Bus Inf Syst Eng 61(3):299–309 (2019)
https://doi.org/10.1007/s12599-019-00590-7

RESEARCH PAPER

An Adaptive Scheduling Algorithm for Dynamic Jobs for Dealing with the Flexible Job Shop Scheduling Problem

Zhengcai Cao · Lijie Zhou · Biao Hu · Chengran Lin

Received: 1 July 2018 / Accepted: 17 December 2018 / Published online: 6 March 2019

Abstract Modern manufacturing systems build on an effective scheduling scheme that makes full use of the system resource to increase the production, in which an important aspect is how to minimize the makespan for a certain production task (i.e., the time that elapses from the start of work to the end) in order to achieve the economic profit. This can be a difficult problem, especially when the production flow is complicated and production tasks may suddenly change. As a consequence, exact approaches are not able to schedule the production in a short time. In this paper, an adaptive scheduling algorithm is proposed to address the makespan minimization in the dynamic job shop scheduling problem. Instead of a linear order, the directed acyclic graph is used to represent the complex precedence constraints among operations in jobs. Inspired by the heterogeneous earliest finish time (HEFT) algorithm, the adaptive scheduling algorithm can make some fast adaptations on the fly to accommodate new jobs which continuously arrive in a manufacturing system. The performance of the proposed adaptive HEFT algorithm is compared with other state-of-the-art algorithms and further heuristic methods for minimizing the makespan. Extensive experimental results demonstrate the high efficiency of the proposed approach.

Accepted after two revisions by the editors of the special issue.

Z. Cao · L. Zhou · B. Hu (✉) · C. Lin
College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China
e-mail: hubiao@mail.buct.edu.cn

Z. Cao
e-mail: giftczc@163.com

L. Zhou
e-mail: zlj_rb@126.com

C. Lin
e-mail: chranlin@163.com

Keywords Makespan · Flexible job shop · Adaptive scheduling · HEFT

1 Introduction

In order to increase market competitiveness, manufacturing enterprises are trying to find effective ways to reduce their product cost while maintaining high quality (Fink et al. 2014; Schryen et al. 2015). A very important aspect in this domain is to optimize the manufacturing production schedule by taking the manufacturing system information into consideration (Ulmer et al. 2017; Hoffmann et al. 2017). In Wang et al. (2014), a discrete scheduling system for enterprises is built to minimize a discounted expense, in which an information system called collaborative manufacturing execution system (cMES) plays an important role. Currently, most companies obtain real-time production information collected by manufacturing execution systems, and will use it in scheduling procedures (Cao et al. 2014). Zhou et al. (2015) and Raileanu et al. (2014) propose two cMES-based job-shop scheduling algorithms for the steel-making industry and semiconductor manufacture, respectively. Except for cMES, a progressive information system, named Cyber-Physical Production Systems (CPPS), is utilized in many manufacture production lines. In Varvara (2016), a multi-agent systems for the semiconductor final testing secluding problem is designed on the basis of the information from CPPS. Xie et al. (2017) propose an an automotive cyber-physical scheduling technique that is system-based as well as adaptive and dynamic for integrated automotive architecture. In general, the production system is a process-aware information system as it tries to manage and execute operational processes which involves people, production resources and

flows, and information sources on the basis of process models (Ma 2010; Kriglstein et al. 2016).

An illustrative workflow of scheduling in manufacturing enterprises is presented in Fig. 1. The manufacturing factory will obtain production tasks from the received order. Based on the manufacturing information system and key features of the production line, a production schedule should be planned to maximize the production effectiveness so that enterprises can gain as much production benefit as possible. The production effectiveness can be represented by an objective function. Under the constraint of manufacturing resources (people, machines, information sources, production workflows etc.), a classic objective is to minimize the makespan for a certain production task (i.e., the time that elapses from the start of work to the end) as in this way enterprises can reduce the cost of labor and electricity while achieving a high quality of products. In this paper, we study the problem of minimizing the production makespan in manufacturing enterprises, where it is also a job-shop problem.

The makespan minimization has been studied in a variety of job shop problems, ranging from the classic job shop problem to the extended flexible job shop one. The classic job shop problem is a well-known NP-hard problem to schedule n jobs in an environment with m identical machines, where a job is composed by linear precedent operations (Garey et al. 1976). A generalization of the job shop problem is the flexible job shop scheduling problem (FJS) in which there may be several heterogeneous machines that are capable of handling job operations (Brucker and Schlie 1990). Heterogeneous here means that operations may need different processing times in different machines. The extended flexible job shop problem is an extension of the FJS problem where instead of a linear order, an arbitrary directed acyclic graph (DAG) needs to be used to model the precedence between the operations (Birgin et al. 2015; Borenstein 2000). A typical example of jobs that need to be modeled as DAG comes from the printing and boarding industry, where jobs like printing and prepress need to perform a set of operations whose inside structures are not linear (Zeng et al. 2010).

To make use of DAG jobs in practical applications, a few approaches have been proposed to address their scheduling optimization problems in production. In Vilcota and Billautb (2008), against the backdrop of the printing and boarding industry, a tabu search and a genetic algorithm are applied to find an approximation of the Pareto frontier for the makespan and the maximum lateness criteria. In Lee et al. (2012), a heuristic approach is proposed to solve FJS with 'AND/OR' precedence constraints in the job operations. Besides, another genetic and tabu search algorithm has been developed to produce a legal and feasible solution for a schedule builder. A scheduling problem of producing a variety of manufactured glass objects in a glass factory has been addressed in Alvarez-Valdesabacc (2005). This problem also belongs to FJS problems with some special characteristics, such as no-wait constraints. A heuristic list scheduling approach and its beam search extension have been proposed in Birgin et al. (2015) to minimize the makespan in extended FJS problems.

All aforementioned scheduling approaches are proposed to handle the static job shop problem, which cannot be applied to dynamic environments where jobs may arrive in stochastic ways. From the perspective of smart production,

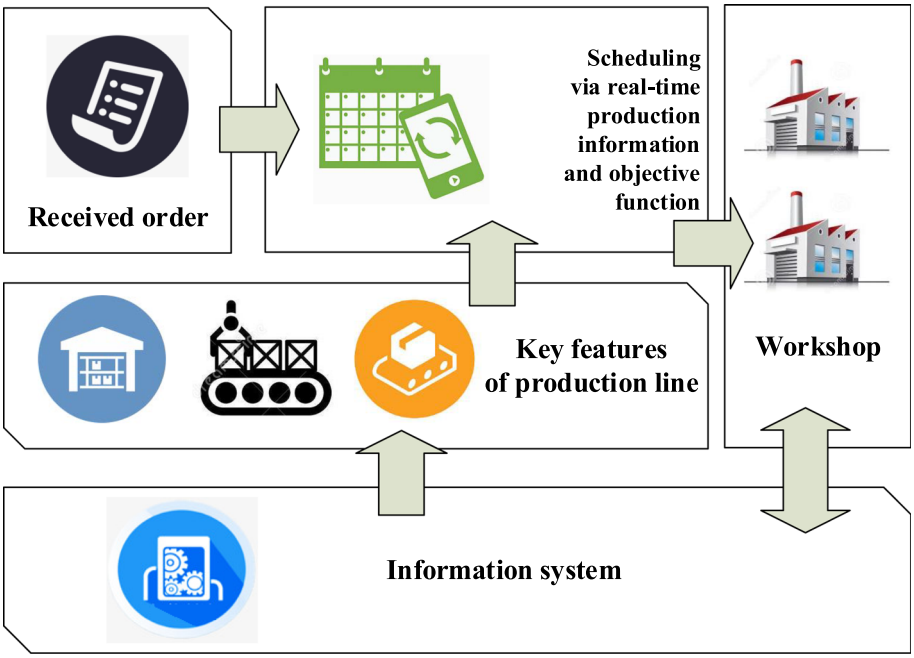

Fig. 1 An illustrative workflow of scheduling in manufacturing enterprises

it is important to quickly make an adjustment to the current schedule when production jobs are suddenly changed. This is however a non-trivial problem. On the one hand, every time a new job arrives, the previous job shop schedule needs to be updated because the job shop problem has changed. This involves handling the processed, processing, and unprocessed operations. On the other hand, new jobs may arrive unpredictably, which makes it impossible to prepare the schedule offline. An adaptive scheduling algorithm needs to be developed that can fast respond to new jobs and find a new schedule in a speedy way. In dynamic manufacturing environments, intelligent algorithms or meta-heuristics are often impracticable because recursive computations that widely exist in those algorithms are time-consuming and cannot provide a reschedule plan fast.

In this paper, we aim to propose an efficient algorithm that can reschedule the production for stochastic arrival jobs in a short time. We also incorporate two important manufacturing aspects into the scheduling. One aspect is that a machine needs some setup time to process operations from different jobs because different jobs may require machines to work differently. For example, to test a semiconductor product, a tester, a handler and an enabler are simultaneously needed. Different jobs may match different resource combinations. To assemble and calibrate the machine for the incoming new job type, a setup time is usually required for the setup activities in the above problem (Cao et al. 2017). The other aspect is that in order to ensure the positioning and functioning accuracy of some key components, it is often necessary to link or connect two or more parts together. In this paper, we call this situation the matching operation. An example is that template process and electrode process need to be processed in combination (Gan and Lee 2002).

This paper presents an adaptive scheduling algorithm named A-HEFT to minimize the makespan of dynamically changing jobs by taking the setup time and mating operation into account. This algorithm is inspired by the Heterogeneous Earliest Finish Time (HEFT) algorithm that schedules applications modeled as DAGs based on the earliest finish time principle in a heterogeneous distributed system (Topcuouglu et al. 2002). There are two phases in the algorithm A-HEFT. In the first phase, operations of each job are prioritized based on the upward rank value. In the second phase, based on the round-robin policy, operations are successively dispatched to a machine according to the principle of the earliest finish time. For handling the jobs newly arriving, all unprocessed operations will be re-dispatched again, while processing operations remain unchanged. Experimental results demonstrate that A-HEFT can achieve shorter makespan than any other state-of-the-art scheduling approaches.

The rest of this paper is organized as follows. In Sect. 2 we present the dynamic problem with some relevant notations. In Sect. 3 we construct the scheduling model and elaborate the adaptive scheduling algorithm, i.e., A-HEFT. In Sect. 4 we evaluate the proposed algorithm in comparison with some other state-of-the-art algorithms. This paper is concluded in Sect. 5.

2 Problem Description

In this section, we present the DAG model in the dynamic scheduling environment and formulate the problem of minimizing the scheduling makespan.

2.1 Job Operation Precedence Model

A job may consist of two or two more operations whose relationships can be represented as a DAG model. Generally a DAG model is able to represent sequential operations that can only be processed in a given sequential order and parallel operations that can be processed simultaneously in different machines. Besides, the DAG model is also able to represent the matching operation where multiple operations from different jobs need to be processed together in one machine. Depending on the operation constraint type, the route in a DAG model may fork or merge. A typical type of a DAG job is presented in the example of Fig. 2. We use $O_{i,j}$ to represent the j-th operation of the i-th job. As shown in Fig. 2a, DAG_1 is composed of two jobs J_1 and J_2, where the last operation $O_{1,3}$ and $O_{2,3}$ of each job need to be processed in combination. In DAG_2, there are also two jobs whose operations, i.e., $O_{3,3}$ and $O_{4,2}$, need to be processed in combination (Fig. 2).

To generalize the DAG form, two special operations are defined in every DAG, which are the entry operation and the exit operation. The entry operation is the first operation that needs to be processed ahead of all the other operations. The exit operation is the last operation that needs to be processed after completing all the other operations. For instance, the matching operation of $O_{1,3}$ and $O_{2,3}$ is the exit operation of DAG_1. Since there is no entry operation in DAG_1, a virtual entry operation is created. This virtual entry operation is a dummy operation that does not consume any production resources and will not influence the scheduling performance. It is created to assist the priority ranking in our proposed algorithm. Analogously, a virtual entry operation and a virtual exit operation are created in DAG_2. In this way, all DAG jobs will start to be processed from the entry operation and end at the exit operation. Furthermore, we take the setup time that a machine may need to incorporate an operation from a different job into

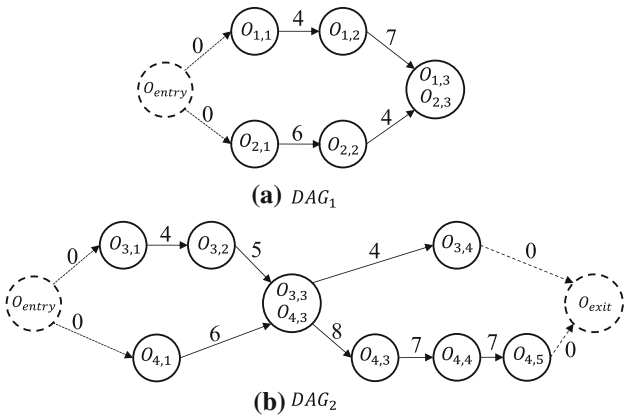

Fig. 2 Example of the precedence constraints in jobs modeled as DAG

the DAG model. As shown in DAG_1 and DAG_2, the edge number represents such setup times.

2.2 Notations and Problem Formulation

For an easier description of the extended FJS problem, the indices, parameters and decision variables used in this paper are defined as shown in Table 1.

The scheduling problem in this paper can be described as follows. Suppose the manufacturing system needs to be rescheduled at time t ($t \geq 0$). There are n jobs that need to be processed on m machines. Each job has one or more operations whose relationship can be modeled as DAG. Before the time instance of t, job operations have been either processed or assigned to a specific machine to be processed according to the previous schedule. If operations have been processed or are being processed at time t, their schedule cannot be changed because all machines are non-preemptive. Hence, only those operations which are unprocessed can be re-assigned to a different machine with a different start time. The objective of rescheduling is to minimize the makespan of completing all n jobs. The makespan is denoted by C_{max} and the objective function is formulated as below:

Objective function:

$$\text{Minimize } C_{\max} \tag{1}$$

subject to the following constraints.

1. Start time constraint:

$$st_{i,j} \geq 0, \tag{2}$$

where any operation should start to be processed after 0 time instance.

2. Job allocation constraints:

Table 1 General notations for describing the extended FJS problem

Notation	Definition
A. Indices	
J_i	Index of jobs, $i = 1, 2, \ldots, n$
$O_{i,j}$	Index of operations, $i = 1, \ldots, n$, $j = 1, \ldots, o_i$
B. Parameters	
n	Total number of jobs
m	Number of available machines
o_i	Number of operations in job J_i
$\Omega_{O_{i,j}}$	The set of machines that can process the operation $O_{i,j}$
C_{max}	The makespan
$P_{i,j,k}$	The processing time of $O_{i,j}$ on the machine k
$\tau_{[i,j;i',j']}$	The setup time between $O_{i,j}$ and $O_{i',j'}$. Note that $\tau_{[i,j;i',j']}$ is a sequence independent setup time, which depends on operations and jobs
C. Decision variables	
$\varepsilon_{i,j,k}$	$\varepsilon_{i,j,k} = 1$ if operation $O_{i,j}$ is assigned to be processed on machine k. Otherwise, $\varepsilon_{i,j,k} = 0$, $k \in \Omega_{O_{i,j}}$
$st_{i,j}$	The start time of processing $O_{i,j}$, $st_{i,j} \geq 0$
$\lambda_{[i,j;i',j']}$	If $O_{i,j}$ is processed previous to $O_{i',j'}$ on the same machine, $\lambda_{[i,j;i',j']} = 1$. Otherwise, $\lambda_{[i,j;i',j']} = 0$

$$\sum_{k \in \Omega_{O_{i,j}}} \varepsilon_{i,j,k} = 1, \forall t, 0 \leq t \leq C_{max}. \tag{3}$$

Note that $\varepsilon_{i,j,k}$ is fixed for those operations $st_{i,j} \leq t$. The optimization algorithm only needs to optimize the schedule on operations whose start time are greater than t.

3. Actual processing time:

$$p'_{i,j} = \sum_{k \in \Omega_{O_{i,j}}} \varepsilon_{i,j,k} p_{i,j,k}, \quad \forall i \leq n, j \leq o_i, \tag{4}$$

where $p'_{i,j}$ defines the actual processing time of each operation $O_{i,j}$ and is decided by the assigned machine.

4. The Makespan calculation:

$$C_{\max} = \max\{st_{i,j} + p'_{i,j}, \forall i \leq n, j \leq o_i\}, \tag{5}$$

where the makespan is the last moment of finishing all operations.

5. Operation precedence constraint:

$$st_{i,j} + p'_{i,j} + \tau_{[i,j;i',j']} \leq st_{i',j'} + p'_{i',j'}, \tag{6}$$

where $O_{i,j}$ is a precedent operation of $O_{i',j'}$.

6. Machine capacity constraint:

Table 2 Parameters and upward rank value of DAG_1

Machines	Job operations				
	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{1,3}$ and $O_{2,3}$
M_1	3	×	6	×	×
M_2	4	×	3	×	×
M_3	5	×	5	×	×
M_4	×	4	×	7	5
M_5	×	×	×	3	3
$Rank_u$	23	15	24	13	4
Select priority	2	3	1	4	5

$$\begin{aligned} &\lambda_{[i,j;i',j']} + \lambda_{[i,j;i',j']} \geq \varepsilon_{i,j,k} + \varepsilon_{i',j',k} - 1,\ k \in \Omega_{O_{i,j}} \cap \Omega_{O_{i',j'}} \\ &st_{i,j} + p'_{i,j} + \tau_{[i,j;i',j']} - (1 - \lambda_{[i,j;i',j']}) \cdot L \leq st_{i',j'} \end{aligned}, \tag{7}$$

where $O_{i,j}$ and $O_{i',j'}$ are two operations that are assigned to the same machine and L is a sufficiently large positive constant. Based on Birgin et al. (2015), the above constraint provides that both operations cannot be processed at the same time and determine which one is processed first.

3 The Adaptive Scheduling Algorithm

In this section, we first introduce the concept of the upward rank value $rank_u$ in the heterogeneous earliest finish time algorithm. Next, we present the framework of the adaptive scheduling algorithm A-HEFT to handle the stochastic arrival jobs in the extended flexible job shop problem, followed by an example to explain our scheduling approach.

3.1 Upward Rank Value

The operations are selected one by on by a machine. The selecting rules are very important to minimize the makespan. Here we present it using the upward rank value to decide the operation assignment order. The upward rank value of a task is defined as

$$rank_u(O_{i,j}) = \lceil \overline{w}_{i,j} \rceil + \max_{O_{i',j'} \in succ(O_{i,j})} \{\tau_{i,j;i',j'} + rank_u(O_{i',j'})\}, \tag{8}$$

where $\lceil \overline{w}_{i,j} \rceil$ rounds up the average processing time of task $O_{i,j}$ on available machines $\Omega_{O_{i,j}}$; $succ(O_{i,j})$ represents the successor operations of operation $O_{i,j}$. Operation selecting priorities are ordered based on the descending order of $rank_u$. The job operation parameters of DAG_1 and DAG_2 in Sect. 2.1 are presented in Tables 2 and 3, where the symbol × denotes that the machine is not capable of processing this operation. Based on Eq. 8, the upward rank values of job operations are presented in the $Rank_u$ row of Tables 2 and 3. The select priority of operations are also presented with a smaller number representing a higher priority. Specifically, in DAG_1, $O_{2,1}$ will be first selected to be assigned and the matching operation is the last one. Based on the upward rank value, the operation precedence constraint can be strictly satisfied and also perform well in the operation scheduling.

3.2 Dynamic Scheduling Framework

We propose an adaptive scheduling algorithm named A-HEFT to minimize the makespan of dynamic jobs by taking the setup time and matching operation into account. Multiple jobs represented as a DAG set $\mathbf{D} = \{D_1, D_2, \ldots, D_d\}$ will be processed on a heterogeneous machine set $\mathbf{M} = \{M_1, M_2, \ldots, M_m\}$, where d is the DAG set size. A dynamic scheduling framework is proposed to reschedule the production jobs as presented in Fig. 3. There are three types of queue: operation priority queue, common buffer queue, and operation collocation queue. The circles colored dark represent matching operation in the job pool. The dynamic scheduling framework has two key points:

Table 3 Parameters and upward rank value of DAG_2

Machines	Job operations							
	$O_{3,1}$	$O_{3,2}$	$O_{4,1}$	$O_{3,3}$ and $O_{4,2}$	$O_{3,4}$	$O_{4,3}$	$O_{4,4}$	$O_{4,5}$
M_1	5	7	2	×	×	6	6	×
M_2	4	6	2	3	×	6	5	×
M_3	3	5	2	×	3	4	4	×
M_4	×	×	×	×	×	×	×	5
M_5	×	×	×	×	×	×	×	×
$Rank_u$	60	52	49	41	3	30	17	5
Select priority	1	2	3	4	8	5	6	7

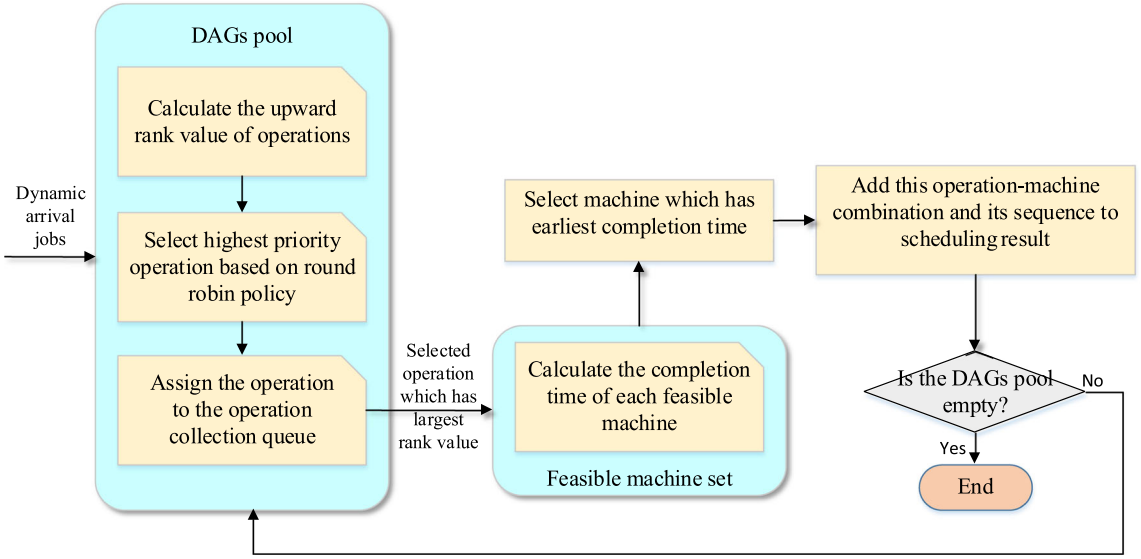

Fig. 3 The dynamic scheduling framework

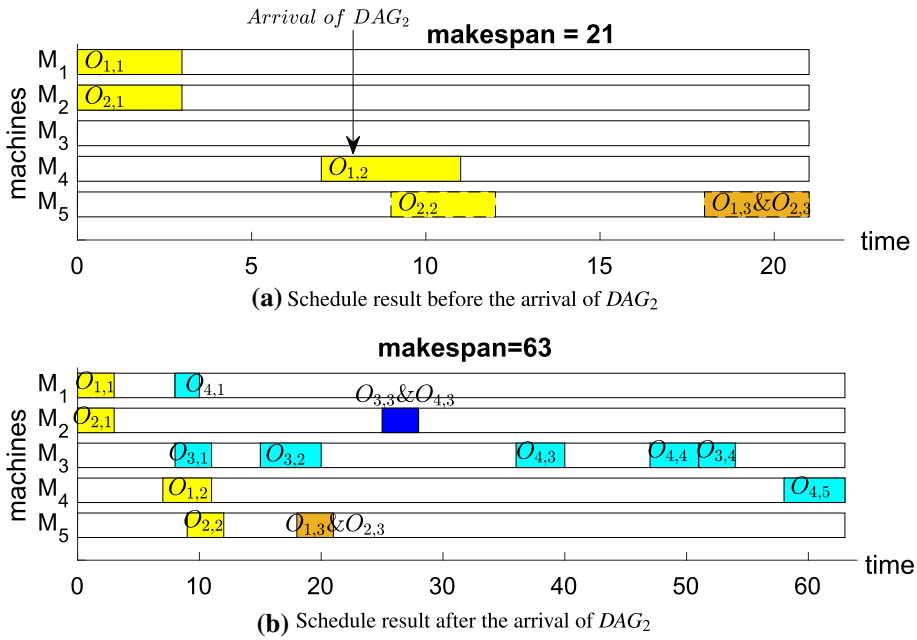

Fig. 4 Gantt chart of scheduling DAG jobs

1. The first key point is that new jobs may arrive unpredictably. The previous job shop schedule needs to be updated timely. It is actually a vital problem to schedule the stochastic jobs.
2. The second key point is that a machine needs some setup time to process operations from different jobs.

3.3 A-HEFT Scheduling Algorithm

Inspired by the heterogeneous earliest finishing time (-HEFT) algorithm that schedules DAGs based on the earliest finish time principle in a heterogeneous distributed system, we propose an adaptive scheduling algorithm called A-HEFT to reschedule job operations when necessary. The A-HEFT scheduling algorithm includes six steps as follows and its pseudo-code is presented in Algorithm 1:

Step(1) First of all, the current operations of all the jobs arrive at the pool of DAGs.

Step(2) Calculate the upward rank value $rank_u(O_{i,j})$ of every operation and put every operation in the operation priority queue according to the descending order of $rank_u(O_{i,j})$. The operation priority queue of a DAG D_i is denoted as D_{i_OPQ}.

Step(3) From the operation priority queue of each DAG, the highest priority operation is selected based on the round robin policy and is transferred to the common buffer queue. The round robin policy indicates that every DAG will release only one operation in one round. When the common buffer queue is empty again, the next round of operation release will start.

Step(4) In this step, operations will be successively selected from the common buffer queue and assigned to the operation collection queue. Every machine has its own operation collection queue and operations stored in the operation collection queue will be successively processed by this machine. The operation collection queue of a machine k is denoted as M_{k_OCQ}. In the common buffer queue, an operation with maximum $rank_u(O_{i,j})$ will be selected and assigned to the machine that can finish its process at the earliest moment compared to all other available machines in $\Omega_{O_{i,j}}$. Note that each operation is first allocated to the operation collection queue rather than the machine itself. Only when the start time of the operation is equal to the current time instant, is the operation assigned to the machine to be processed. Return to the Step(3) until no operation is left.

Step(5) Schedule all operations collected in the operation collection queue of each machine.

Step(6) When the new jobs arrive, all unprocessed operations which are waiting to be scheduled will be cancelled and re-dispatched, while processing operations will be unchanged. The above steps are repeated until all jobs are accomplished.

3.4 The Simple Example of A-HEFT Scheduling Algorithm

Based on the above steps, an example of scheduling DAG_1 and DAG_2 in Sect. 2.1 is given. Suppose DAG_1 arrives at $t = 0$ and DAG_2 arrives at $t = 8$. Based on Algorithm 1, we can obtain a schedule as presented in Fig. 4a. At time $t = 8$, DAG_2 arrives and triggers a reschedule. At the arrival time of DAG_2, the operation $O_{2,2}$ and matching operation $O_{1,3}\&O_{2,3}$ (marked out as dash line in Fig. 4a) will be removed from the schedule and return back to the DAGs pool. Then the removed operations from DAG_1 will be rescheduled together with operations from DAG_2 based on Algorithm 1. The reschedule result is shown in Fig. 4b.

4 Experimental Results Anlysis

Job shop scheduling algorithms can be classified into two main categories: static scheduling algorithms and dynamic scheduling algorithms. Our proposed scheduling approach can either schedule static jobs or dynamic jobs. Now two experiments are conducted to evaluate the performance of our proposed scheduling algorithm against some existing approaches. The proposed algorithm is coded in Matlab and run on a Core i5 2.7 GHz PC with 4 GB memory.

Algorithm 1 The A-HEFT Scheduling Algorithm

Require: A machine set $\mathbf{M} = \{M_1, M_2, \cdots, M_m\}$ and a DAG set $\mathbf{D} = \{D_1, D_2, \cdots, D_d\}$;
Ensure: Schedule results with minimum makespan
for $(i \leftarrow 1; i \leq d; i++)$ **do**
  The current operations of all the jobs are add to the job pool, and calculate $rank_u(O_{i,j})$;
  Operation in each DAG is dispatched to the operation priority queue based on the descending order of upward rank value, $D_{i\_OPQ}.add()$;
end for//Step(1)and Step(2)
while (jobs to be scheduled in DAG_m exist) **do**
  for $(i \leftarrow 1; i \leq d; i++)$ **do**
    $O_{i,j} \leftarrow D_{i\_OPQ}.out()$;
    Put $O_{i,j}$ to the common ready queue;
  end for//Step(3)
  while (Common Buffer Queue is not empty **do**
    From the common buffer queue, select the operation with highest upward rank value;
    From all available machines, a machine M_k is chosen that process it at the earliest moment and this operation is put into its operation collection queue i.e., $M_{k\_OCQ} \leftarrow O_{i,j}$;
  end while//Step(4)
  if D_{i_OPQ}, $\forall i \leq d$ is not empty **then**
    Return to Step(3);
  end if
  Schedule all operations according to the assignment in operation collocation queues; //Step (5)
  if (new jobs arrive) **then**
    All unprocessed job operations will be returned to the operation pool;
    Repeat Step(1) to Step(5);
  end if//Step(6)
end while

4.1 Experimental Result of Static Job Shop Environment

In the first experiment, we evaluate the A-HEFT algorithm in comparison with the state-of-the-art heuristic scheduling algorithms named LIST in Birgin et al. (2015) and Earliest Starting Time (EST) in Birgin et al. (2014) and some other intelligent algorithms including Genetic Algorithm (Li 2015), Particle Swarm Optimization Algorithm (Kennedy 2011), Differential Evolution Algorithm (Tian et al. 2016) and Estimation of Distribution Algorithm (Wang et al. 2015). We evaluate their performance by conducting experiments on job configurations from Birgin et al. (2015) and on job configurations generated randomly.

4.1.1 Experimental Result 1

For the job configurations, we use the 20 instances YFJS01-YFJS20 from the LIST approach in Birgin et al. (2015). In Table 4, the results of the C_{max} obtained by the LIST and EST heuristics are presented. We also present the optimization results by running the exact solver CPLEX for 1 h. The results of LIST, EST and CPLEX originate from the paper (Birgin et al. 2015).

The scheduling optimization results are listed in Table 4, where the 'Average' row shows the average value of makespan for those 20 instances. The relative difference (short RD in Table 4) is calculated by the flowing equation:

$$RD = \frac{average_{C_{max_{A-HEFT}}} - average_{C_{max_{another}}}}{average_{C_{max_{another}}}} \times 100\% \quad (9)$$

On average, A-HEFT improves the scheduling result by 14.85% compared with LIST and 28.71% compared with EST. The makespan of A-HEFT is only 1.78% worse than the exact CPLEX solver. Considering that A-HEFT is a heuristic algorithm with a low complexity, A-HEFT is a very competitive algorithm.

4.1.2 Experimental Result 2

In order to evaluate the scheduling performance of A-HEFT more comprehensively, various instances are randomly generated. The compared optimization approaches include LIST, and four intelligent algorithms including Genetic Algorithm (GA), Particle Swarm Optimization Algorithm (PSO), Differential Evolution Algorithm (DE) and Estimation of Distribution Algorithm (EDA). The

Table 4 Simulation results of 20 instances

Instances	C_{max}	LIST		EST		CPLEX (1 h limit)	
		C_{max}	*Improve* (%)	C_{max}	*Improve* (%)	C_{max}	*Improve* (%)
YFJS01	886	1130	21.59	1318	32.78	773	− 14.62
YFJS02	944	1133	16.68	1243	24.06	825	− 14.42
YFJS03	406	575	29.39	439	7.52	347	− 17.00
YFJS04	628	576	− 9.0	569	− 10.36	390	− 61.02
YFJS05	689	608	− 13.32	566	− 21.73	445	− 54.83
YFJS06	606	633	4.27	633	4.27	447	− 35.57
YFJS07	714	628	− 13.69	628	− 13.69	444	− 60.81
YFJS08	411	485	15.26	531	22.60	353	− 16.43
YFJS09	297	402	26.12	506	41.31	242	− 22.73
YFJS10	452	513	11.89	541	16.45	399	− 13.28
YFJS11	874	745	− 17.31	740	− 18.11	526	− 66.16
YFJS12	669	744	10.08	813	17.71	512	− 30.66
YFJS13	546	553	1.27	717	23.85	405	− 34.81
YFJS14	1443	1555	7.20	2055	29.78	1317	− 9.57
YFJS15	1454	1690	13.97	2296	36.67	1244	− 16.81
YFJS16	1468	1769	17.02	2006	26.82	1243	− 18.10
YFJS17	1274	1734	26.53	2408	47.09	1622	21.45
YFJS18	1297	1735	25.25	2082	37.71	2082	37.70
YFJS19	1224	1604	23.69	2038	39.94	1525	19.74
YFJS20	1184	1700	30.35	2369	50.02	2020	41.39
Average	873.3	1025.60	11.36	1224.9	19.73	858.05	− 18.33
RD (%)		14.85		28.71		− 1.78	

intelligent algorithm runs ten times and iterates 500 times in each run.

The job instances are generated randomly with the parameter setting being the same as that in Li (2015), Kennedy (2011), Tian et al. (2016), and Wang et al. (2015). More specifically, GA uses the same chromosome representation, two-cut points crossover, and two-cut points mutation. PSO uses inertia weight to adjust its parameters. DE creates new candidate solutions by combining existing ones according to its standard formulae. EDA uses the probability distribution from current elitist solutions and samples new solutions based on it. All job instances are supposed to be ready simultaneously at the beginning.

The scheduling results are presented in Tables 5 and 6, where Size n_1/n_2 denotes that there are n_1 jobs and n_2 machines; CPUtime denotes the computation time spent on optimizing the schedule. We observe that in most cases A-HEFT has a smaller makespan compared to other algorithms. The CPUtime of A-HEFT is also the least among all algorithms. The superiority of A-HEFT becomes larger with the increase of scheduling size, because the CPUtime of four intelligent algorithms exponentially increases with the scheduling size and cannot achieve a better optimization result. Besides, compared to the heuristic algorithm LIST, A-HEFT also performs better in achieving a smaller makespan and taking less CPUtime.

Table 5 Makespan results of random generated instances

Size	C_{max}	LIST C_{max}	GA C_{max}	PSO C_{max}	EDA C_{max}
25/5	273	331	302.5	395	484
25/10	175	206	166	318	438
30/5	244.67	266.67	240	423	467.33
35/10	172	231.5	255	415	519.5
35/15	83	141.5	187.5	326.5	352
40/5	241	256	346	464	715
40/10	115.5	146	153.5	385	393
45/5	384.5	414.5	415.5	747.5	851.5
45/10	183.5	260	270.5	685.5	724
50/10	128.5	223	271	540	660
55/10	261	291	321	925	1009
60/5	470	635	550	758	1113
70/5	445	516	583	901	1353
70/10	286	336	472	1113	1358
75/5	516.5	532.5	568	1189.5	1371.5
75/10	294	342	472	1162	1258
80/5	464	493	668	1085	1470
90/5	388	473	688.5	1507.5	1674
95/10	330	426	616	1445	1680
100/10	269	354	660	1555	1581.5
115/5	433	494	922	1774	1707
120/10	379	429	813	1701	2290
130/10	396	509	915	1917	2366
220/25	596	328	1254	2949	3802
300/25	393	328	1809.3330	4289.6670	4704

Table 6 CPU computation time of random generated instances

Size	CPUtime	LIST CPUtime	GA CPUtime	PSO CPUtime	EDA CPUtime
25/5	0.0354	0.0282	31.8222	30.5549	34.8216
25/10	0.0038	0.0086	28.6410	26.7424	32.1509
30/5	0.0099	0.0257	38.7911	37.2753	45.2682
35/10	0.0046	0.0134	54.4313	52.7051	59.7962
35/15	0.0066	0.0250	58.1557	56.4637	65.4297
40/5	0.0046	0.0443	81.1389	73.8999	84.6620
40/10	0.0053	0.0301	63.8040	63.6976	72.0407
45/5	0.0196	0.0329	75.3547	73.5840	80.4679
45/10	0.0063	0.0677	6805376	67.0708	78.9245
50/10	0.0071	0.0311	91.6843	90.7940	101.3794
55/10	0.0065	0.0208	94.8058	93.3506	100.7389
60/5	0.0057	0.0223	117.2803	116.1261	125.4103
70/5	0.0122	0.0299	168.4582	158.7811	168.1338
70/10	0.0105	0.0350	173.1519	168.4507	182.6882
75/5	0.0139	0.0435	203.2268	197.6029	216.8134
75/10	0.0121	0.0451	216.9260	198.7615	188.7718
80/5	0.0083	0.0302	255.9793	223.1716	228.6762
90/5	0.0116	0.0543	311.2892	266.2380	283.4535
95/10	0.0697	0.1619	318.3920	295.5461	318.6999
100/10	0.0414	0.0780	339.6179	329.2611	425.5674
115/5	0.0708	0.1482	431.1600	486.4686	482.2659
120/10	0.0116	0.0650	416.2304	407.1660	435.0559
130/10	0.0220	0.0883	470.5142	466.4677	490.6642
220/25	0.0227	0.3698	1734.6420	1524.807	1641.4780
300/25	0.1011	0.6094	2186.8840	2205.1020	2333.7050

4.2 Experimental Result of Dynamic Job Shop Environment

In the second experiment, we test the performance of A-HEFT by simulating the jobs arriving continuously over time in a dynamic manufacturing system. Whenever a new job arrives, the production schedule will be newly optimized to ensure that the makespan with new arrival jobs will be minimized. The reschedule will only work for unprocessed operations because processed and processing operations cannot be rescheduled.

The interval between two successive arrival jobs is around $(10, 30]$. The list scheduling algorithm from Birgin et al. (2015) is extended to handle the stochastic arrival jobs. The intelligent algorithms including PSO, GA, and

Table 7 Simulation results of dynamic DAG jobs

Size	C_{max}	LIST C_{max}	PSO C_{max}	GA C_{max}	HEDA C_{max}
25/5	463	507.5	459	444	423
30/5	582	787	982	842	659
35/5	781	874	1161.9	956	742.5
40/5	644	586	667	650	631
60/5	810	792	1009	879	834
65/5	1096	995	1522	1356	1223
70/5	1034	1143	1599	1392	1365
70/10	838	779	1376	1187	1045
75/5	1209	1050	1677	1429	1397
75/10	896	964	1566	1342	1277
80/5	1220	1013	1696	1501	1478
85/10	651.5	876.5	1365	1168	1102
90/10	851	920	1554	1348	1303
95/5	1092	1154	1764	1676	1597
95/10	826	1095	1672.5	1468	1447
100/10	1042	1029	1712	1564	1479
110/10	1294	1191	1823	1722	1567
115/5	1251	1089	2072.5	1909	1786
115/10	848	979	1675	1481	1336
120/10	1077.67	1219.33	1825	1631	1471
125/10	1085	1347	1930	1699	1601
130/10	1290	1384	2091	1844	1689
220/25	1995.33	2348.7	2672	2368.1	2029
290/25	2379.25	2406	2974.1	2861	2802

Hybrid Estimation of Distribution Algorithm (HEDA) Wang et al. (2016) are used to compare the performance of A-HEFT. The scheduling results are presented in Table 7. It can be seen that A-HEFT still performs best among all scheduling approaches.

5 Conclusions

Due to the increasingly competitive market, it is very important to minimize makespan which can reduce costs and increase production profits of many companies. In this paper, an adaptive scheduling algorithm named A-HEFT is presented to minimize the makespan of the jobs arriving continuously over time in a manufacturing system by taking the setup time and matching operation into account. The job operation processing model is presented as a directed acyclic graph in the context of the extended flexible job shop problem. Experimental results demonstrate that A-HEFT has a shorter makespan than the other state-of-the-art algorithms when scheduling static jobs. Besides, when scheduling dynamic jobs, our adaptive scheduling algorithm can obtain the optimal solutions for large-size job shop problems in real time.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable feedback. This work was supported in part by the Beijing Municipal Natural Science Foundation under Grant 4162046, in part by the National Natural Science Foundation of China under Grant 61802013, in part by the Talent Foundation of Beijing University of Chemical University under Grant buctrc201811, in part by the Open Research Project of the State Key Laboratory of Synthetical Automation for Process Industries under Grant H2018294, and in part by the Fundamental Research Funds for the Central Universities under Grant XK1802-4.

References

Alvarez-Valdesabacc R (2005) A heuristic to schedule flexible job-shop in a glass factory. Eur J Oper Res 165(2):525–534

Birgin EG, Feofiloff P, Fernandes CG, De Melo EL, Oshiro MT, Ronconi DP (2014) A milp model for an extended version of the flexible job shop problem. Optim Lett 8(4):1417–1431

Birgin EG, Ferreira JE, Ronconi DP (2015) List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility. Eur J Oper Res 247(2):421–440

Borenstein D (2000) A directed acyclic graph representation of routing manufacturing flexibility. Eur J Oper Res 127(1):78–93

Brucker P, Schlie R (1990) Job-shop scheduling with multi-purpose machines. Computing 45(4):369–375

Cao Y, Jia LW, Cao S, Bai Y (2014) Visualized modeling and simulation of manufacturing execution system in dynamic job-shop scheduling. Appl Mech Mater 496–500:1498–1501

Cao Z, Lin C, Zhou M, Huang R (2017) An improved cuckoo search algorithm for semiconductor final testing scheduling. In: Automation science and engineering (CASE), pp 1040–1045

Fink A, Kliewer N, Mattfeld D, Mönch L, Rothlauf F, Schryen G, Suhl L, Voß S (2014) Model-based decision support in manufacturing and service networks. Bus Inf Syst Eng 6(1):17–24

Gan PY, Lee KS (2002) Scheduling of flexible-sequenced process plans in a mould manufacturing shop. Int J Adv Manuf Technol 20(3):214–222

Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1(2):117–129

Hoffmann K, Buscher U, Neufeld JS, Tamke F (2017) Solving practical railway crew scheduling problems with attendance rates. Bus Inf Syst Eng 59(3):147–159

Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning. Springer, Berlin, pp 760–766

Kriglstein S, Leitner M, Kabicher-Fuchs S (2016) Evaluation methods in process-aware information systems research with a perspective on human orientation. Bus Inf Syst Eng 58(6):1–18

Lee S, Moon I, Bae H, Kim J (2012) Flexible job-shop scheduling problems with 'and'/'or' precedence constraints. Int J Prod Res 50(7):1979–2001

Li Y (2015) Combined scheduling algorithm for re-entrant batch-processing machines in semiconductor wafer manufacturing. Int J Prod Res 53(6):1866–1879

Ma H (2010) Process-aware information systems: bridging people and software through process technology. J Assoc Inf Sci Technol 58(3):455–456

Raileanu S, Borangiu T, Morariu O, Stocklosa O (2014) Ilog-based mixed planning and scheduling system for job-shop

manufacturing. In: IEEE international conference on automation, quality and testing, robotics, pp 1–6

Schryen G, Rauchecker G, Comes T (2015) Resource planning in disaster response. Bus Inf Syst Eng 57(4):243–259

Tian G, Ren Y, Zhou MC (2016) Dual-objective scheduling of rescue vehicles to distinguish forest fires via differential evolution and particle swarm optimization combined algorithm. IEEE Trans Intell Transp Syst 17(11):3009–3021

Topcuouglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274

Ulmer MW, Heilig L, Voß S (2017) On the value and challenge of real-time information in dynamic dispatching of service vehicles. Bus Inf Syst Eng 59(2):1–11

Varvara G (2016) Service architecture for CSP based planning for holonic manufacturing execution systems. In: International conference on exploring services science, pp 403–416

Vilcota G, Billautb JC (2008) A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. Eur J Oper Res 190(2):398–411

Wang H, Liu L, Fei Y, Liu T (2014) A collaborative manufacturing execution system oriented to discrete manufacturing enterprises. In: International conference on cooperative design, visualization and engineering, pp 277–285

Wang HK, Chien CF, Gen M (2015) An algorithm of multi-subpopulation parameters with hybrid estimation of distribution for semiconductor scheduling with constrained waiting time. IEEE Trans Semicond Manuf 28(3):353–366

Wang L, Wang S, Zheng X, Automation DO, University T (2016) A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times. IEEE/CAA J Autom Sin 3(3):235–246

Xie G, Zeng G, Li Z, Li R, Li K (2017) Adaptive dynamic scheduling on multifunctional mixed-criticality automotive cyber-physical systems. IEEE Trans Veh Technol 66(8):6676–6692

Zeng J, Jacson S, Lin L, Gustafson J, Hoarau E, Mitchell R (2010) On-demand digital print operations a simulation based case study. Hewlett-Packard. Techchnical, report

Zhou L, Chen Z, Chen S (2015) An effective detailed operation scheduling in MES based on hybrid genetic algorithm. J Intell Manuf 29:1–19