225010 | 0.97 | 1096.56 | 56 | 5258.77 | 3157.84 |
| 3 | 224841 | 0.96 | 1151.68 | 54 | 5542.39 | 3280.18 |
| 4 | 225006 | 1.04 | 1062.06 | 49 | 5114.32 | 2373.77 |
| 5 | 224656 | 1.02 | 1127.24 | 41 | 5472.22 | 2722.81 |
| 6 | 225149 | 0.99 | 1102.89 | 45 | 5334.49 | 2436.30 |
| 7 | 224653 | 0.97 | 1047.58 | 47 | 5049.89 | 2360.55 |
| 8 | 225366 | 0.94 | 1103.20 | 55 | 5296.01 | 2690.37 |
| 9 | 224820 | 1.07 | 1188.77 | 39 | 5787.84 | 2541.07 |
| 10 | 224629 | 1.03 | 1120.43 | 42 | 5434.17 | 3166.59 |
| Average per instance | 224929 | 1 | 1109.57 | 47.4 | 5358.28 | 2751.11 |
| Average per sample path | **74.98** | 1 | 1109.57 | 47.4 | 5358.28 | **0.92** |

of all 3000 sample paths. The overall objective, makespan, and energy cost is the converged value function of the ADP. This table also reports the average values per instance and average values per sample path. The last row of this table shows the value for each sample path. In this row, the average number of jobs per sample path and CPU time are simply obtained by $\frac{224929}{3000}$ and $\frac{2751.11}{3000}$, respectively. The reason is that, once we obtained the approximate value of $\eta_f$, we can simply solve any sample path by applying the decision problem of Sec 4.3. It should be mentioned that the value of *overall objective, Makespan, and Energy cost* are not divided by 3000 since they are approximate value after 3000 iterations where the values are converged. Finally, from this row, it can be concluded that after approximating $\eta_f$, we can find the best decision in about 0.92 seconds and schedule 99% of jobs.

Table 8 shows the results of comparison of three methods. In this table, we considered the first five instances of Table 7, and within each instance, we considered the first five sample paths. Since each sample path is a realization of a deterministic parallel machine scheduling problem, we were able to apply the IP and the heuristic methods to each sample path. These three methods can be compared via three criteria, namely, percentage of the dropped jobs, the value of the objective functions, and computational time. In this table, all values of ADP (except the CPU time) for the first five instances are copied from the associated columns in Table 7. To obtain the CPU time, we divide the CPU time of each instance by $N$, for example the CPU time of Instance 1 is obtained by $\frac{2781.61}{3000} = 0.93$. The ADP approach is better than the other two methods in processing of jobs and computational time. Although at the first glance it may seem that the IP returns the minimum values for the objective among all these three approaches, considering its rate of assignment (it assigns only 50% of jobs), its actual objective values can be higher than ADP. Therefore, we can claim that the ADP outperforms the other two approaches for all three criteria.

In our second experiment, we randomly generated 10 instances of the second class, $C2$, and the results are tabulated in Table 9. In this table, the overall objective, makespan, and energy cost is the converged value function of the ADP are reported. From last row of this table, it can be seen that the ADP can find the best schedule for more that 85% jobs in instance of 300 jobs in about 13.90 seconds. The comparison results of the ADP, the IP, and the heuristics are given in Table 10.

Table 8: Comparing results of ADP, the IP, and the heuristic for $C1$

| Instance | Sample path | ADP | | | | | IP | | | | | Heuristic | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) |
| 1 | 1 | 1.01 | 1095.33 | 46 | 5292.66 | 0.93 | 50.65 | 963.40 | 50 | 4617.00 | 84.33 | 22.07 | 952.59 | 54 | 4547.0 | 3.99 |
| | 2 | | | | | | 54.59 | 828.59 | 50 | 3942.00 | 102.53 | 26.89 | 981.79 | 52 | 4701.0 | 4.38 |
| | 3 | | | | | | 55.56 | 794.60 | 50 | 3773.00 | 34.85 | 17.93 | 909.19 | 39 | 4390.0 | 1.62 |
| | 4 | | | | | | 56.06 | 51.80 | 27 | 151.00 | 102.43 | 20.3 | 958.4 | 54 | 4576.0 | 2.69 |
| | 5 | | | | | | 54.71 | 805.00 | 50 | 3825.00 | 78.21 | 21.71 | 979.8 | 53 | 4687.0 | 2.63 |
| Average | | | | | | | **54.31** | **688.68** | **45.4** | **3261.6** | **80.47** | **21.78** | **956.35** | **50.4** | **5480.2** | **3.06** |
| 2 | 1 | 0.97 | 1096.56 | 56 | 5258.77 | 1.05 | 57.73 | 778.0 | 50 | 3690.00 | 127.92 | 46.39 | 1001.79 | 53 | 4797.0 | 6.46 |
| | 2 | | | | | | 59.88 | 49.20 | 36 | 102.00 | 93.61 | 40.92 | 975.19 | 52 | 4668.0 | 5.03 |
| | 3 | | | | | | 57.46 | 850.80 | 50 | 4054.00 | 80.75 | 38.68 | 962.59 | 55 | 4593.0 | 3.29 |
| | 4 | | | | | | 55.89 | 67.80 | 29 | 223.00 | 117.96 | 38.6 | 968.9 | 54 | 4629.0 | 3.85 |
| | 5 | | | | | | 55.60 | 892.40 | 50 | 4262.00 | 102.15 | 35.69 | 1001.79 | 53 | 4797.0 | 3.72 |
| Average | | | | | | | **56.91** | **527.64** | **43** | **24466.2** | **104.48** | **40.06** | **982.05** | **53.4** | **4696.8** | **4.47** |
| 3 | 1 | 0.96 | 1151.68 | 54 | 5542.39 | 1.09 | 52.31 | 956.60 | 50 | 4583.0 | 60.92 | 9.23 | 1099.19 | 55 | 5276.0 | 3.62 |
| | 2 | | | | | | 50.61 | 106.60 | 28 | 421.0 | 114.31 | 26.9 | 1097.6 | 53 | 5276.0 | 5.48 |
| | 3 | | | | | | 52.17 | 958.59 | 50 | 4592.9 | 94.03 | 28.91 | 1098.39 | 54 | 5276.0 | 3.9 |
| | 4 | | | | | | 52.24 | 45.79 | 29 | 112.9 | 50.15 | 21.68 | 1110.6 | 53 | 5341.0 | 2.26 |
| | 5 | | | | | | 55.04 | 826.6 | 50 | 3933 | 73.85 | 21.67 | 1115.19 | 52 | 5368.0 | 2.9 |
| Average | | | | | | | **52.47** | **578.84** | **41.4** | **3938** | **78.66** | **26.67** | **1120.19** | **57** | **5373** | **3.63** |
| 4 | 1 | 1.04 | 1062.06 | 49 | 5114.32 | 0.79 | 50.79 | 673.59 | 50 | 3167.9 | 48.19 | 0.0 | 893.59 | 53 | 4256.0 | 2.72 |
| | 2 | | | | | | 51.07 | 721.19 | 50 | 3405.9 | 75.3 | 10.14 | 922.79 | 53 | 4402.0 | 3.7 |
| | 3 | | | | | | 53.84 | 689.39 | 50 | 3246.9 | 50.12 | 8.84 | 909.39 | 54 | 4331 | 3.18 |
| | 4 | | | | | | 52.73 | 846.59 | 50 | 4032.9 | 89.78 | 12.96 | 914.9 | 52 | 4367.0 | 4.35 |
| | 5 | | | | | | 54.28 | 855.59 | 50 | 4077.9 | 79.91 | 15.55 | 920.9 | 59 | 4369.0 | 2.91 |
| Average | | | | | | | **52.54** | **757.27** | **50** | **3586.3** | **68.66** | **9.49** | **912.31** | **54.2** | **43.45** | **3.37** |
| 5 | 1 | 1.02 | 1127.24 | 41 | 5472.22 | 0.91 | 39.0 | 984.19 | 50 | 4720.9 | 103.91 | 35.37 | 1016.39 | 53 | 4870.0 | 5.51 |
| | 2 | | | | | | 46.78 | 802.59 | 50 | 3812.9 | 98.88 | 33.27 | 1039.79 | 53 | 4987.0 | 4.77 |
| | 3 | | | | | | 49.11 | 61.8 | 48 | 117 | 107.77 | 31.34 | 1039.79 | 53 | 4987.0 | 5.33 |
| | 4 | | | | | | 51.69 | 886.4 | 50 | 4232 | 92.67 | 26.41 | 1034.59 | 56 | 4949.0 | 5.54 |
| | 5 | | | | | | 51.73 | 820.79 | 50 | 3903.9 | 95.48 | 25.68 | 1025.6 | 53 | 4916.0 | 6.12 |
| Average | | | | | | | **47.66** | **711.15** | **49.6** | **3357.34** | **99.74** | **30.41** | **1031.23** | **53.6** | **4941.8** | **5.45** |

Table 9: Results for applying the ADP on C2 ($N = 3000$, $\zeta_1 = 0.8$)

| Instance | Total jobs arrived | Dropped jobs (%) | Overall objective | Makespan | Energy cost | Time (seconds) |
|---|---|---|---|---|---|---|
| 1 | 899250 | 16.20 | 4736.49 | 101 | 23278.45 | 46498.14 |
| 2 | 900780 | 16.29 | 4865.05 | 109 | 23889.23 | 45230.64 |
| 3 | 899508 | 16.05 | 4808.98 | 100 | 23644.95 | 42612.61 |
| 4 | 899861 | 16.02 | 4688.44 | 100 | 23042.18 | 39220.79 |
| 5 | 900539 | 16.19 | 4751.73 | 101 | 23354.63 | 39147.19 |
| 6 | 899404 | 15.99 | 4783.79 | 104 | 23502.97 | 39457.44 |
| 7 | 899168 | 15.94 | 4750.81 | 96 | 23370.03 | 38191.84 |
| 8 | 898277 | 16.12 | 4787.29 | 100 | 23536.47 | 36993.39 |
| 9 | 903194 | 16.52 | 4690.01 | 100 | 23050.03 | 37567.63 |
| 10 | 898116 | 16.07 | 4651.51 | 100 | 22857.56 | 52025.46 |
| Average per instance | 899809.7 | 16.14 | 4751.41 | 101.1 | 23352.65 | 41694.51 |
| Average per sample path | **299.94** | 16.14 | 4751.41 | 101.1 | 23352.65 | **13.90** |

The structure of this table is the same as Table 8. As can be seen from this table, the IP could only solve two sample paths out of 25 sample paths (which is only 8% of instances) to optimality. For the other 92%, the IP could not find even a feasible solution in 18000 seconds of computation time. Although the heuristic could solve all sample paths, its performance is inferior to the ADP for all three criteria. Therefore, this class of instances are more difficult than the instances of $C1$.

The results of the third and last experiments are tabulated in Tables 11 and 12, respectively. In this class, every sample path has on 600 jobs on average, which is comparable with other studies in the literature. Interestingly, after approximating parameter $\eta_f$, it took less than 7 seconds to find the best schedule with zero job dropped. The computational time and objective values are higher for the heuristic according to Table 12. The IP could not find any feasible solution in 18000 seconds for any 25 sample paths.

## 6. Conclusion and future research

This paper addresses an online energy-efficient unrelated parallel machine scheduling problem where random jobs arrive at the system over time. We minimize the weighted sum of makespan and total energy cost simultaneously. A double-pass approximate dynamic programming approach was developed to solve this problem. In order to make the best decision at each time slot, a bi-criteria 0-1 program was solved where the objective function is the weighted sum of makespan and total energy cost combined with the estimate of future value function expressed by basis functions. Since the dimension of the problem explodes as the size of the instances increases, it is not possible to solve a dynamic program to find the exact solution for comparison. For this purpose, we compared the results of the approximate dynamic programming algorithm with a 0-1 programming model of the problem, assuming that all information is available at the beginning of the planning horizon, as well as a heuristic developed by Paolucci et al. (2017) which is suitable for solving dynamic unrelated parallel machine scheduling with two objectives.

For numerical study, we considered three medium to large classes of problems and compared the performance of these three methods based on three different criteria. These performance criteria include the percentage of dropped jobs (those jobs which are not scheduled), the value of the

Table 10: Comparing results of ADP, the IP, and the heuristic for $C2$

| | | ADP | | | | | IP | | | | | Heuristic | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Sample path | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) |
| 1 | 1 | 16.20 | 4736.49 | 101 | 23278.45 | 15.50 | - | - | - | - | - | 32.34 | 3343.2 | 105 | 16296.0 | 37.76 |
| | 2 | | | | | | - | - | - | - | - | 37.36 | 3345.4 | 104 | 16311.0 | 35.07 |
| | 3 | | | | | | - | - | - | - | - | 34.83 | 3336.2 | 106 | 16257.0 | 26.55 |
| | 4 | | | | | | - | - | - | - | - | 36.23 | 3309.8 | 108 | 16117.0 | 29.88 |
| | 5 | | | | | | - | - | - | - | - | 36.42 | 3343.2 | 104 | 16300.0 | 33.59 |
| Average | | | | | | | | | | | | **35.44** | **3335.56** | **105.4** | **16256.2** | **32.57** |
| 2 | 1 | 16.29 | 4865.05 | 109 | 23889.23 | 15.08 | 35.41 | 186.6 | 99 | 536.9 | 16863.8 | 24.51 | 3673.2 | 104 | 17950.0 | 29.05 |
| | 2 | | | | | | - | - | - | - | - | 29.21 | 3673.6 | 104 | 17952.0 | 32.03 |
| | 3 | | | | | | - | - | - | - | - | 32.52 | 3673.6 | 104 | 17952.0 | 34.49 |
| | 4 | | | | | | - | - | - | - | - | 33.66 | 3672.8 | 103 | 17952.0 | 32.83 |
| | 5 | | | | | | - | - | - | - | - | 31.89 | 3655.8 | 104 | 17863.0 | 23.96 |
| Average | | | | | | | | | | | | **30.36** | **3669.8** | **103.8** | **17933.8** | **30.47** |
| 3 | 1 | 16.05 | 4808.98 | 100 | 23644.95 | 14.20 | 38.46 | 358.2 | 52 | 1583.0 | 17916.6 | 31.15 | 3513.9 | 104 | 17154.0 | 32.37 |
| | 2 | | | | | | - | - | - | - | - | 37.72 | 3523.8 | 105 | 17199.0 | 41.03 |
| | 3 | | | | | | - | - | - | - | - | 38.68 | 3513.8 | 105 | 17149.0 | 33.72 |
| | 4 | | | | | | - | - | - | - | - | 40.23 | 3519.2 | 106 | 17172.0 | 41.03 |
| | 5 | | | | | | - | - | - | - | - | 41.1 | 3521.9 | 105 | 17190.0 | 42.01 |
| Average | | | | | | | | | | | | **37.78** | **3518.52** | **105** | **17172.8** | **38.03** |
| 4 | 1 | 16.02 | 4688.44 | 100 | 23042.18 | 13.07 | - | - | - | - | - | 42.2 | 3162.6 | 105 | 15393.0 | 41.71 |
| | 2 | | | | | | - | - | - | - | - | 43.03 | 3182.8 | 106 | 15490.0 | 38.74 |
| | 3 | | | | | | - | - | - | - | - | 40.43 | 3203.6 | 104 | 15602.0 | 35.96 |
| | 4 | | | | | | - | - | - | - | - | 38.31 | 3167.9 | 107 | 15412.0 | 37.15 |
| | 5 | | | | | | - | - | - | - | - | 39.02 | 3171.9 | 107 | 15432.0 | 38.22 |
| Average | | | | | | | | | | | | **40.59** | **3177.76** | **105.8** | **15465.8** | **38.36** |
| 5 | 1 | 16.19 | 4751.73 | 101 | 23354.63 | 13.04 | - | - | - | - | - | 22.36 | 3277.8 | 106 | 15965.0 | 25.24 |
| | 2 | | | | | | - | - | - | - | - | 29.11 | 3154.8 | 109 | 15338.0 | 32.51 |
| | 3 | | | | | | - | - | - | - | - | 28.25 | 3343.8 | 106 | 16295.0 | 27.57 |
| | 4 | | | | | | - | - | - | - | - | 23.94 | 3313.4 | 107 | 16139.0 | 17.49 |
| | 5 | | | | | | - | - | - | - | - | 27.69 | 3257.8 | 107 | 15861.0 | 42.73 |
| Average | | | | | | | | | | | | **31.81** | **3269.52** | **107** | **15919.6** | **29.11** |

Table 11: Results for applying the ADP on C3 ($N = 3000$, $\zeta_1 = 0.8$)

| Instance | Total jobs arrived | Dropped jobs (%) | Overall objective | Makespan | Energy cost | Time (seconds) |
|---|---|---|---|---|---|---|
| 1 | 1799139 | 0.0 | 4838.71 | 80 | 23873.55 | 19093.89 |
| 2 | 1798442 | 0.0 | 4692.94 | 83 | 23132.69 | 17934.68 |
| 3 | 1800738 | 0.0 | 4570.06 | 81 | 22526.31 | 18510.46 |
| 4 | 1798376 | 0.0 | 4739.52 | 79 | 23381.62 | 18384.18 |
| 5 | 1797067 | 0.0 | 4884.62 | 79 | 24107.09 | 18577.93 |
| 6 | 1800606 | 0.0 | 4791.59 | 78 | 23645.97 | 23645.97 |
| 7 | 1796514 | 0.0 | 4651.82 | 85 | 22919.12 | 16509.95 |
| 8 | 1798873 | 0.0 | 4601.54 | 81 | 22683.72 | 19876.89 |
| 9 | 1800149 | 0.0 | 4837.14 | 79 | 23869.72 | 18324.70 |
| 10 | 1801347 | 0.0 | 4598.12 | 81 | 22666.62 | 24436.80 |
| Average per instance | 1799125.1 | 0.0 | 4720.61 | 80.6 | 23289.64 | 19529.55 |
| Average per sample path | **599.71** | 0.0 | 4720.61 | 80.6 | 23289.64 | **6.51** |

weighted objectives, and run time. These results can be summarized as follows: 1) our ability to solve every instances using the ADP after approximating parameter $\eta_f$ in a minimum amount of time, 2) the ADP drops the minimum number of jobs, while this value is over 50% for the IP and the heuristic), 3) comparing the results of the ADP and the heuristic reveals that in large scale instances the heuristic returns lower values for the objective function at the cost dropping about 50% of jobs which we can conclude that the ADP is more reliable in handling such scenarios, and 4) although the value of objective functions using the IP is better than the ADP in small instances, the IP can schedule on average 50% of jobs, therefore enforcing the IP to schedule more jobs will either deteriorate the objective value or make the problem infeasible.

For future research, an algorithms for generating Pareto solutions could be devised. The performance of the algorithm with other policies could be investigated. The reliability of machines could be considered as an extension of the current model. Also, models and algorithms for other performance criteria as well as machine setting could be developed.


## Acknowledgements

The authors wish to thank the two anonymous referees for several very helpful comments that strengthened this paper in many ways. The first author is partially supported by the Australian Research Council through the Centre for Transforming Maintenance through Data Science (grant number IC180100030).


## A. Appendix 1

In this appendix, we present a 0-1 programming model for the off-line version of the scheduling problem, where its results are compared with the approximate dynamic programming approach in the computational study. A version of this model with three objectives was introduced by Nezami et al. (2017) for the deterministic model of the scheduling problem. The following decision variables are used in this model:

Table 12: Comparing results of ADP, the IP, and the heuristic for C3

| | | ADP | | | | | IP | | | | | Heuristic | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Sample path | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) | Dropped jobs (%) | Overall obj. | $C_{max}$ | Energy cost | Time (sec) |
| 1 | 1 | 0.0 | 4838.71 | 80 | 23873.55 | 6.36 | - | - | - | - | - | 0.0 | 7479.6 | 119 | 36922.0 | 78.32 |
| | 2 | | | | | | - | - | - | - | - | 0.0 | 7507.9 | 118 | 37068.0 | 87.09 |
| | 3 | | | | | | - | - | - | - | - | 0.0 | 7548.2 | 129 | 37225.0 | 113.08 |
| | 4 | | | | | | - | - | - | - | - | 0.0 | 7482.4 | 95 | 37032.0 | 40.86 |
| | 5 | | | | | | - | - | - | - | - | 0.0 | 7516.2 | 132 | 37053.0 | 118.77 |
| Average | | | | | | | | | | | | **0.0** | **7506.86** | **118.6** | **37.060** | **87.62** |
| 2 | 1 | 0.0 | 4692.94 | 83 | 23132.69 | 5.98 | - | - | - | - | - | 0.0 | 6975.4 | 100 | 34477.0 | 42.18 |
| | 2 | | | | | | - | - | - | - | - | 0.0 | 7102.6 | 133 | 34981.0 | 108.94 |
| | 3 | | | | | | - | - | - | - | - | 0.0 | 7062.9 | 131 | 34791.0 | 99.03 |
| | 4 | | | | | | - | - | - | - | - | 0.0 | 7035.4 | 134 | 34641.0 | 104.26 |
| | 5 | | | | | | - | - | - | - | - | 0.0 | 7131.8 | 129 | 35143.0 | 88.23 |
| Average | | | | | | | | | | | | **0.0** | **7061.62** | **125.4** | **34806.6** | **88.53** |
| 3 | 1 | 0.0 | 4570.06 | 81 | 22526.31 | 6.17 | - | - | - | - | - | 0.0 | 7052.9 | 125 | 34765.0 | 76.45 |
| | 2 | | | | | | - | - | - | - | - | 0.0 | 7052.4 | 125 | 34762.0 | 84.13 |
| | 3 | | | | | | - | - | - | - | - | 0.0 | 7040.6 | 126 | 34699.0 | 100.86 |
| | 4 | | | | | | - | - | - | - | - | 0.0 | 7076.8 | 122 | 34896.0 | 78.18 |
| | 5 | | | | | | - | - | - | - | - | 0.0 | 7018.4 | 125 | 34592.0 | 86.24 |
| Average | | | | | | | | | | | | **0.0** | **7048.22** | **124.6** | **34742.8** | **85.17** |
| 4 | 1 | 0.0 | 4739.52 | 79 | 23381.62 | 6.13 | - | - | - | - | - | 0.0 | 7095.9 | 121 | 34996.0 | 88.21 |
| | 2 | | | | | | - | - | - | - | - | 0.0 | 7096.8 | 122 | 34996.0 | 72.64 |
| | 3 | | | | | | - | - | - | - | - | 0.0 | 7162.9 | 130 | 35295.0 | 114.04 |
| | 4 | | | | | | - | - | - | - | - | 0.0 | 7179.8 | 135 | 35359.0 | 126.60 |
| | 5 | | | | | | - | - | - | - | - | 0.0 | 7106.2 | 119 | 35055.0 | 62.62 |
| Average | | | | | | | | | | | | **0.0** | **7128.32** | **125.4** | **35140.2** | **92.82** |
| 5 | 1 | 0.0 | 4884.62 | 79 | 24107.09 | 6.19 | - | - | - | - | - | 0.0 | 7130.8 | 100 | 35254.0 | 53.33 |
| | 2 | | | | | | - | - | - | - | - | 0.0 | 7315.6 | 115 | 36118.0 | 80.22 |
| | 3 | | | | | | - | - | - | - | - | 0.0 | 7398.4 | 131 | 36468.0 | 128.26 |
| | 4 | | | | | | - | - | - | - | - | 0.0 | 7417.1 | 131 | 36562.0 | 122.82 |
| | 5 | | | | | | - | - | - | - | - | 0.0 | 7417.8 | 128 | 36577.0 | 99.18 |
| Average | | | | | | | | | | | | **0.0** | **7335.94** | **121** | **36195.8** | **96.76** |

- $x_{tj}^m = \begin{cases} 1 & \text{if machine } m \text{ starts processing on job } j \text{ in time } t, \\ 0 & \text{Otherwise} \end{cases}$
- $\omega_{tj}^m = \begin{cases} 1 & \text{if machine } m \text{ is processing job } j\text{in time } t, \\ 0 & \text{Otherwise} \end{cases}$
- $y_t^m = \begin{cases} 1 & \text{if machine } m \text{ is in idle mode in time } t, \\ 0 & \text{Otherwise} \end{cases}$
- $z_t^m = \begin{cases} 1 & \text{if machine } m \text{ is turned on in time slot } t, \\ 0 & \text{Otherwise.} \end{cases}$

The parameters are the same as in defined in the paper.

$$\min \quad \zeta_1 C_{\max} + \zeta_2 \sum_t E_t \Big( \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}:r_j\geq t} \mu_t^m \omega_{tj}^m + \sum_{m\in\mathcal{M}} \kappa_t^m y_t^m + \sum_{m\in\mathcal{M}} \lambda_t^m z_t^m \Big) \tag{33}$$

subject to

$$\sum_{m\in\mathcal{M}} \sum_{t=r_j}^{T-p_{mj}+1} x_{tj}^m \leq 1, \qquad \forall j \in \mathcal{J}, \tag{34}$$

$$\sum_{j\in\mathcal{J}:r_j\leq t} x_{tj}^m \leq 1, \qquad \forall m \in \mathcal{M}, \forall t, \tag{35}$$

$$\sum_{j\in\mathcal{J}:r_j\leq t} \omega_{tj}^m \leq 1, \qquad \forall m \in \mathcal{M}, \forall t, \tag{36}$$

$$p_{mj} x_{tj}^m \leq \sum_{\theta=t}^{t+p_{mj}-1} \omega_{\theta j}^m, \quad \forall m \in \mathcal{M}, \forall j \in \mathcal{J}, \forall t \in \{r_j, r_j+1, \ldots, T-p_{mj}+1\}, \tag{37}$$

$$\omega_{tj}^m = \sum_{\theta=r_j}^{t} x_{\theta j}^m, \qquad \forall m \in \mathcal{M}, \forall j \in \mathcal{J}, \forall t \in \{r_j, r_j+1, \ldots, r_j+p_{mj}-1\}, \tag{38}$$

$$\omega_{tj}^m = \sum_{\theta=t-p_{mj}+1}^{t} x_{\theta j}^m, \qquad \forall m \in \mathcal{M}, \forall j \in \mathcal{J}, \forall t \in \{r_j+p_{mj}, \ldots, T\}, \tag{39}$$

$$\sum_t z_t^m \leq 1, \qquad \forall m \in \mathcal{M}, \tag{40}$$

$$\sum_{j\in\mathcal{J}:r_j\geq t} x_{tj}^m - \sum_{\theta=1}^{t} z_\theta^m \leq 0, \qquad \forall m \in \mathcal{M}, \forall t, \tag{41}$$

$$\sum_{\theta=1}^{t} z_\theta^m + (1 - \sum_{j\in\mathcal{J}:r_j\leq t} \omega_{tj}^m) \leq 1 + y_t^m \qquad \forall m \in \mathcal{M}, \forall t, \tag{42}$$

$$1 - \sum_{j\in\mathcal{J}:r_j\leq t} \omega_{tj}^m \geq y_t^m \qquad \forall m \in \mathcal{M}, \forall t, \tag{43}$$

$$\sum_{\theta=1}^{t} z_{\theta}^{m} \geq y_{t}^{m} \qquad \forall m \in \mathcal{M}, \forall t, \tag{44}$$

$$\sum_{m \in \mathcal{M}} \sum_{t \in T} (t + p_{mj} - 1) x_{tj}^{m} \leq C_{\max} \qquad \forall j \in \mathcal{J}, \tag{45}$$

$$x_{tj}^{m}, \omega_{tj}^{m}, z_{t}^{m}, y_{t}^{m} \in \{0, 1\}, \; C_{\max} \geq 0. \tag{46}$$

The objective function (33) minimizes a linear combination of makespan and total energy cost. Constraint (34) ensures that jobs are assigned to exactly one time slot on all machines combined. Constraints (35)-(36) ensure that each machine processes at most one job at a time. Constraints (37)-(39) ensure the necessary processing times for each job without preemption. Constraint (40) limit the number of times a machine can be switched on. Constraint (41) ensure that a job can be processed by a working machine. Constraints (42)-(44) define the state of each machine during each time slot. Constraint (45) defines the completion time of each job and links it to the makespan, $C_{\max}$. Constraint (46) is the integrality constraint. It should be noted that the IP (33)-(46) extends the decision problem of the approximate dynamic programming to the entire planning horizon.

## B. Appendix 2: Paolucci-Anghinolfi-Tonelli algorithm

Presented below is the algorithm of Paolocci (Paolucci et al. 2017) that we used to compare the performance of our proposed method. First, we define the following notations:

- $h$: the iteration index,
- $H$: the maximum number of iteration,
- $SB$: the current best solution,
- $SR$: the current reference solution,
- $SC$: the candidate schedule,
- $\Gamma_k$: the list of jobs that are ready to be processed on machine $k$,
- $O$: number of oredered candidate job (an algorithm parameter),
- $\pi_{\min}$: a small positive number (an algorithm parameter),
- $q_0$: a fixed exploitation threshold,
- $f^{SB}$: the weighted sum of the objective function of the best schedule,
- $f^{SC}$: the weighted sum of the objective function of the candidate schedule,
- $T_h$: temperature at iteration $h$,
- $\chi$: the cooling factor.

The steps of algorithm are outlined below:

**Step 1.** Read all relevant data such as number of jobs and number of machines. Create a list of future events based on the jobs arrival and completion times. Sort this list in the chronological order of events. Set $h = 0$.

**Step 2.** Let $t$ be the current simulation time at which the next event to be processed occurs. At each event time, $t$, an existing job in the system should be assigned to one of the available machines.

**Step 2.1.** Consider all jobs ready to start and available machines at event time $t$. Machines are ordered randomly.

**Step 2.2.** Select machine $k$ from list of machines. Create a list of jobs that are ready to be processed on machine $k$, and denote it by $\Gamma_k$. From list $\Gamma_k$, choose job $\tau$ as follows:

**Step 2.2.1.** Jobs in $\Gamma_k$ is sorted according to their start time in the SR (ties are broken randomly). At the first iteration, SR = 0.

**Step 2.2.2.** Create list of $O$ ordered jobs, $C_k = (\tau_1, \ldots, \tau_O)$, from jobs in $\Gamma_k$. Assign the following selection probability to each job in $C_k$:

$$P_i = \frac{\frac{O-i}{O-1} + \pi_{\min}}{O\pi_{\min} + \sum_{j=1}^{O} \frac{O-j}{O-1}}, \qquad i = 1, \ldots, O. \tag{47}$$

**Step 2.2.3.** Using (47), select job $\tau_C \in C_k$.

**Step 2.2.4.** Generate a random number $q \sim U[0, 1]$, if $q < q_0$, assign job $\tau_1$ to machine $k$; otherwise, assign job $\tau_C$ to machine $k$.

**Step 2.2.5** Update machine and set $M = M \setminus \{k\}$ and event list. If $M \neq \emptyset$, go to Step 2.1; otherwise, go to Step 3.

**Step 3.** If the event list is empty, a complete schedule SC has been generated, go to Step 4; otherwise, go to Step 2.

**Step 4.** Evaluate the candidate schedule SC of Step 3 by the percentage deviation $\Delta Z(SC)$ with respect to $SB$ defined as follows

$$\Delta Z(SC) = \frac{f^{SB} - f^{SC}}{f^{SB}}. \tag{48}$$

If $\Delta Z(SC) > 0$, then set $SB = SC$, $SR = SC$, and $h = h + 1$, go to Step 6; otherwise, go to step 5.

**Step 5.** Use a simulated annealing acceptance rule. Set $SR = SC$ if random number $p \sim U[0, 1] < e^{\left(\frac{\Delta Z(SC)}{T_h}\right)}$. Set $T_{h+1} = \alpha T_h$ and $h = h + 1$. Go to Step 2.

**Step 6.** If $h \leq H$, go to Step 2; otherwise STOP.

## References


Abedi, M., Chiong, R., Noman, N., and Zhang, R. (2020). A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines. *Expert Systems with Applications*, page 113348.

Aghelinejad, M., Ouazene, Y., and Yalaoui, A. (2018). Production scheduling optimisation with machine state and time-dependent energy costs. *International Journal of Production Research*, 56(16):5558–5575.

Al-Kanj, L., Nascimento, J., and Powell, W. B. (2020). Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. *European Journal of Operational Research*, 284(3):1088–1106.

Albers, S. (2009). Online scheduling.

Alkaabneh, F., Diabat, A., and Gao, H. O. (2020). Unified framework for efficient, effective, and fair resource allocation by food banks: Approximate dynamic programming approach. *Omega*, page 102300.

Anghinolfi, D., Paolucci, M., and Ronco, R. (2020). A bi-objective heuristic approach for green identical parallel machine scheduling. *European Journal of Operational Research*.

Cao, J., Pan, R., Xia, X., Shao, X., and Wang, X. (2020). An efficient scheduling approach for an iron-steel plant equipped with self-generation equipment under time-of-use electricity tariffs. *Swarm and Evolutionary Computation*, page 100764.

Che, A., Zhang, S., and Wu, X. (2017). Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, 156:688–697.

Chen, J.-f., Wang, L., and Peng, Z.-p. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation*, 50:100557.

Cheng, J., Chu, F., Chu, C., and Xia, W. (2016). Bi-objective optimization of single-machine batch scheduling under time-of-use electricity prices. *RAIRO-Operations Research*, 50(4-5):715–732.

Cheng, J., Chu, F., Liu, M., Wu, P., and Xia, W. (2017a). Bi-criteria single-machine batch scheduling with machine on/off switching under time-of-use tariffs. *Computers & Industrial Engineering*, 112:721–734.

Cheng, J., Chu, F., and Zhou, M. (2017b). An improved model for parallel machine scheduling under time-of-use electricity price. *IEEE Transactions on Automation Science and Engineering*, 15(2):896–899.

Cohon, J. (1978). *MULTIOBJECTIVE PROGRAMMING AND PLANNING*.

Dai, M., Tang, D., Giret, A., and Salido, M. A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, 59:143–157.

Ding, J.-Y., Song, S., Zhang, R., Chiong, R., and Wu, C. (2015). Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, 13(2):1138–1154.

Fang, K., Uhan, N. A., Zhao, F., and Sutherland, J. W. (2016). Scheduling on a single machine under time-of-use electricity tariffs. *Annals of Operations Research*, 238(1-2):199–227.

Gahm, C., Denz, F., Dirr, M., and Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research*, 248(3):744–757.

Gajic, D., Hadera, H., Onofri, L., Harjunkoski, I., and Di Gennaro, S. (2017). Implementation of an integrated production and electricity optimization system in melt shop. *Journal of Cleaner Production*, 155:39–46.

Giglio, D., Paolucci, M., and Roshani, A. (2017). Integrated lot sizing and energy-efficient job shop scheduling problem in manufacturing/remanufacturing systems. *Journal of Cleaner Production*, 148:624–641.

Giret, A., Trentesaux, D., and Prabhu, V. (2015). Sustainability in manufacturing operations scheduling: A state of the art review. *Journal of Manufacturing Systems*, 37:126–140.

Godfrey, G. A. and Powell, W. B. (2002). An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science*, 36(1):21–39.

Golpîra, H., Khan, S. A. R., and Zhang, Y. (2018). Robust smart energy efficient production planning for a general job-shop manufacturing system under combined demand and supply uncertainty in the presence of grid-connected microgrid. *Journal of Cleaner Production*, 202:649–665.

Gong, G., Chiong, R., Deng, Q., Han, W., Zhang, L., Lin, W., and Li, K. (2020). Energy-efficient flexible flow shop scheduling with worker flexibility. *Expert Systems with Applications*, 141:112902.

Hadera, H., Harjunkoski, I., Sand, G., Grossmann, I. E., and Engell, S. (2015). Optimization of steel production scheduling with complex time-sensitive electricity cost. *Computers & Chemical Engineering*, 76:117–136.

Heydar, M., O'Reilly, M. M., Trainer, E., Fackrell, M., Taylor, P. G., and Tirdad, A. (2021). A stochastic model for the patient-to-bed assignment problem with random arrivals and departures. *Annals of Operations Research*.

Hulshof, P. J., Mes, M. R., Boucherie, R. J., and Hans, E. W. (2016). Patient admission planning using approximate dynamic programming. *Flexible Services and Manufacturing Journal*, 28(1-2):30–61.

Jenkins, P. R., Robbins, M. J., and Lunday, B. J. (2021). Approximate dynamic programming for the military aeromedical evacuation dispatching, preemption-rerouting, and redeployment problem. *European Journal of Operational Research*, 290(1):132–143.

Ji, M., Wang, J.-Y., and Lee, W.-C. (2013). Minimizing resource consumption on uniform parallel machines with a bound on makespan. *Computers & Operations Research*, 40(12):2970–2974.

Jiang, E.-d. and Wang, L. (2019). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *International Journal of Production Research*, 57(6):1756–1771.

Koch, S. and Klein, R. (2020). Route-based approximate dynamic programming for dynamic pricing in attended home delivery. *European Journal of Operational Research*, 287(2):633–652.

Lei, D., Zheng, Y., and Guo, X. (2017). A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. *International Journal of Production Research*, 55(11):3126–3140.

Leung, J. Y.-T., Lee, K., and Pinedo, M. L. (2012). Bi-criteria scheduling with machine assignment costs. *International Journal of Production Economics*, 139(1):321–329.

Li, M., Lei, D., and Cai, J. (2019). Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. *Swarm and Evolutionary Computation*, 49:34–43.

Liu, M., Yang, X., Chu, F., Zhang, J., and Chu, C. (2020). Energy-oriented bi-objective optimization for the tempered glass scheduling. *Omega*, 90:101995.

Liu, Y., Dong, H., Lohse, N., and Petrovic, S. (2016). A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance. *International Journal of Production Economics*, 179:259–272.

Lu, C., Gao, L., Li, X., Pan, Q., and Wang, Q. (2017). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *Journal of Cleaner Production*, 144:228–238.

Luo, H., Du, B., Huang, G. Q., Chen, H., and Li, X. (2013). Hybrid flow shop scheduling considering machine electricity consumption cost. *International Journal of Production Economics*, 146(2):423–439.

Mansouri, S. A. and Aktas, E. (2016). Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem. *Journal of the Operational Research Society*, 67(11):1382–1394.

Mansouri, S. A., Aktas, E., and Besikci, U. (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *European Journal of Operational Research*, 248(3):772–788.

May, G., Stahl, B., Taisch, M., and Prabhu, V. (2015). Multi-objective genetic algorithm for energy-efficient job shop scheduling. *International Journal of Production Research*, 53(23):7071–7089.

Meng, L., Zhang, C., Shao, X., and Ren, Y. (2019). Milp models for energy-aware flexible job shop scheduling problem. *Journal of Cleaner Production*, 210:710–723.

Mes, M. R. and Rivera, A. P. (2017). Approximate dynamic programming by practical examples. In *Markov Decision Processes in Practice*, pages 63–101. Springer.

Mokhtari, H. and Hasani, A. (2017). An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers & Chemical Engineering*, 104:339–352.

Moon, J.-Y., Shin, K., and Park, J. (2013). Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *The International Journal of Advanced Manufacturing Technology*, 68(1-4):523–535.

Mouzon, G., Yildirim, M. B., and Twomey, J. (2007). Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research*, 45(18-19):4247–4271.

Nezami, F. G., Heydar, M., and Berretta, R. (2017). Optimizing production schedule with energy consumption and demand charges in parallel machine setting. In *8th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2017)*, pages 133–143.

Nouiri, M., Bekrar, A., and Trentesaux, D. (2018). Towards energy efficient scheduling and rescheduling for dynamic flexible job shop problem. *IFAC-PapersOnLine*, 51(11):1275–1280.

Paolucci, M., Anghinolfi, D., and Tonelli, F. (2017). Facing energy-aware scheduling: a multiobjective extension of a scheduling support system for improving energy efficiency in a moulding industry. *Soft Computing*, 21(13):3687–3698.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons.

Powell, W. B. (2008). Approximate dynamic programming: lessons from the field. In *2008 Winter Simulation Conference*, pages 205–214. IEEE.

Powell, W. B. (2016). Perspectives of approximate dynamic programming. *Annals of Operations Research*, 241(1-2):319–356.

Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821.

Ronconi, D. P. and Powell, W. B. (2010). Minimizing total tardiness in a stochastic single machine scheduling problem using approximate dynamic programming. *Journal of Scheduling*, 13(6):597–607.

Rubaiee, S. and Yildirim, M. B. (2019). An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling. *Computers & Industrial Engineering*, 127:240–252.

Safarzadeh, H. and Niaki, S. T. A. (2019). Bi-objective green scheduling in uniform parallel machine environments. *Journal of Cleaner Production*, 217:559–572.

Salido, M. A., Escamilla, J., Giret, A., and Barber, F. (2016). A genetic algorithm for energy-efficiency in job-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 85(5-8):1303–1314.

Saure, A., Patrick, J., Tyldesley, S., and Puterman, M. L. (2012). Dynamic multi-appointment patient scheduling for radiation therapy. *European Journal of Operational Research*, 223(2):573–584.

Shrouf, F., Ordieres-Meré, J., García-Sánchez, A., and Ortega-Mier, M. (2014). Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, 67:197–207.

Silva, T. A. and de Souza, M. C. (2020). Surgical scheduling under uncertainty by approximate dynamic programming. *Omega*, 95:102066.

Tan, Y. and Liu, S. (2014). Models and optimisation approaches for scheduling steelmaking–refining–continuous casting production under variable electricity price. *International Journal of Production Research*, 52(4):1032–1049.

Wan, G. and Qi, X. (2010). Scheduling with variable time slot costs. *Naval Research Logistics (NRL)*, 57(2):159–171.

Wang, L., Peng, Z.-p., et al. (2020a). Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition. *Swarm and Evolutionary Computation*, 58:100745.

Wang, S., Liu, M., Chu, F., and Chu, C. (2016). Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration. *Journal of Cleaner Production*, 137:1205–1215.

Wang, S., Wang, X., Chu, F., and Yu, J. (2020b). An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *International Journal of Production Research*, 58(8):2283–2314.

Yang, Y., Chen, W., Wei, L., and Chen, X. (2018). Robust optimization for integrated scrap steel charge considering uncertain metal elements concentrations and production scheduling under time-of-use electricity tariff. *Journal of Cleaner Production*, 176:800–812.

Yin, L., Li, X., Gao, L., Lu, C., and Zhang, Z. (2017). Energy-efficient job shop scheduling problem with variable spindle speed using a novel multi-objective algorithm. *Advances in Mechanical Engineering*, 9(4):1687814017695959.

Yuan, Y. and Tang, L. (2017). Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse. *European Journal of Operational Research*, 262(2):424–437.

Zhang, H., Zhao, F., Fang, K., and Sutherland, J. W. (2014). Energy-conscious flow shop scheduling under time-of-use electricity tariffs. *CIRP Annals*, 63(1):37–40.

Zhang, R. and Chiong, R. (2016). Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production*, 112:3361–3375.

Zhang, Y., Wang, J., and Liu, Y. (2017). Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact. *Journal of Cleaner Production*, 167:665–679.

Zheng, X., Zhou, S., Xu, R., and Chen, H. (2020). Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimisation algorithm. *International Journal of Production Research*, 58(13):4103–4120.