Computers & Operations Research 134 (2021) 105401

Contents lists available at ScienceDirect

Computers and Operations Research

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

Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints

Huali Fan [a,*], Hegen Xiong [b], Mark Goh [c]

[a] School of Electrical and Electronic Engineering, Nanyang Technological University. 50 Nanyang Avenue 639798, Singapore
[b] School of Mechanical Engineering and Automation, Wuhan University of Science and Technology. No. 947, Heping Avenue, Qingshan District, Wuhan 430081, China
[c] NUS Business School & The Logistics Institute - Asia Pacific, National University of Singapore. 21 Lower Kent Ridge Road 119077, Singapore

ARTICLE INFO

Keywords:
Dynamic job shop scheduling
Dispatching rules
Genetic programming
Hyper-heuristic
Extended technical precedence constraints

ABSTRACT

Extended technical precedence constraints (ETPC) in dynamic job shop scheduling problem (DJSP) are the precedence constraints existing between different jobs instead of the conventional technical precedence constraints existing in the operations of the same job. This paper presents the mathematical programming model of the DJSP with ETPC to minimize the mean weighted tardiness of the jobs. The mathematical model contributes to the solution and modelling of the DJSP with ETPC and it is used to solve small-sized problems to optimality. To solve industry-sized problems, a constructive heuristic called the dispatching rule (DR) is employed. This paper investigates the use of genetic programming (GP) as a hyper-heuristic in the automated generation of the problem-specific DRs for solving the problem under consideration. The genetic programming-based hyper heuristic (GPHH) approach constructs the DRs which are learned from the training instances and then verified on the test instances by the simulation experiments. To enhance the efficiency of the approach when evolving effective DRs to solve the problem, the approach is improved with strategies which consist of a problem-specific attribute selection for GP and a threshold condition mechanism for fitness evaluation. The simulation results verify the effectiveness and efficiency of the evolved DRs to the problem under consideration by comparing against the existing classical DRs. The statistical analysis of the simulation results shows that the evolved DRs outperform the selected benchmark DRs on the problem under study. The sensitivity analysis also shows that the DRs generated by the GPHH approach are robust under different scheduling performance measures. Moreover, the effects of the model parameters, including the percentage of jobs with ETPC and the machine utilization, on the performance of the DRs are investigated.

1. Introduction

Scheduling is concerned with the allocation of resources to tasks over given time periods, seeking to optimize one or more objectives (Pinedo, 2008). The industry contains many types of production scheduling problems, and the job shop scheduling problem (JSP) is one of the most complex ones. The JSP attracts lots of attention from researchers due to its wide applicability in the manufacturing sector and its inherent difficulty as a NP-complete combinatorial optimization problem.

Many research works have considered the extended models of the classical JSP by relaxing several assumptions of the problem with the intent to make them more practical for specified application scenarios. For example, Holthaus (1999) considered scheduling in dynamic job shops with machine breakdown, Rossi and Dini (2007) addressed the JSP with routing flexibility and separable setup times, Nie et al. (2013) studied scheduling in a job shop with jobs arriving over time, Ahmadi et al. (2016) investigated the flexible JSP, and Xiong et al. (2017) presented a dynamic JSP with extended technical precedence constraints (ETPC). The extant literature suggests that the classical JSP with the objective of minimizing makespan is a NP-complete combinatorial optimization problems (Brucker et al., 2007; Garey et al., 1976; Lenstra et al., 1977), so the extended models of the JSP are NP-complete problems as well.

Moreover, there are usually a large number of jobs which arrive to the shop intermittently over time but they need to be scheduled quickly for the practical application of JSP in manufacturing sector. The JSP

* Corresponding author.
E-mail addresses: hl.fan@hotmail.com (H. Fan), mark.goh@nus.edu.sg (M. Goh).

https://doi.org/10.1016/j.cor.2021.105401
Received 3 November 2020; Received in revised form 14 April 2021; Accepted 25 May 2021
Available online 4 June 2021

with intermittently arriving jobs is considered as dynamic job shop scheduling problem (DJSP). DJSP is an extension of the general JSP in the presence of real time events. The real time events could be job arrival (such as randomly arriving jobs, continuously arriving new jobs, and intermittently arriving jobs), machine breakdown, tool unavailability, job cancellation, due date change, et al. To solve the DJSP, frequent schedule modification and rapid reaction to the changing system requirements are required. Therefore, the classical search-based techniques that can obtain good quality solutions are not applicable as they search the solution space which is not only computationally expensive but also cannot react fast enough to the dynamic changes. Thus, scheduling heuristics which can deliver acceptable but not necessarily optimal solutions in a short computational time have been adopted for solving the DJSP. Among these heuristics, a constructive heuristic which starts with an empty solution and repeatedly extends the current solution until a complete solution is obtained is more prominent. The constructive heuristic, or the dispatching rule (DR), which is a particularly simple type of scheduling heuristic, is now widely used due to its ease of implementation and low-time complexities.

In short, the DR (Haupt, 1989; Panwalkar and Iskander, 1977) is a particularly simple type of scheduling heuristics that are used to determine the next job with the highest priority amongst the waiting jobs to be processed on a machine whenever the machine is available. The priority of the jobs is evaluated based on the priority function of the DR which usually consists of the attribute information of the jobs and machines. The priority function of the DR is expected to consist of information that reflects the next state of the production system so that the decisions made are more global for the entire system. The intuitive nature of the DRs affords them a prominent advantage, namely they are robust to the variability and uncertainty in production system. This is because as a constructive heuristic, it computes the job priorities based on the shop attribute information in the priority function and makes decisions at the last possible moment. These characteristics make them possess the ability to better react to the dynamic changes in system. Researchers in the field have proposed various types of DRs for solving a large amount of complicated, dynamic and large-sized production scheduling problems, and most of the DRs have shown good performance for the specified scheduling environment (Ferrell et al., 2000; Chan et al., 2003; Branke and Pickardt, 2011; Chen and Matis, 2013; Schaller and Valente, 2019; Siebert et al., 2018; Sweeney et al., 2019; Xanthopoulos and Koulouriotis, 2018).

While popular, the DRs have a drawback. Existing studies have shown that no single DR outperforms all others across all shop configurations, operating conditions, and objective functions (Blackstone et al., 1982; Holthaus and Rajendran, 1997; Rajendran and Holthaus, 1999). Hence, many researchers focused on designing rules for specific scheduling scenarios (Baker and Bertrand, 1982; Jayamohan and Rajendran, 2004; Raghu and Rajendran, 1993; Siebert et al., 2018) and objective functions (Chen and Matis, 2013; Rajendran and Holthaus, 1999; Thiagarajan and Rajendran, 2005; Schaller and Valente, 2019) manually by making combinations of the existing rules to achieve better scheduling outcomes for the specified scheduling environment. The manually designed rules usually perform well under the specified scheduling environment, as they take into account more shop attribute information related to the specific scheduling scenarios than the benchmark rules do. However, developing the DRs in this way requires much expertise and time as the procedure involves trial-and-error. This time-consuming process can be automated by hyper-heuristic approaches such as machine learning, especially genetic programming (GP), to search the space of heuristics to generate effective problem-specific DRs. This paper investigates the genetic programming-based hyper-heuristic (GPHH) approach for the automated design of the DRs to solve the DJSP with ETPC so as to minimize the mean weighted tardiness of the jobs.

The remainder of the paper is organized as follows. Section 2 gives a detailed literature review of the research works that studied the GP as a hyper-heuristic to design DRs for solving the production scheduling problems. In Section 3, the formulation of DJSP with ETPC is addressed, the mathematical programming model is presented and then verified by applying the model to solve a problem instance. Section 4 illustrates the framework of the GPHH approach and the improvement strategies of the approach. The experimental design for training the candidate DRs and testing the DRs evolved by the GPHH approach is given in Section 5, and the experimental results are presented. In Section 6, the results are analyzed to verify the effectiveness and efficiency of the DRs generated by the GPHH approach, and the effects of the model parameters on the performance of DRs are discussed. Finally, Section 7 concludes the paper with some suggestions for future work.

2. Literature review

Hyper-heuristic is an automated method for selecting or generating heuristics through machine learning to solve computational search problems (Burke et al., 2010). To generate heuristics for solving a problem, the hyper-heuristic usually combines the components used in the existing heuristics by various operators to form new heuristics which are then trained on the training problem instances and evolved to become more effective. GP is an evolutionary computation-based method. It generates an initial population randomly and let the population evolve generation by generation based on the genetic manipulation operation (Koza and NetLibrary, 1992). The individuals in GP can be represented by different representations (Branke et al., 2015), and the tree-based representation is the most popular one. Tree-based representation let the individuals be represented as trees of various lengths. The intuitive characteristic of the individual representation scheme of the GP grants the GP the ability to create trees with different formats and lengths, making it suitable for use in generating the DRs automatically as a hyper-heuristic.

To solve the production scheduling problems, the GPHH approach searches the space of heuristics through the GP to discover the heuristics that can be used to solve the problem effectively. The GPHH approach has been used to automate the design of the DRs for a variety of production scheduling problems with various scheduling scenarios and objective functions.

Atlan et al. (1994) presented a GP-based system to design composite DRs for distributed reactive scheduling problems. The rules obtained near-optimal solutions and they were robust to perturbations in the processing time. Dimopoulos and Zalzala (1999) used GP as a hyper-heuristic to combine the existing DRs to solve the one-machine total tardiness problem. Their results showed that the combination of the DRs performed no worse than the individual DRs in most cases. In a follow-up study, Dimopoulos and Zalzala (2001) applied GP to generate the DRs to solve the one-machine total tardiness scheduling problem with various levels of tardiness and due date tightness. The improvement of the latter study was that the fundamental attributes of the problem were used to construct the evolved DRs. The results showed that the evolved rules performed well, or at least they were able to produce tardiness levels that were as good as the benchmarks. Geiger et al. (2006) proposed a system combining an evolutionary learning mechanism with a simulation model of an industrial facility to automate the process of evolving various rules and evaluate their performance. The rules discovered by the GP-based learning system exhibited better or at least similar performance compared with the benchmark DRs for a variety of single machine environments. Their study reported that the relationships of the attributes that comprise the rule and their relevance to the problem structure are more important than the actual rule itself. The study concluded that an effective scheduling rule can be designed if meaningful relationships among the set of system attributes are considered. Geiger and Uzsoy (2008) then extended the developed genetic learning approach to automatically synthesize the priority functions for the DRs for batch scheduling environments. Their simulation experiments showed that the DRs yielded good system performance

compared to the benchmark rules for the problem under study.

Based on the findings and conclusions given by Geiger et al. (2006) and Geiger and Uzsoy (2008), many researchers then investigated the use of GPHH approach to automate the design of the DRs for dynamic production scheduling problems and the generated DRs yielded good performance for the specified scheduling problems in their study (Tay and Ho, 2008; Hildebrandt et al., 2010; Jakobović and Marasović, 2012; Nguyen et al., 2013b; Pickardt et al., 2013; Hart and Sim, 2016). Moreover, the performance of the DRs is enhanced by incorporating machine and shop status information into the rules as the status information can help to make better informative sequencing decisions (Geiger et al., 2006; Nguyen et al., 2013a). So, many research works take not only the job attributes but also the machine and shop attributes into consideration when using the GPHH approach to construct the DRs.

Jakobović and Budin (2006) investigated the use of GP in the automated synthesis of scheduling heuristics in which the next state of the system was included. The evolved new heuristics were compared with the existing scheduling heuristics for solving the dynamic single machine scheduling problem and the JSP. Their experimental results showed that, for some problems, the evolved heuristics exhibited better performance than the existing scheduling heuristics. Nguyen et al. (2012) proposed the GPHH approach for evolving scheduling polices consisting of the DRs and the due-date assignment rules for multi-objective job shops. The terminals of the GP in their study comprise status information about the jobs, machines, and shop. Their experimental results showed that the evolved scheduling polices can outperform various combinations of the existing scheduling rules and they were robust in the stochastic and dynamic job shops. Nguyen et al. (2013a) employed the GP as a hyper-heuristic to automatically discover adaptive DRs for the JSP. Their results showed that the quality of the evolved rules improved after integrating the machine and system attributes in the priority functions and the evolved rules outperformed the selected existing rules. Nguyen et al. (2014) developed the GPHH methods which can handle multiple scheduling decisions and conflicting scheduling objectives for the automated design of the scheduling policies in a job shop environment. The terminals in the GP consists of status information on the job, machine, and shop. The evolved scheduling polices showed dominating performance on simulation scenarios under different shop settings. Hunt et al. (2014) investigated the GPHH approach in which the features of the current state of the shop system were incorporated in the terminals of the GP to develop less myopic DRs for the DJSP. The DRs yielded better average performance and the standard deviation of the performance was also decreased after adding features of the shop state in the terminals of the GP.

To understand the development of the hyper-heuristics for the highly dynamic and stochastic production scheduling problems, Branke et al. (2016b) provided a systematic review of the design choices and critical issues involved in the process of the development. Besides, as the configurations and requirements of GP for the production scheduling are more complicated compared with other applications of GP or evolutionary computation approaches in scheduling applications, Nguyen et al. (2017) gave a unified framework survey on the automated design of production scheduling heuristics to address how GP are applied to the task and the key components involved.

In addition to the aforementioned, the GPHH approach has been used to generate scheduling heuristics for the production scheduling problems with various characteristics, such as the DJSP with dynamic job arrivals and machine breakdowns (Park et al., 2017), and the dual-constrained flow shop scheduling with machines and operators (Branke et al., 2016a). Moreover, some research works focused on improving the performance of the GPHH approach through various strategies. Park et al. (2018) improved the robustness of the GPHH approach by employing ensemble learning for the ensemble GPHH approach through various combination schemes for the DJSP. Zhou et al. (2020) improved the computational efficiency and the offline learning process of the hyper-heuristic without deteriorating its performance by proposing a surrogate-assisted cooperative coevolution GP method for the dynamic flexible JSP.

In this paper, the GPHH approach is used for the automated design of the DRs for solving the DJSP with ETPC. The dynamic event of the DJSP considered here is the intermittently arriving jobs. The ETPC are the extension of the conventional routing-based technical precedence constraints in JSP. It underscores a common situation in the practical production processes in mould and die manufacturing. The ETPC exist in various production processes such as electrical discharge machining, assembly line and casting process (Xiong et al., 2017). It defines the precedence processing relations existed between the operations of different jobs and the time lapse required between the processing of the related jobs. Some similar constraints, such as the precedence relations existed between different jobs and the time lapse required between the processing of different jobs, were considered separately for JSP in existing literature. They are the assembly JSP (Sculli, 1980; Thiagarajan and Rajendran, 2005) and the JSP with the consideration of separable sequence-dependent setup times (Allahverdi et al., 2008; Rossi and Dini, 2007) respectively. The constraint as depicted by the ETPC is a combination of these two types of constraints. The DJSP with the consideration of ETPC is more practical for production scheduling systems in specified manufacturing sectors, and the efficiencies of the systems are improved by considering the ETPC. Xiong et al. (2017) proposed four DRs manually which had good performance for the problem under study. This paper investigates the use of GPHH approach for the automated design of effective and efficient DRs for the DJSP with ETPC to minimize the mean weighted tardiness of jobs. The main contributions of this paper can be summarized as follows.

(1) The mathematical programming model of the DJSP with ETPC is presented and it is used to provide optimal solutions for the small-sized problems by mathematical programming solver. The mathematical model contributes to both the solution and the modelling of the DJSP with ETPC.

(2) The GPHH approach with improvement strategies is proposed for the automated design of DRs for solving the problem under study. The effectiveness and efficiency of the generated DRs to the problem under consideration is verified by statistical analysis.

(3) The effects of the problem model parameters including the percentage of jobs with ETPC and the machine utilization on the scheduling performance of the DRs are studied.

3. Problem formulation

In this section, the problem formulation of the DJSP with ETPC is given, the mathematical programming model of the problem under study is presented, and the effectiveness of the model to the problem is verified by applying the model to solve a small-sized problem instance to optimality.

3.1. Problem formulation

The classical JSP comprises a set of jobs $\mathrm{J} = \{J_i\}_{i=1}^{n}$, each having its own processing order through a set of machines $\mathrm{M} = \{M_k\}_{k=1}^{m}$. Each job J_i, associated with a due date d_i and a job weight w_i, has an ordered set of operations $O_i = \{O_{ij}\}_{j=1}^{l_{(i)}}$, and each operation O_{ij} must be processed on a given machine M_k with a given processing time p_{ij}. Besides, the jobs arrive at the shop intermittently over time for the studied DJSP and the arrival time of job J_i is a_i. The problem follows standard assumptions such as one machine can process only one job at a time, each job is processed on at most one machine at a time, no machine breakdown, and no preemption. The goal of the problem is to find the production sequence of jobs on the machines by following the constraints of the problem such that the scheduling objective is optimized.

The technical precedence constraint in the classical JSP is usually the routing-based precedence constraint between the operations within a

job which dictates that operation $O_{i(j+1)}$ can only be started after its preceding operation O_{ij} has been completed. The ETPC, proposed by Xiong et al. (2017), describe the technical precedence constraints existed between the operations of different jobs and the finite time lapse required between the processing of the related operations of the jobs. According to Xiong et al. (2017), there are four types of ETPC in practical production and they can be transformed to one common type called type *SC*. In this case, if two operations O_{ij} and $O_{i'j'}$ from a pair of two jobs J_i and $J_{i'}$ with the SC form of precedence constraints, it means that operation $O_{i'j'}$ (the hind-relative operation) can be started only after operation O_{ij} (the fore-relative operation) has been completed for a time lapse $g^{ij}_{i'j'}$. The binary parameter $t^{ij}_{i'j'}$ is introduced to denote whether two jobs J_i and $J_{i'}$ are a pair of jobs with SC form of ETPC. If $t^{ij}_{i'j'}$ is equal to 1, it means that the two operations O_{ij} and $O_{i'j'}$ from the job pair have ETPC, then the precedence constraint for the processing of the two operations is expressed as follows.

$$s_{i'j'} \geq c_{ij} + g^{ij}_{i'j'} \quad \forall \mathrm{i}, i' \in \mathrm{J}, \forall \mathrm{j} \in O_i, \forall j' \in O_{i'}, \mathrm{i} \neq i' \tag{1}$$

where $s_{i'j'}$ is the start time of the hind-relative operation $O_{i'j'}$, and c_{ij} is the completion time of the fore-relative operation O_{ij}. In this paper, all the types of the ETPC are transformed to the SC type and the JSP with the SC form of ETPC is studied in the mathematical model and the simulation study.

3.2. Mathematical programming model

Mathematical programming formulation is one of the most general optimization methods for obtaining the optimal solutions of the problems. In this section, the notation, parameters, and variables of the mathematical model are given first, then the mathematical programming model of the problem under study is presented.

(1) Indices and sets

J	set of jobs
M	set of machines
O	set of operations
O_i	ordered set of operations of job i($O_i \subseteq O$), where O_{i1} and $O_{il_{(i)}}$ are the first and last elements of O_i respectively($O_{i1}, O_{il_{(i)}} \in O_i$)
M_i	machine set on which the operations of job i can be processed ($M_i \subseteq M$)
M_k	machine on which operation O_{ij} can be processed ($M_k \in M_i$)
i, i'	job index ($i, i' \in J$)
j, j'	operation index ($j, j' \in O$)
k	machine index ($k \in M$)

(2) Parameters

n	number of jobs
m	number of machines
p_{ij}	processing time of operation j of job i (O_{ij}) on machine$k$
L	a large positive number
a_i	arrival time of job$i$
d_i	due date of job$i$
w_i	job weight
$t^{ij}_{i'j'}$	binary parameter denotes whether two operations O_{ij} and $O_{i'j'}$ have ETPC. It is equal to 1 if the two operations have ETPC, 0 otherwise
$g^{ij}_{i'j'}$	time lapse required between the processing of O_{ij} and $O_{i'j'}$ with ETPC

(3) Decision variables

s_{ij}	start time of operation O_{ij} on machine$k$
c_{ij}	completion time of operation O_{ij} on machine$k$
c_i	completion time of job$i$
T_i	tardiness time of job$i$
$Y_{i'ik}$	binary variable that is equal to 1 if job i' precedes job i on machine k, 0 otherwise
T_a	mean weighted tardiness of the schedule

(4) Mathematical programming model

The mathematical programming model of the DJSP with ETPC may be stated as follows.

Objective:

$$\text{Min } T_a$$

subject to:

$$s_{i1} \geq a_i \quad \forall \mathrm{i} \in \mathrm{J} \tag{3}$$

$$c_{ij} \geq s_{ij} + p_{ij} \quad \forall \mathrm{i} \in \mathrm{J}, \forall \mathrm{j} \in O_i \tag{4}$$

$$s_{i'j'} \geq c_{ij} - (Y_{i'ik})L \quad \forall \mathrm{i}, i' \in \mathrm{J}, \forall \mathrm{j} \in O_i, \forall j' \in O_{i'}, \mathrm{i} \neq i', \forall k \in M_i \cap M_j \tag{5}$$

$$s_{ij} \geq c_{i'j'} - (1 - Y_{i'ik})L \quad \forall \mathrm{i}, i' \in \mathrm{J}, \forall \mathrm{j} \in O_i, \forall j' \in O_{i'}, \mathrm{i} \neq i', \forall k \in M_i \cap M_j \tag{6}$$

$$Y_{i'ik} + Y_{ii'k} = 1 \quad \forall \mathrm{i}, i' \in \mathrm{J}, \forall k \in M_i \cap M_j, \mathrm{i} \neq i' \tag{7}$$

$$c_{ij} + g^{ij}_{i'j'} - s_{i'j'} \leq (1 - t^{ij}_{i'j'})L \quad \forall \mathrm{i}, i' \in \mathrm{J}, \forall \mathrm{j} \in O_i, \forall \mathrm{j}' \in O_{i'}, \mathrm{i} \neq i' \tag{8}$$

$$s_{ij} \geq c_{i(j-1)} \quad \forall \mathrm{i} \in \mathrm{J}, \forall \mathrm{j} \in O_i \backslash O_{i1} \tag{9}$$

$$c_i = c_{il_{(i)}} \quad \forall \mathrm{i} \in \mathrm{J} \tag{10}$$

$$T_i = \max\{0, c_i - d_i\} \quad \forall \mathrm{i} \in \mathrm{J} \tag{11}$$

$$T_a = \left(\sum\nolimits_{i=1}^{n} w_i T_i\right) \Big/ \left(\sum\nolimits_{i=1}^{n} w_i\right) \quad \forall \mathrm{i} \in \mathrm{J} \tag{12}$$

$$s_{ij} \geq 0 \quad \forall \mathrm{i} \in \mathrm{J}, \forall \mathrm{j} \in O_i \tag{13}$$

$$c_{ij} \geq 0 \quad \forall \mathrm{i} \in \mathrm{J}, \forall \mathrm{j} \in O_i \tag{14}$$

$$c_i \geq 0 \quad \forall \mathrm{i} \in \mathrm{J} \tag{15}$$

$$Y_{i'ik} \in \{0, 1\} \quad \forall \mathrm{i}, i' \in \mathrm{J}, \mathrm{i} \neq i', \forall k \in M_i \cap M_j \tag{16}$$

Constraint (3) states that the start time of the first operation of a job should be later than the arrival time of the job to the shop. Constraint (4) guarantees that the difference between the completion time and start time of an operation is at least equal to its processing time on the machine. Constraints (5) and (6) ensure that two operations O_{ij} and $O_{i'j'}$ processed on the same machine cannot be done simultaneously. Constraint (7) states that two jobs processed on the same machine can only have one processing sequence. Constraint (8) guarantees that the ETPC are obeyed for the operations from the job pair with the SC form of extended precedence constraints. Constraint (9) ensures that the precedence relationships between the operations of a job are not violated, i.e. operation O_{ij} can only be started after its preceding operation $O_{i(j-1)}$ has been completed. Constraints (10) and (11) determines the value of the completion time and the tardiness of the job respectively. Constraints (12) defines the mean weighted tardiness of the jobs in the schedule. Constraints (13)–(16) are the non-negativity and integrity conditions on the variables.

3.3. Verification of mathematical programming model

To verify the effectiveness of the presented mathematical model to the DJSP with ETPC, the model is applied to solve a problem instance to optimality using the mathematical programming solver Gurobi in

Python.

(1) Problem instance information

The problem instance includes 6 jobs that need to be processed on 5 machines with the machine utilization of 85%. The job information of the problem instance and their corresponding value are given in Table 1. The symbol "-" in the table denotes that the job does not need to be processed on the corresponding machine in the column. For example, job J_1 has 4 operations that need to be processed on 4 machines following the processing sequence of O_{11} on M_4, O_{12} on M_1, O_{13} on M_5 and O_{14} on M_3.

For the information related to the jobs with ETPC, the pairs of jobs with ETPC and the related fore-relative operation and hind-relative operation are listed in Table 2. As shown in Table 2, there are 2 pairs of jobs with ETPC, and their related operations are given. For example, the first column in Table 2 says that a pair of jobs J_4 and J_2 has ETPC. The fore relative operation is O_{41} and the hind-relative operation is O_{22}, and the required time lapse between the processing of the two operations is 6.

(2) Application of model to the problem instance

To make comparative study, the optimal solutions of the problem instance with and without ETPC are obtained and the optimal schedules are expressed by their equivalent Gantt charts respectively. For the two scenarios of the problem instance, their job information is the same as shown in Table 1. The only difference between the two scenarios is that the instance with ETPC has additional ETPC with the related information as given in Table 2.

The optimal solution of the instance without ETPC is given by the Gantt chart as shown in Fig. 1. The production sequences of the dynamically arriving jobs on the processing machines are displayed in the Gantt chart. The corresponding objective value T_a of the optimal schedule to the problem instance is 1.64. Besides, the makespan of the schedule is 93.

The optimal solution of the problem instance with ETPC is obtained by the presented mathematical programming model. The optimal schedule is expressed by the Gantt chart as shown in Fig. 2. The production sequences of the dynamically arrival jobs on the processing machines are given. The optimal solution as expressed by the Gantt chart shows that the schedule follows the ETPC given in Table 2. The corresponding objective value T_a of the optimal schedule to the problem instance is 2.76 and the makespan of the schedule is 109.

As given in Figs. 1 and 2, the optimal solutions to the problem instance with and without ETPC are different. The Gantt chart in Fig. 2 shows that the jobs with ETPC follow the constraints, and this verifies the effectiveness of the presented mathematical model to the problem under study. So the mathematical model not only provides a formal description of the problem, but also contributes to the solution and modelling of the DJSP with ETPC.

Although the mathematical model can be used to obtain optimal solutions for the problems by using commercial mathematical solvers like Gurobi and CPLEX, such software is limited to solving problems with relatively small size. It cannot obtain satisfactory solutions in a reasonable amount of time for large-sized problems due to the computational challenge. This is because JSP is NP-hard, the number of variables and constraints of the problem grow quickly with the increase of the number of jobs. The DJSP with ETPC in practical applications is usually a large-scale problem with the intermittently arriving jobs. To solve such large-sized problems, heuristics that can yield near-optimal solutions need to be employed.

Table 2
Jobs with ETPC in the problem instance

Job pair with ETPC	Fore-relative operation	Hind-relative operation	Time lapse
(J_4, J_2)	O_{41}	O_{22}	6
(J_3, J_5)	O_{32}	O_{53}	8

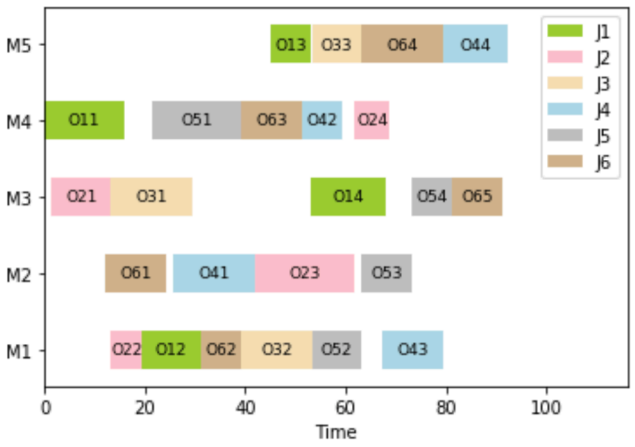


Fig. 1. The Gantt chart of the problem instance without ETPC

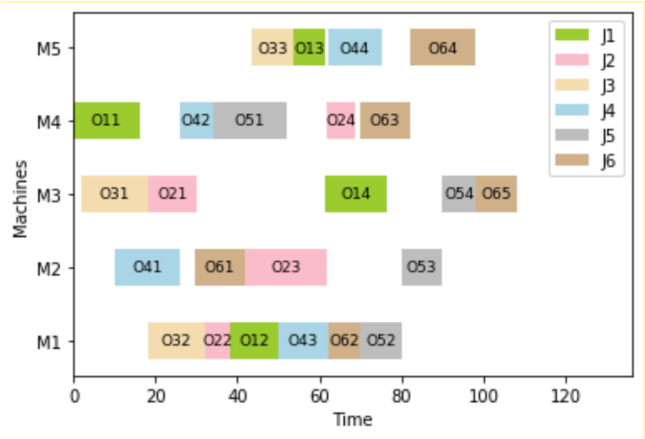


Fig. 2. The Gantt chart of the problem instance with ETPC

Table 1
Job information for the problem instance

Job	Job weight	Arrival time	Job operations and their processing machine									
			M_1		M_2		M_3		M_4		M_5	
J_1	4	0.00	12	O_{12}	-	-	15	O_{14}	16	O_{11}	8	O_{13}
J_2	2	1.30	6	O_{22}	20	O_{23}	12	O_{21}	7	O_{24}	–	–
J_3	2	2.03	14	O_{32}	–	–	16	O_{31}	–	–	10	O_{33}
J_4	2	10.01	12	O_{43}	16	O_{41}	–	–	8	O_{42}	13	O_{44}
J_5	1	11.08	10	O_{52}	10	O_{53}	8	O_{54}	18	O_{51}	–	–
J_6	2	12.09	8	O_{62}	12	O_{61}	10	O_{65}	12	O_{63}	16	O_{64}

4. GPHH approach

In this section, the GPHH approach with improvement strategies for evolving DRs for solving the DJSP with ETPC is presented. First, the framework of GPHH approach is introduced. Next, the design of GP as a hyper-heuristic is addressed. Then the improvement strategies for the GPHH approach are proposed.

4.1. Framework of GPHH approach

The GPHH approach generates new DRs by learning from the problem instances in the training set. The effective DRs are learned by unsupervised learning approach. This is achieved by applying the candidate DRs generated by GP to the problem instances in the training set to evaluate their performance, and then guiding the search of the candidate DRs towards more promising ones in the search space based on the performance evaluation.

The GPHH approach consists of two stages which are the training stage and the test stage. In the training stage, the DRs are trained by the problem instances with different scheduling scenarios in the training set to become more adaptive. Then the effectiveness and efficiency of the best evolved DRs are verified by comparing their performance with the benchmark rules on the problem instances with different scheduling scenarios in the test set. Fig. 3 provides an illustration of the systematic framework of the GPHH approach for the DRs generation and the performance evaluation.

In the training stage, the GPHH approach starts with an initial population of randomly generated individuals which are the candidate DRs. The quality of each candidate DR in the population is assessed by its fitness that is calculated based on the fitness function. The next population of the candidate DRs is evolved based on the current population by the genetic manipulation operation if the current population does not meet the termination condition. The quality of the candidate DRs in the next population is evaluated, and the cycle continues until the termination condition is met.

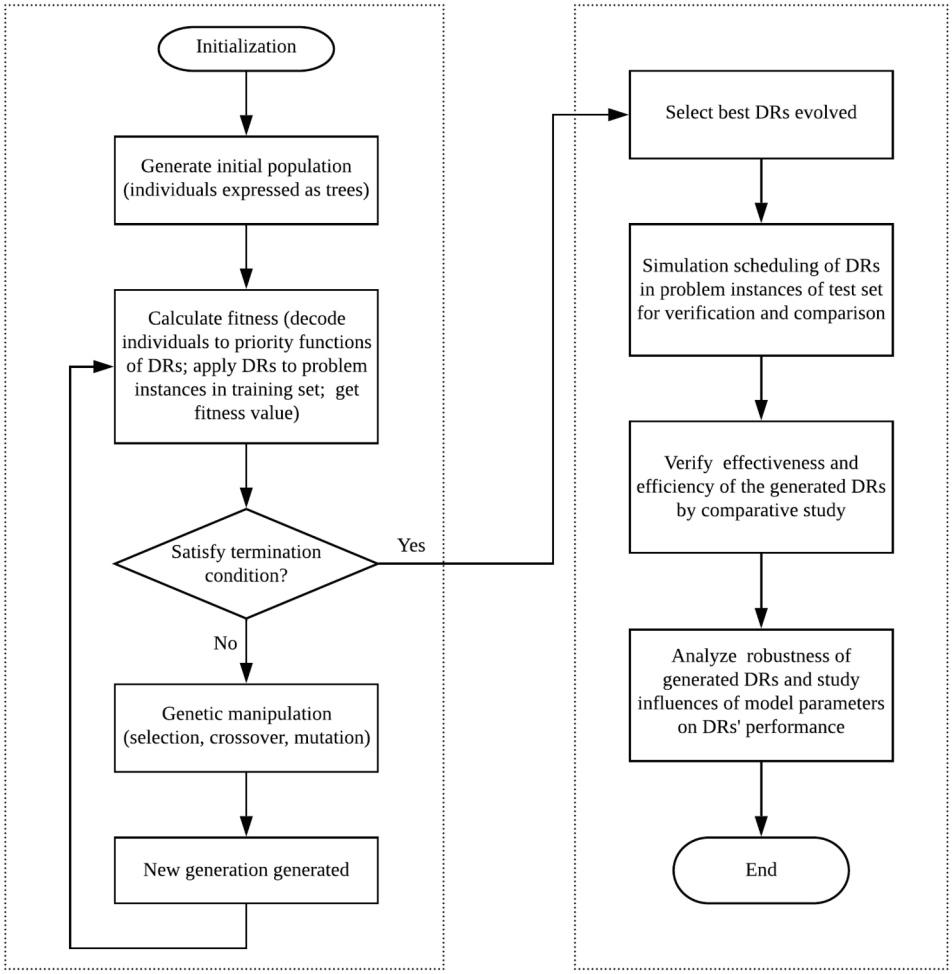
In the test stage, the performance of the best evolved DRs generated by GP are evaluated by applying the DRs to the problem instances in the test set, and their performance are compared to the existing benchmark rules to show their performance superiority over others for the problem under study.

The pseudocode of the GPHH approach for the evolution of the DRs in the training stage is given in Algorithm 1.

Algorithm 1: Pseudocode of GPHH approach for the evolution of DRs in training stage

Input: A number of problem instances in training set (*trainSet*)
Output: A set (*indSetBest*) of best evolved DRs (*indBest*)
1: **Initialization:** Generate the initial population randomly
2: Set each *indBest* in *indSetBest* ← *null* and *fitness*(*indBest*) ←+∞
3: *gen* ← 0
4: **while** *gen* < *maxGen* **do**
5: // **Evaluation**: evaluate the individuals in the population by calculating their fitness
6: **for** $i = 1$ **to** *populationSize* **do**

(*continued on next page*)



(a) Training stage (b) Test stage

Fig. 3. Framework of GPHH approach for DRs generation and performance evaluation

(*continued*)

Algorithm 1: Pseudocode of GPHH approach for the evolution of DRs in training stage

```
        objValInd_i ← 0
      for j = 1 to trainSet.length do
        // Calculate objective value by applying individual (DR) ind_i to instance trainset[j]
        objValIndPro_ij: objective value of ind_i for solving problem instance trainset[j]
        // Update sum of scheduling objective value of ind_i on problem instances in trainSet
        objValInd_i = objValInd_i + objValIndPro_ij
      endfor
      // Calculate fitness of ind_i as the average value of objective values of ind_i on problem instances in trainSet
      fitness(ind_i) = objValInd_i / trainSet.length
      // Select the worst individual (index denoted as k) indSetBest[k] from the indSetBest
      k ← 1
      for j = 1 to indSetBest.length do
        // If the expression of indSetBest[j] is the same with ind_i, then set k as 0 and break for loop
        if expression(ind_i) == expression(indSetBest[j]) then
          k ← 0
          break
        // If indSetBest[j] is inferior to indSetBest[k], then set the value of k as the index of j
        else if fitness(indSetBest[j]) > fitness(indSetBest[k]) then
          k ← j
        endif
      endfor
      // Update the individuals in indSetBest
      if (k != 0) && (fitness(ind_i) < fitness(indSetBest[k])) then
        Set indSetBest[k] in indSetBest ← ind_i and fitness(indSetBest[k]) ← fitness(ind_i)
      endif
    endfor
      Evolution: Generate new population by genetic manipulation
    gen ← gen + 1
    endwhile
  return indSetBest
```

4.2. The design of GP as a hyper-heuristic

This section introduces the design of GP as a hyper-heuristic for the generation of DRs to solve the DJSP with ETPC. The evolution processes of the individuals in GP is introduced first, then the parameter tuning of GP is presented.

4.2.1. Evolution processes of the individuals in GP

The main procedures for the evolution of the individuals in GP are the initial population generation, fitness evaluation, genetic manipulation of individuals, and the termination condition. The initial population is generated using the ramped half-and-half approach (Koza and NetLibrary, 1992). This approach ensures that half of the initial population are randomly generated trees with maximum allowed tree depth, and another half are randomly generated trees with depth ranging from 1 to the maximum allowed tree depth.

The initial generation evolved are a set of individuals represented as trees that can be decoded into the corresponding candidate DRs. The performance of each candidate DR is assessed by its fitness. The fitness of a DR is determined by the quality of the solution generated by the DR which is obtained by applying the DR to the problem instances in the training set. In order to measure the candidate DRs' performance properly, the selection of the training instances should reflect the critical characteristics of the problems that the evolved DRs are likely to solve in the future. As the aim of this study is to evolve effective DRs to solve the DJSP with ETPC to minimize the mean weighted tardiness of the jobs, the problem instances in the training set are nine randomly generated instances of DJSP with ETPC with the same objective function. The instances have different configurations of shop parameters to represent the various levels of difficulty. For the fitness function of GP that is used to determine the overall fitness of the DR, it is the average value of the objective values, i.e. the mean weighted tardiness of the jobs, obtained by the candidate DR on the training instances.

The next generation is evolved based on its parent generation by manipulating the individuals using genetic manipulation which includes the selection, crossover, and mutation operation. The selection method used is tournament selection. The individuals are randomly selected from the parent population and put in the tournament set until the size of the tournament set meets the specified tournament size. Then, the individual with the best fitness in the tournament set is selected and kept in the new generation. For the crossover operation, the subtree crossover is used to create new individuals for the new generation by swapping the randomly selected subtrees from two selected parent trees in the parent generation. The mutation operation is performed by subtree mutation, whereby a subtree is selected randomly from the parent tree and then it is replaced by a newly generated subtree to form a new tree in the new generation. Besides, the elitism strategy is adopted to guarantee that the optimal individuals in the parent generation can survive to the next generation. The cycle of the evolution of the new generation continues until the termination condition, which is a predefined number of generations, is met.

4.2.2. Parameter tuning of GP

To improve the performance of the proposed GPHH approach, proper parameter setting is necessary. In this study, the parameter setting for the employed GP is based on the literature survey for the parameter tuning of the algorithm and also based on empirical tests. Table 3 summarizes the detailed parameter setting of the GP used.

Most of the parameters used are modified based on the standard set of parameters used for GP as suggested by Koza and NetLibrary (1992). The initial population generation method is the same as proposed by Koza and NetLibrary (1992). The parameters used for population size, maximum allowed tree depth in initial population, selection method, the tournament size and the termination condition are based on the recommendations in Pickardt et al. (2013). Besides, a maximum tree depth is given to bound the complexity as the tree depth decides the complexity of the priority functions of the evolved DRs. Since there is no theoretical guideline on the determination of an appropriate maximum tree depth for the evolution of the priority functions of DRs (Branke et al., 2016b), the maximum tree depth after manipulation operation is set to 14 in this work as it has been shown to lead to the best results in existing study (Jakobović and Marasović, 2012).

The effects of the genetic manipulation operations on the new generation is adjusted by assigning corresponding probability to each genetic operator in the evolution process. As there is a positive correlation between the population diversity and the performance of the individuals (Branke et al., 2016b), a high rate of the crossover and mutation operations is set for the genetic manipulation on the parent population. The parameter for the elitism strategy and the genetic manipulation

Table 3
Parameter setting of GP

Parameter	Value and description
Population size	1000
Initial population generation	Ramped half-and-half approach
Maximum allowed tree depth in initial population	6
Elitism strategy	6% probability
Selection	Tournament selection, 30% probability
Tournament size	7
Crossover	Subtree crossover, 60% probability
Mutation	Subtree mutation, 4% probability
Maximum allowed tree depth after manipulation operation	14
Termination condition	Fixed number of generations, 50

operation including selection, crossover and mutation operation are set by doing some simulation experiments and evaluating the results with respect of solution quality and computational time. The specific parameter setting of the GP are as shown in Table 3.

4.3. The improvement strategies for GPHH approach

In this section, the improvement strategies for the GPHH approach are proposed to improve the efficiency of the approach for evolving effective DRs for solving the DJSP with ETPC. The improvement strategies consist of a problem-specific attribute selection for GP by incorporating the critical shop status information related to the scheduling scenario, and a mechanism for fitness evaluation in training stage by introducing the threshold condition.

4.3.1. Problem-specific attribute selection for GP

An appropriate attribute selection for the terminal set in GP is important for the successful application of the GPHH approach to evolve DRs for solving the problem under study. The individuals in GP are encoded as tree structures comprising terminals and functions. The elements in the terminal set decide the components of the job and shop attributes in the priority functions of the DRs that can be evolved. To improve the effectiveness of the evolved DRs, a problem-specific attribute selection for the terminal set which incorporates the critical shop status information related to the scheduling scenario is considered for the construction of the individuals in GP.

The job attributes are basic as they are used for prioritizing the jobs waiting in the queue of the processing machines. The inclusion of the shop attributes can also help the generated DRs to adapt better to the dynamically changing job shop conditions. Nguyen et al. (2013a) concluded that incorporating the system and machine attributes into the DRs improves the quality of the rules evolved. Hence, the exhaustive inclusion of the shop attributes in the terminal set can help to improve the capability of the evolved DRs. However, the search space grows quickly with the increase in the number of elements in the terminal set, making the approach computationally expensive. Thus, there should be a compromise between a sufficient inclusion of the attributes and the complexity of the search space.

In this study, the basic job attributes are included in the terminal set of the GP. Some important machine attributes which represent the status of the machines in the shop are also incorporated in the terminals. Besides, as the DRs are evolved for solving the DJSP with ETPC, so the information related to the jobs with ETPC is considered for the construction of DRs. Two job attributes which are sensitive to the existence of ETPC are incorporated. One attribute is related to the jobs with fore-relative operations from the job pairs with ETPC (denoted as frWJETPC). This attribute gets twice the value of the job weight with the intent to let the job have higher priority of being processed over others, and the normal job weight when the job does not have fore-relative operations from the job pairs with ETPC. The other one is the time lapse for the jobs with hind-relative operations from the job pairs with ETPC (denoted as hrTLETPC). The attribute gets the value of 0 if the job does not have hind-relative operations from the job pairs with ETPC.

In addition, some aggregate forms of the attributes, such as the relative due date (job due date minus the time scheduling decision to be made), relative operational due date (operational due date minus the time scheduling decision to be made), and non-negative slack, are used as they have been proven to be more meaningful than individual attributes. For most of the attributes, the most basic forms are provided for the hyper-heuristic to let it search for the best combinations of these attributes. Table 4 lists the elements in the terminal set of the GP.

For the function set, the most relevant functions are selected as large function set increases the search space. The function set used in this study consists of four basic mathematical operators (addition, subtraction, multiplication, and division), and two functions namely max and min. Among them, the division operator is an unprotected one which means that a very large number is returned when the divisor is 0.

Table 4
Elements in the terminal set of GP

Attribute	Notation	Description
Job Attributes	PTO	Processing time of operation
	tPTJ	Total processing time of job
	rDDJ	Relative due date of job
	rODD	Relative operational due date
	RTJ	Release time of job
	nOJ	Number of operations for the job
	atPTJ	Average total processing time of job
	WJ	Weight of job
	SJ	Slack of job
	rPTJ	Remaining processing time of job
	rAJ	Remaining allowance of job
	rnUOJ	Remaining number of uncompleted operations of job
	nxPTO	Processing time of next operation
	WTO	Waiting time of operation
	frWJETPC	Twice the value of the weight of job with fore-relative operation of ETPC
	hrTLETPC	Time lapse for the processing of the hind-relative operations with ETPC
Machine Attributes	nJMWPQ	Number of jobs in machine waiting processing queue
	aOPTWPQ	Mean operation processing time in machine waiting processing queue
	aWTWPQ	Mean waiting time of all operations in machine waiting processing queue
System Attribute	TSDM	Time scheduling decision to be made

Fig. 4 shows an example of the tree representation of a DR generated by the GP. The leaf node of the tree are the elements from the terminal set which comprises the attributes of the problem. The non-leaf node are the elements from the function set which comprises the operators. The decoding of the tree uses the in-order traversal, i.e. the operator at the root node of the tree is applied to the values obtained by recursively evaluating the left and right sub-trees. The priority function of the DR shown in Fig. 4 is WJ *(rDDJ-rPTJ).

4.3.2. Threshold condition mechanism for fitness evaluation in GPHH approach

The DRs generated by GPHH approach can be executed very fast when they are used to derive solutions to the problems. However, the evolution process of the DRs in the GPHH approach is time-consuming as the fitness evaluation of the candidate DRs on the training instances is usually computationally expensive. To improve the efficiency of the GPHH approach in its evolution process for generating candidate DRs, methods have been presented to reduce the computational time by reducing the time spent on the fitness evaluations and controlling the complexity of candidate DRs. To reduce the time on evaluation, various methods, such as duplicate detection technique to avoid the evaluation of the same candidate DRs twice (Branke et al., 2015), threshold mechanism to terminate the evaluation of inferior candidate DRs early

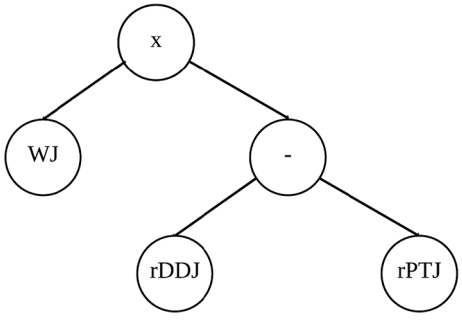


Fig. 4. Tree representation of DR in GP

(Branke et al., 2015; Hildebrandt et al., 2010), and surrogate models to approximate the fitness of candidate DRs (Hildebrandt and Branke, 2015; Zhou et al., 2020) have been employed. To control the complexity of candidate DRs, effective bloating control (Whigham and Dick, 2010) and program simplification techniques (Kinzett et al., 2009) are used.

In this research work, a threshold condition mechanism is integrated in the GPHH approach to reduce the computational time on the fitness evaluation of inferior candidate DRs in the training stage. The threshold mechanism monitors the scheduling objective value after a predefined number of completed jobs to detect inferior candidate DRs and uses the value of a probability function to decide whether to terminate the evaluation of the candidate DR or not once the threshold condition is met. Fig. 5 gives a flow chart of the threshold condition mechanism in GPHH approach for the fitness evaluation of candidate DRs in the training stage.

The threshold condition detects inferior candidate DRs based on the solution quality derived by the DRs. During the fitness evaluation of the individuals in the population, an individual, which is applied to the training instance, is judged as an inferior candidate DR if the objective value of the predefined number of completed jobs in the instance exceeds the threshold value. If the individual is an inferior candidate DR, then whether to terminate the fitness evaluation or not is decided probabilistically by a probability function. If it is terminated, then penalize it by assigning a fixed bad fitness to it. Otherwise, the fitness evaluation of the candidate DR continues until complete solution to the problem instance is obtained and the fitness of the DR is the average value of the objective values obtained by the candidate DR on training instances.

The threshold value used in the mechanism is the fitness of the worst candidate DR among the individuals in the parent generation and it is calculated as the objective value of the candidate DR on a predefined number of completed jobs in the training instance. It is to be noted that the threshold value in the initial generation is set to a very large number, then it keeps updating dynamically during the evolution throughout the generations.

The probability function decides probabilistically whether to terminate the fitness evaluation of an inferior candidate DR or not and it is given as follows.

$$p = e^{\left(\frac{-c}{x}\right)} \tag{17}$$

The probability of terminating the inferior candidate DR fitness evaluation depends on the parameter of number of generations evolved (denoted as x). When x is small in the early stage of the evolution, there

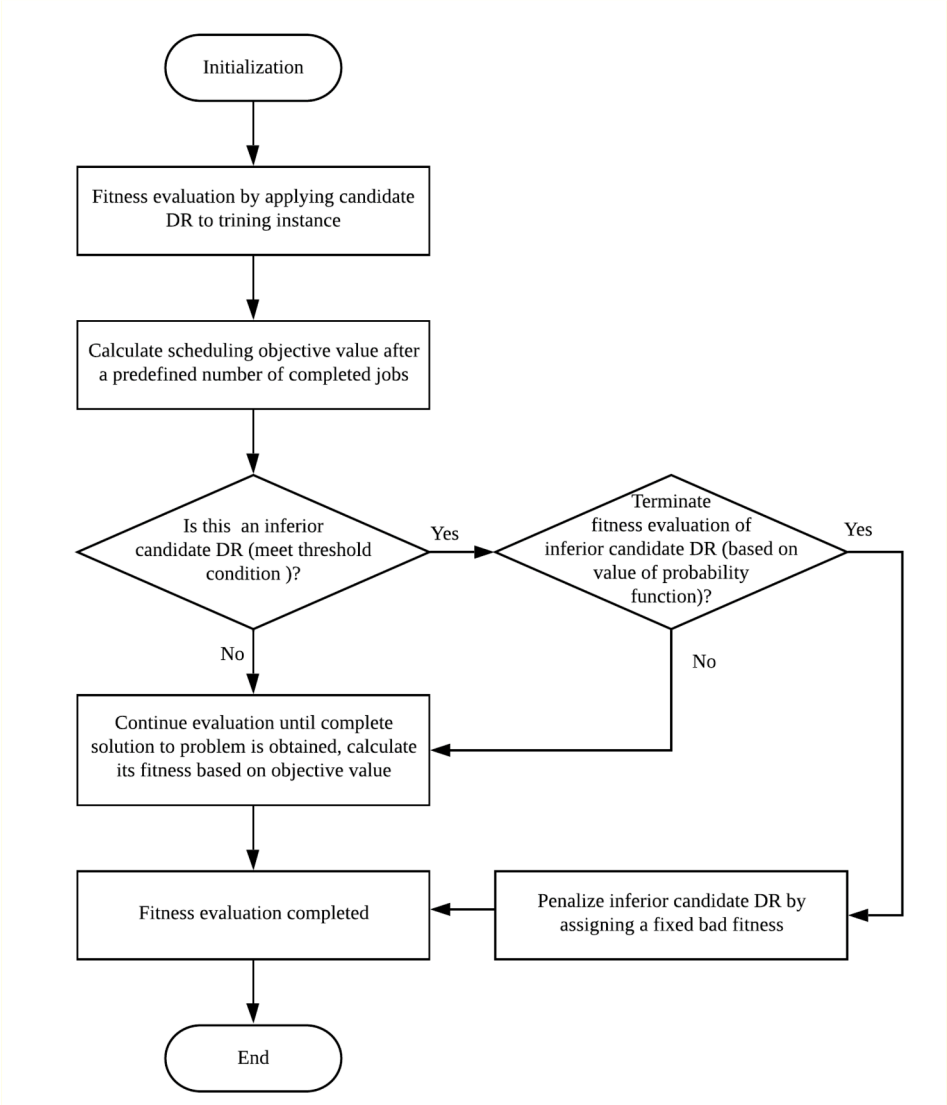


Fig. 5. Flow chart of threshold condition mechanism in GPHH approach for fitness evaluation

is less probability for terminating the fitness evaluation of inferior candidate DRs. As evolution continues and x increases, more inferior candidate DR are terminated in the fitness evaluation. Here in Eq. (17), c is a constant factor and its value is set to 5 in this study.

5. Experimental design and simulation results

The GPHH approach generates new DRs by learning from the problem instances in the training set. The effectiveness and efficiency of the evolved DRs are verified by comparing with the benchmark rules on the problem instances in the test set. In this section, the experimental design of the simulation study for the training stage and test stage are introduced, the benchmark DRs used for the comparative study are given, and the simulation results are presented.

5.1. Experimental design

5.1.1. Problem instances generation in training set and test set

The scheduling scenarios representing the various levels of difficulty in the training set and test set are randomly generated using different levels of experimental parameters. In the simulation experiment, the symmetrical balanced job shop simulation model (Hildebrandt et al., 2010; Land, 2006; Nguyen et al., 2012, 2014; Thürer et al., 2013) is used so that the ten machines in the shop have the same level of congestion over the simulation runs. During the simulation, the jobs are released to the shop intermittently over time. The parameters of the job, such as the number of operations, the operation processing times, and the routings of the job, are generated randomly. For the distributions of the job parameters used in this simulation experiment, we follow the conclusions of the parameter settings in job shop given in the surveys by Kiran and Smith (1984) and Ramasesh (1990) so that they are consistent with the job parameter settings in existing studies.

As for the arrival time of jobs, it is commonly determined in accordance with the machine utilization (Rajendran & Holthaus, 1999). In this study, the job arrival is assumed to follow a Poisson process, and the mean interarrival time of jobs is calculated as follows.

$$\bar{t}_v = \left(\bar{n}_p \times \bar{p}\right) \Big/ \left(m \times u_g\right) \tag{18}$$

Where $\bar{t}_v$ denotes the mean interarrival time of jobs, $\bar{n}_p$ denotes the mean number of operations per job, $\bar{p}$ denotes the mean processing time per operation, m is the number of machines, and u_g is the machine utilization. Giving a value of machine utilization u_g, the interarrival time of jobs can be obtained by the exponential distribution $E^{\left(1/\bar{t}_v\right)}$.

Besides, the due dates of the jobs are set by the due date assignment rule called the total work content (Baker, 1984). The value of the allowance factor is randomly selected from the set [2, 6, 8] to represent the tight, medium, and loose due-date tightness settings respectively. For the weights of the jobs, the 4:2:1 rule is employed as suggested in Pinedo and Singer (1999), so a weight of 4 is assigned to 20% of the jobs which are very important, a weight of 2 is assigned to 60% of jobs which are of average importance, and a weight of 1 is assigned to the final 20% of jobs which are of lesser importance.

In the simulation study, the information related to the pairs of jobs with ETPC and the related operations within a job pair are also generated randomly from the jobs in the simulation experiment following the ways as indicated in Xiong et al. (2017). It is to be noted that the ETPC only constrain the processing of the hind-relative operation of the latter job for a pair of two jobs with ETPC, i.e. the hind-relative operation can be processed only after its fore-relative operation has been completed for a time lapse. This means that the fore-relative operation of the former job within the pair can be processed immediately if it is available without considering the status of its hind-relative operation in the latter job when making scheduling decisions. The time lapse for the operations with extended constraints is generated randomly following a uniform distribution as indicated in Table 5. In addition, the different levels of the percentage of jobs with ETPC on the total number of jobs under the scheduling system (denoted as E_c) and the machine utilization (denoted as U_g) are considered. In this study, E_c is set to 3%, 5%, and 8%, and U_g is set to 70%, 85%, and 95% respectively to represent the various levels of difficulty.

The scheduling scenarios in the test set used for validating the performance of the DRs are set in the same way as the scheduling scenarios in the training set. Table 5 summarizes the parameter settings of the scheduling scenarios used for the training and test stage in simulation.

As shown in Table 5, two experimental factors (U_g and E_c) with three levels are used to generate nine scheduling scenarios in the training set, and the two factors with two levels are used to generate four scheduling scenarios in the test set. In the simulation experiment, the problem instances, which are used to represent the nine scheduling scenarios in the training set and the four scheduling scenarios in the test set, are randomly generated with the parameter settings given in Table 5. In the experiment, the candidate DRs in GP are trained on a training set of nine scheduling scenarios for a fixed number of generations. The best evolved DRs in the final generation are then compared with the selected benchmark rules on the simulation scenarios in the test set. Each test experiment is run 10 times in order to let the simulation results be statistically significant, and the average performance of all the rules is given in Section 5.2. It is to be noted that in the training stage, the generated problem instances representing the nine scheduling scenarios are kept unchanged in each generation in simulation for fair fitness evaluation over candidate DRs. In the test stage, the problem instances representing the four scheduling scenarios are randomly generated for the studied DRs in each replication in simulation, so different problems instances are used throughout the replications to test the DRs' performance on various scheduling scenarios.

In addition, for each replication of a scheduling scenario in simulation, the steady-state simulation is used by starting the simulation with an empty shop and the interval from the start of the simulation until the arrival of the 500th job is considered as the warm-up time to overcome initial bias in sampling. The simulation length is from the arrival of the 501th job to the next completed 2500 jobs. The statistics of the next completed 2500 jobs in the simulation length are recorded to evaluate the performance of the DRs. It is to be noted that during the steady-state simulation, the shop is continuously loaded with jobs arriving intermittently until the 2500 jobs are completed to avoid the truncation error.

5.1.2. Benchmark rules for comparative study

To verify the effectiveness and efficiency of the evolved new rules to

Table 5
Parameter setting of scheduling scenarios in training and test set

Parameter	Training set	Test set
Warm-up period	Arrival of first 500 jobs	Arrival of first 500 jobs
Simulation length	Next completed 2500 jobs	Next completed 2500 jobs
Replication	10 runs	10 runs
Number of machines	10	10
Arrival pattern of jobs	Poisson process	Poisson process
Number of operations	Integer uniform distribution [1, 10]	Integer uniform distribution [3, 6]
Operation processing times	Uniform distribution [5, 20]	Uniform distribution [5, 15]
Allowance factor	2, 6, 8	2, 6, 8
Job weight	4:2:1 rule	4:2:1 rule
Machine utilizationU_g	70%, 85%, 95%	70%, 95%
Time lapse of extended constraints	Uniform distribution [5, 15]	Uniform distribution [5, 10]
Extended constraints levelE_c	3%, 5%, 8%	3%, 8%

the problem under consideration, the comparative study is conducted to compare the performance of the evolved rules with the existing manually designed rules. Some widely used rules for optimizing the tardiness-related measure are adopted as the benchmarks. The well-known Earliest Due Date (EDD) is an optimal rule for many scheduling problems involving due dates (Mosheiov and Oron, 2004). The Modified Due Date (MDD) is reported to have considerable promise for complex production systems (Baker and Bertrand, 1982; Naidu, 2003). The Raghu & Rajendran (RR) rule (Raghu and Rajendran, 1993) shows the dynamic and global characteristics and has superior performance under tardiness-related measures as it considers the processing time, slack, and waiting time of the next operation (Rajendran and Holthaus, 1999). Hence, the above three rules serve as the benchmark rules in our study.

As the objective function of the problem under consideration is minimizing the mean weighted tardiness of the jobs, some existing DRs which incorporate the information of job weights for the production scheduling problem with similar objective functions are selected for comparative study. Parthasarathy and Rajendran (1997) proposed the Earliest Weighted Due Date (EWDD) rule by modifying the classical EDD to cover the case when jobs are associated with relative weights to minimize the mean weighted tardiness in flow shops. Kanet and Li (2004) generalized MDD to the weighted modified due date (WMDD) rule to deal with the objective of weighted tardiness. Besides, the Weighted Shortest Processing Time (WSPT) rule is a popular DR in minimizing weighted tardiness for JSP (Zhang et al., 2007). The Weighted Apparent Tardiness Cost (WATC) rule is found to be effective for weighted tardiness problems (Chen and Matis, 2013). So, the four DRs, EWDD, WMDD, WSPT, and WATC, are also employed as the benchmark rules.

Xiong et al. (2017) proposed four DRs manually to solve the DJSP with ETPC. Their simulation results suggest the four DRs have good performance on the tardiness-related measures, especially under relatively loose due date settings. Thus, the four DRs, Slack of Operation (SOP), MSOP, RR + SOP, and RR + MSOP, are also taken as the benchmark DRs for our comparative study.

In addition, some rules evolved by GP-based approach proposed in existing studies are employed as benchmark DRs in order to perform efficient comparison. Tay and Ho (2008) evolved five DRs for multi-objective flexible JSP using GP and showed the superiority of the evolved DRs over the selected rules on minimizing makespan, mean tardiness and mean flow time. Therefore, the five evolved rules ($GPTHRule_1$ to $GPTHRule_5$) from their study are used as benchmarks to compare to the best rules evolved by our GPHH framework in this study.

5.2. Simulation results

The simulation experiment is implemented using Java in Eclipse and run on a Microsoft Windows operating system with a 3.4 GHz Intel Core i5 processor and a 16 GB of RAM. To verify the effectiveness and efficiency of the new DRs evolved by GPHH approach, five best evolved DRs on the training set are selected from the population of the last generation for comparative study. The five best rules evolved by the GPHH approach with improvement strategies are denoted by $GPISRule_i$, and the expressions of their priority functions after simplifying algebraically are attached in Appendix.

In the comparative study, the five best evolved rules and the benchmark rules are used to solve the problem instances representing the various scheduling scenarios in the test set. The simulation results for the performance of the rules are presented in Table 6.

In Table 6, the yield scheduling objective value of the studied DRs under different levels of machine utilization and percentage of jobs with ETPC are presented. In addition, the post hoc Duncan's Multiple Range Test (DMRT) is used to obtain the homogeneity subset of the mean weighted tardiness of jobs for the DRs under various shop parameter settings, and the homogeneity is provided. The superscripts (a, b, and c) in Table 6 indicate the performance levels and the homogeneity of the mean weighted tardiness under various shop parameter settings. The top three levels of the DRs are marked as follows. The DRs having the best performance are marked with "a", the DRs within secondary performance level are marked with "b", and the DRs within third level are marked with "c". All the DRs belonging to a homogeneous subset correspond to the same performance level based on the DMRT.

Table 6
Mean weighted tardiness of jobs for DRs under different parameter settings

Rule	$U_g = 0.7$		$U_g = 0.95$	
	$E_c = 0.03$	$E_c = 0.08$	$E_c = 0.03$	$E_c = 0.08$
EDD	14,465^{c}	14,420^{c}	11,553 ^{bc}	11,502 ^{bc}
MDD	11,613 ^{ab}	11,174 ^{ab}	9800 ^{ab}	10,180 ^{ab}
RR	14,714^{c}	13,888 ^{bc}	11,991 ^{bc}	11,981 ^{bc}
SOP	14,051^{c}	13,940 ^{bc}	11,150 ^{bc}	11,067 ^{bc}
MSOP	11,985 ^{ab}	13,111 ^{bc}	10,208^{b}	10,333 ^{ab}
RR + SOP	14,456^{c}	14,817^{c}	11,714 ^{bc}	12,268 ^{bc}
RR + MSOP	14,570^{c}	13,951 ^{bc}	11,796 ^{bc}	12,233 ^{bc}
WATC	18,290	17,761	15,753	15,513
EWDD	16,179	16,212	12,977^{c}	13,077^{c}
WSPT	18,309	17,532	15,587	15,513
WMDD	15,576^{c}	14,964^{c}	12,946^{c}	13144^{c}
$GPTHRule_1$	14,400^{c}	14,546^{c}	12,335^{c}	11,415^{b}
$GPTHRule_2$	14,475^{c}	14,544^{c}	12,338^{c}	11,453^{b}
$GPTHRule_3$	14,507^{c}	14,623^{c}	12,339^{c}	11,424^{b}
$GPTHRule_4$	14,516^{c}	14,567^{c}	12,259^{c}	11,471^{b}
$GPTHRule_5$	15,199^{c}	14,160^{c}	12,928^{c}	12,059 ^{bc}
$GPISRule_1$	9601^{a}	9905 ^{a}	7906 ^{a}	8377 ^{a}
$GPISRule_2$	11,520 ^{ab}	11,562 ^{ab}	9614 ^{ab}	9080 ^{a}
$GPISRule_3$	11,516 ^{ab}	11,428 ^{ab}	9427 ^{ab}	9242 ^{a}
$GPISRule_4$	14,010^{c}	12,756^{b}	9565 ^{ab}	11,637^{b}
$GPISRule_5$	13,646^{c}	11,955 ^{ab}	11,293 ^{bc}	10,610 ^{ab}

It can be seen from Table 6 that the generated DRs have outstanding performance. $GPISRule_1$ has the best performance over others on the scheduling scenario for all the studied shop parameter settings. $GPISRule_2$ and $GPISRule_3$ rank second on the best performance list. $GPISRule_4$, $GPISRule_5$ are slightly inferior, but still behave well on the given performance measure.

6. Analysis and discussions of experimental results

This section covers the evaluation of the effectiveness and efficiency of the generated DRs to the problem under consideration, the sensitivity analysis of the robustness of the evolved DRs to scheduling performance measures, and the discussion of the effects of the problem model parameters on the performance of DRs.

6.1. Evaluation of effectiveness and efficiency of evolved DRs

To evaluate the effectiveness and efficiency of the evolved DRs to the problem under consideration, their performance is compared with the performance of the selected benchmark rules on the test problem instances. In the comparative studies, at first the Independent–Samples T-test is conducted at a 95% confidence level to compare the means of the scheduling objective values of the two groups, i.e. one group of generated rules and the other of benchmark rules, to determine if the associated group means are significantly different. The statistical results show that the p-value for the test to compare the two groups is less than the chosen significance level (0.05). Hence, there is a statistically significant difference in mean scheduling objective value between generated rules and benchmark rules. Further, the mean scheduling objective value for evolved rules is smaller than that of benchmark rules based on the results reported by group statistics. Therefore, the Independent–Samples T-test shows that the evolved DRs group perform better than the benchmarks group at a 95% confidence level.

Next, the one-way ANOVA is employed to compare the means of the scheduling objective values of the DRs used in the simulation experiment with the aim of determining whether there are statistically

significant differences between the performance of various DRs. The statistical results show that the p-value for the one-way ANOVA is less than the significance level of 0.05. Therefore, there are statistically significant differences between the means of scheduling objective values of different DRs.

To further understand which specific DRs are statistically significant from each other, the notched boxplot is employed to graphically depicting the means of scheduling objective values of different DRs. For the means of each DR, the value with 95% confidence interval is presented by boxplot to show its median, upper and lower quartiles, and upper and lower whiskers. The notch in the boxplot displays the confidence interval around the median. If the notches of the boxplots do not overlap, there is strong evidence that the difference between their medians is statistically significant at the 95% confidence level (Chambers et al., 1983). Fig. 6 shows the mean weighted tardiness distribution of the five DRs generated by GPHH approach and the selected benchmark rules with 95% confidence interval.

In Fig. 6, the vertical axis shows the mean weighted tardiness of jobs and the horizontal axis shows the DRs. The graph shows that the median mean weighted tardiness yield by $GPISRule_1$, $GPISRule_2$ and $GPISRule_3$ on the problem instances is smaller than that of other DRs, and the interquartile range is lower (the smaller the better) on the objective value scale in $GPISRule_1$ than in others. Besides, the range of the mean weighted tardiness for $GPISRule_1$ which is shown by the distance between the ends of the two whiskers for the boxplot does not overlap with that of other benchmark rules. This means that $GPISRule_1$ achieves better results than the others, and its confidence interval does not overlap with all the other rules, so it is considered to be the best rule among the benchmark rules and the evolved new rules for minimizing the mean weighted tardiness of jobs for the problem under consideration. Hence, the graph suggests that the best generated DR tends to perform better than all the other selected benchmark rules, i.e. the classical benchmark rules and the manually designed rules.

The results validate that the GPHH approach with improvement strategies proposed in this study can construct effective and efficient DRs to minimize the mean weighted tardiness of jobs for the DJSP with ETPC. The evolved DRs can be applied directly on practical problems as they are evolved by learning from a large set of training instances and validated on test problem instances with various shop parameter settings.

6.2. Sensitivity analysis of robustness of evolved DRs to scheduling performance measures

To evaluate the robustness of the DRs, their objective values under various scheduling performance measures as well as their homogeneity are presented in Table 7. To make a fair comparison, the results shown in Table 7 are obtained under the same parameter setting of the scheduling scenarios in test set as mentioned in Section 5. For each scheduling performance measure, the mean values of the objective values of the various DRs are given. The superscripts (a, b, and c) in Table 7 are the same as the ones used in Table 6 to indicate the top three levels of DRs.

The experimental results of the rules show that the generated rules evolved by the GPHH approach outperformed the benchmarks with respect to the objective of minimizing mean weighted tardiness as well as the objective of minimizing mean flow time and minimizing mean tardiness for the problem under consideration. Further, for the scheduling objective of minimizing maximum flow time, minimizing variance of flow time, minimizing maximum tardiness, and minimizing variance of tardiness, the generated rules yield good outcomes, suggesting that the DRs generated by the GPHH approach are effective and robust for solving the DJSP with ETPC with different scheduling objective functions. Therefore, the DRs evolved by the GPHH approach are reusable for the instances of the considered problem with different scheduling objective functions. For the effectiveness of the DRs generated by hyper-heuristic to other related problems, researchers have argument about this and there is no theoretical perspective to support the argument so far (Branke et al., 2016b). As heuristics are usually problem-specific solution methods, the evolved DRs may not have outstanding performance for problems with different scheduling environments. However, the framework of the GPHH approach proposed in this work provides a systematic approach for evolving effective problem-specific DRs for the tailored specific problem.

6.3. Discussion of effects the model parameters on the performance of DRs

The simulation study for the DJSP with ETPC in this paper considers various levels of E_c and U_g. To investigate the effects these two model parameters on the scheduling performance of various DRs, the two-way ANOVA is employed. The results is used to analyze whether there is an interaction between the level of model parameters (E_c and U_g) and the various DRs on the scheduling performance at a 5% level of significance, i.e. whether the effects of various levels of E_c or U_g on the scheduling

Fig. 6. Boxplot for mean weighted tardiness distribution with 95% confidence interval

Table 7
The performance of DRs under different scheduling objective functions

DR	Objective value of DRs under different scheduling performance measure					
	Mean flow time	Max flow time	Variance of flow time	Mean tardiness	Max tardiness	Variance of tardiness
EDD	13,218^{bc}	17,307 ^{bc}	5,134,849 ^{ab}	12,977^{c}	16,824 ^{bc}	5,053,841^{b}
MDD	10,939 ^{ab}	32,220	74,391,980	10,698 ^{a}	31,756	72,951,731
RR	13,393^{bc}	31,155	37,500,120	13,152^{c}	30,813	36,369,496
SOP	12,786^{b}	16,260 ^{ab}	4,040,509 ^{a}	12,545^{bc}	15,760 ^{ab}	3,966,072^{a}
MSOP	11,649^{b}	21,450	15,640,929	11,408^{b}	21,179	15,269,237
RR + SOP	13,558^{c}	20,970	10,709,520^{c}	13,318^{c}	20,597	10,033,933
RR + MSOP	13,377^{bc}	31,287	38,197,290	13,137^{c}	30,939	37,048,389
WATC	17,078	26,794	15,909,723	16,837	26,534	15,563,276
EWDD	16,522	25,922	18,854,799	16,281	25,576	18,796,340
WSPT	19,275	34,795	47,953,378	19,034	34,543	47,721,304
WMDD	17,169	35,968	67,282,000	16,929	35,680	66,662,478
$GPTHRule_1$	13,409^{bc}	17,023^{b}	4,864,066 ^{a}	13,168^{c}	16,904 ^{bc}	4,851,314^{b}
$GPTHRule_2$	12,995^{b}	16,872 ^{ab}	5,160,470 ^{ab}	12,753^{bc}	16,743^{b}	5,149,265^{b}
$GPTHRule_3$	13,458^{c}	17,107^{b}	5,001,580 ^{ab}	13,217^{c}	16,989 ^{bc}	4,989,522^{b}
$GPTHRule_4$	13,438^{c}	17,242 ^{bc}	5,044,466 ^{ab}	13,197^{c}	16,979 ^{bc}	5,014,756^{b}
$GPTHRule_5$	13,826	19,095^{c}	5,887,874 ^{bc}	13,585	18,648^{c}	5,720,979^{c}
$GPISRule_1$	9706^{a}	18,239^{c}	13,037,827^{c}	9501 ^{a}	17,975^{c}	12,721,733
$GPISRule_2$	11,295 ^{ab}	14,586^{a}	4,262,814 ^{a}	11,090 ^{ab}	14,257 ^{a}	4,221,714^{a}
$GPISRule_3$	11,252 ^{ab}	14,661^{a}	4,222,166 ^{a}	11,047 ^{ab}	14,240 ^{a}	4,153,688^{a}
$GPISRule_4$	12,957^{b}	19,511^{c}	13,774,222^{c}	12,752^{bc}	19,375	13,519,664
$GPISRule_5$	12,816^{b}	19,373^{c}	5,903,961 ^{bc}	12,611^{bc}	19,089	5,793,239^{c}

performance is the same for all DRs. To study the effects, the two model parameters and the DRs are taken as the independent variables; the scheduling performance, i.e. the mean weighted tardiness of jobs, is the dependent variable.

A two-way ANOVA is conducted to examine the effect of E_c level and various DRs on scheduling performance. Statistical results show that the p-value is greater than the significance level of 0.05, so there is no statistically significant interaction between the effects of E_c level and various DRs on scheduling performance, F (20, 42) = 0.062, p = 0.862. That is to say that different levels of E_c does not affect the scheduling performance of various DRs. Besides, simple main effects analysis shows that there are statistically significant differences between DRs on scheduling performance (p = 0.000), but there is no statistically significant difference between E_c levels on scheduling performance (p = 0.569).

The statistical results of the two-way ANOVA conducted for examining the effect of U_g level and various DRs on scheduling performance show that there is no statistically significant interaction between the effects of U_g level and various DRs on scheduling performance, F (20, 42) = 0.795, p = 0.705. So different levels of U_g does not affect the scheduling performance of various DRs. In addition, simple main effects analysis shows that there are statistically significant differences between DRs on scheduling performance (p = 0.000), and the differences between U_g level on scheduling performance (p = 0.000) are statistically significant.

7. Conclusion

In this paper, the DJSP with ETPC is studied to minimize the mean weighted tardiness of the jobs. The ETPC are the extension of the conventional routing-based technical precedence constraints which depict a common situation in the practical production in the mould and die manufacturing industry. The mathematical programming model of the problem under study is presented. The effectiveness of the model to the problem is verified by applying the model to solve a small-sized problem instance optimally. The presented mathematical model provides a formal description of the problem, and it contributes to both the solution and the modelling of the DJSP with ETPC.

To solve large-sized problems, the DRs which belong to the constructive heuristic are used. This paper investigated the automated design of the problem-specific DRs for solving the problem based on using GP as a hyper heuristic. The GPHH approach constructs the DRs which are evolved in the training stage and then verified in the test stage by the simulation experiments. The GPHH approach is equipped with improvement strategies to improve its efficiency when evolving effective DRs to solve the DJSP with ETPC. The strategies consist of a problem-specific attribute selection for GP by incorporating the critical shop status information related to the scheduling scenario, and a mechanism for fitness evaluation in training stage by introducing the threshold condition.

The performance of the evolved DRs is evaluated under a test set which includes a number of scheduling scenarios with different levels of difficulty. The statistical analysis of the simulation results shows that the DRs designed automatically for the problem perform better than the selected benchmark DRs and the manually designed DRs. This verified the effectiveness and efficiency of the generated DRs. The simulation results also demonstrate that the generated rules are robust under different scheduling performance measures although they are not as prominent as they were when minimizing the mean weighted tardiness. This means that the DRs evolved by the GPHH approach are reusable for the instances of the considered problem with different scheduling objective functions. So the framework of the GPHH approach proposed in this work provides a systematic approach for evolving effective problem-specific DRs for the tailored specific problem. In addition, the effects the model parameters including the percentage of jobs with ETPC and the machine utilization on the scheduling performance of the DRs are studied. The two-way ANOVO results suggest that both the two model parameter levels do not affect the scheduling performance of various DRs, but there are statistically significant differences between DRs on scheduling performance, and the differences between machine utilization level on scheduling performance are statistically significant.

Moving forward, we can study the GPHH framework with dynamic problem-oriented terminal sets and fitness functions for the specific shop configurations of the production system in response to the requirements of the current state of the system so that the DRs evolved by the GPHH approach can be more effective to lift the overall performance for the system with different characteristics. More synthesized system status attributes can also be used in the terminal set of the GP to construct more informative DRs. Further, non-scenario specific DRs can be developed and used on machines in different groups of the job shop categorized by set indicators to achieve better overall scheduling performance.

Acknowledgements

The authors are grateful to the editors and the anonymous referees for their helpful and constructive comments and suggestions, which improved the quality of this paper significantly.

Appendix

The expressions of the priority functions of the five best rules evolved by the GPHH approach are given as follows after simplifying algebraically.

$GPISRule_1$:
max{rDDJ, rPTJ + hrTlETPC } / frWJETPC
$GPISRule_2$:
rDDJ / (rPTJ * frWJETPC)
$GPISRule_3$:
SJ + PTO + hrTlETPC + tPTJ / nOJ
$GPISRule_4$:
(rDDJ – rPTJ) * WJ / frWJETPC
$GPISRule_5$:
min{atPTJ * nJMWPQ – WTO, aWTWPQ} / frWJETPC / WJ

References

Ahmadi, E., Zandieh, M., Farrokh, M., Emami, S.M., 2016. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. Comput. Oper. Res. 73, 56–66.
Allahverdi, A., Ng, C.T., Cheng, T.E., Kovalyov, M.Y., 2008. A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res. 187 (3), 985–1032.
Atlan, L., Bonnet, J., Naillon, M., 1994. Learning distributed reactive strategies by genetic programming for the general job shop problem. IEEE Press, Pensacola, Florida, USA.
Baker, K.R., 1984. Sequencing rules and due-date assignments in a job shop. Manage. Sci. 30 (9), 1093–1104.
Baker, K.R., Bertrand, J.W.M., 1982. A dynamic priority rule for scheduling against due-dates. J. Oper. Manage. 3 (1), 37–42.
Blackstone, J.H., Phillips, D.T., Hogg, G.L., 1982. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. Int. J. Prod. Res. 20 (1), 27–45.
Branke, J., Groves, M.J., Hildebrandt, T., 2016a. Evolving control rules for a dual-constrained job scheduling scenario. Paper presented at the 2016 Winter Simulation Conference.
Branke, J., Hildebrandt, T., Scholz-Reiter, B., 2015. Hyper-heuristic evolution of dispatching rules: A comparison of rule representations. Evol. Comput. 23 (2), 249–277.
Branke, J., Nguyen, S.u., Pickardt, C.W., Zhang, M., 2016b. Automated design of production scheduling heuristics: A review. IEEE Trans. Evol. Comput. 20 (1), 110–124.
Branke, J., Pickardt, C.W., 2011. Evolutionary search for difficult problem instances to support the design of job shop dispatching rules. Eur. J. Oper. Res. 212 (1), 22–32.
Brucker, P., Sotskov, Y.N., Werner, F., 2007. Complexity of shop-scheduling problems with fixed number of jobs: A survey. Math. Methods Oper. Res. 65 (3), 461–481.
Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R., 2010. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*. Springer, pp. 449–468.
Chan, F.T.S., Chan, H.K., Lau, H.C.W., Ip, R.W.L., 2003. Analysis of dynamic dispatching rules for a flexible manufacturing system. J. Mater. Process. Technol. 138 (1-3), 325–331.
Chambers, J.M., Cleveland, W.S., Kleiner, B., Tukey, P.A., 1983. Comparing Data Distributions. In: Graphical Methods for Data Analysis. Wadsworth International Group, Belmont, California, p. 62.
Chen, B., Matis, T.I., 2013. A flexible dispatching rule for minimizing tardiness in job shop scheduling. Int. J. Prod. Econ. 141 (1), 360–365.
Dimopoulos, C., & Zalzala, A. M. (1999). A genetic programming heuristic for the one-machine total tardiness problem. Paper presented at the Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on.
Dimopoulos, C., Zalzala, A.M.S., 2001. Investigating the use of genetic programming for a classic one-machine scheduling problem. Adv. Eng. Softw. 32 (6), 489–498.
Ferrell, W., Sale, J., Sams, J., Yellamraju, M., 2000. Evaluating simple scheduling rules in a mixed shop environment. Comput. Ind. Eng. 38 (1), 39–66.
Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Math. Oper. Res. 1 (2), 117–129.
Geiger, C.D., Uzsoy, R., 2008. Learning effective dispatching rules for batch processor scheduling. Int. J. Prod. Res. 46 (6), 1431–1454.
Geiger, C.D., Uzsoy, R., Aytuğ, H., 2006. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. J. Sched. 9 (1), 7–34.
Hart, E., Sim, K., 2016. A hyper-heuristic ensemble method for static job-shop scheduling. Evol. Comput. 24 (4), 609–635.
Haupt, R., 1989. A survey of priority rule-based scheduling. Oper.-Res.-Spektrum 11 (1), 3–16.
Hildebrandt, T., Branke, J., 2015. On using surrogates with genetic programming. Evol. Comput. 23 (3), 343–367.
Hildebrandt, T., Heger, J., Scholz-Reiter, B., 2010. Towards improved dispatching rules for complex shop floor scenarios: A genetic programming approach. Paper presented at the Proceedings of the 12th annual Conference on Genetic and Evolutionary Computation.
Holthaus, O., 1999. Scheduling in job shops with machine breakdowns: An experimental study. Comput. Ind. Eng. 36 (1), 137–162.
Holthaus, O., Rajendran, C., 1997. Efficient dispatching rules for scheduling in a job shop. Int. J. Prod. Econ. 48 (1), 87–105.
Hunt, R., Johnston, M., Zhang, M., 2014. Evolving less-myopic scheduling rules for dynamic job shop scheduling with genetic programming. Paper presented at the Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation.
Jakobović, D., Budin, L., 2006. Dynamic scheduling with genetic programming. Paper presented at the European Conference on Genetic Programming.
Jakobović, D., Marasović, K., 2012. Evolving priority scheduling heuristics with genetic programming. Appl. Soft Comput. 12 (9), 2781–2789.
Jayamohan, M.S., Rajendran, C., 2004. Development and analysis of cost-based dispatching rules for job shop scheduling. Eur. J. Oper. Res. 157 (2), 307–321.
Kanet, J.J., Li, X., 2004. A weighted modified due date rule for sequencing to minimize weighted tardiness. J. Sched. 7 (4), 261–276.
Kinzett, D., Johnston, M., Zhang, M., 2009. Numerical simplification for bloat control and analysis of building blocks in genetic programming. Evol. Intel. 2 (4), 151–168.
Kiran, A.S., Smith, M.L., 1984. Simulation studies in job shop scheduling - I: A survey. Comput. Ind. Eng. 8 (2), 87.
Koza, J.R., NetLibrary, I., 1992. Genetic Programming: On the Programming of Computers by means of Natural Selection. MIT Press, Cambridge, Mass.
Land, M., 2006. Parameters and sensitivity in workload control. Int. J. Prod. Econ. 104 (2), 625–638.
Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In Annals of Discrete Mathematics (Vol. 1, pp. 343-362): Elsevier.
Mosheiov, G., Oron, D., 2004. A note on the SPT heuristic for solving scheduling problems with generalized due dates. Comput. Oper. Res. 31 (5), 645–655.
Naidu, J.T., 2003. A note on a well-known dispatching rule to minimize total tardiness. Omega 31 (2), 137–140.
Nguyen, S.u., Mei, Y.i., Zhang, M., 2017. Genetic programming for production scheduling: A survey with a unified framework. Complex Intell. Syst. 3 (1), 41–66.
Nguyen, S., Zhang, M., Johnston, M., Tan, K.C., 2012. A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. Paper presented at the IEEE Congress on Evolutionary Computation.
Nguyen, S.u., Zhang, M., Johnston, M., Tan, K.C., 2013a. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. IEEE Trans. Evol. Comput. 17 (5), 621–639.
Nguyen, S.u., Zhang, M., Johnston, M., Tan, K.C., 2013b. Learning iterative dispatching rules for job shop scheduling with genetic programming. Int. J. Adv. Manufact. Technol. 67 (1-4), 85–100.
Nguyen, S.u., Zhang, M., Johnston, M., Tan, K.C., 2014. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Trans. Evol. Comput. 18 (2), 193–208.
Nie, L.i., Gao, L., Li, P., Shao, X., 2013. Reactive scheduling in a job shop where jobs arrive over time. Comput. Ind. Eng. 66 (2), 389–405.
Panwalkar, S.S., Iskander, W., 1977. A survey of scheduling rules. Oper. Res. 25 (1), 45–61.
Park, J., Mei, Y., Nguyen, S., Chen, G., Zhang, M., 2017. Investigating the generality of genetic programming based hyper-heuristic approach to dynamic job shop scheduling with machine breakdown. Paper presented at the Australasian Conference on Artificial Life and Computational. Intelligence.
Park, J., Mei, Y.i., Nguyen, S.u., Chen, G., Zhang, M., 2018. An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. Appl. Soft Comput. 63, 72–86.
Pickardt, C.W., Hildebrandt, T., Branke, J., Heger, J., Scholz-Reiter, B., 2013. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. Int. J. Prod. Econ. 145 (1), 67–77.
Parthasarathy, S., Rajendran, C., 1997. A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs-a case study. Prod. Plan. Control 8 (5), 475–483.
Pinedo, M., 2008. Scheduling: Theory, Algorithms, and Systems, 3rd ed. Springer, London.
Pinedo, M., Singer, M., 1999. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Nav. Res. Logist. 46 (1), 1–17.
Raghu, T.S., Rajendran, C., 1993. An efficient dynamic dispatching rule for scheduling in a job shop. Int. J. Prod. Econ. 32 (3), 301–313.
Rajendran, C., Holthaus, O., 1999. A comparative study of dispatching rules in dynamic flowshops and jobshops. Eur. J. Oper. Res. 116 (1), 156–170.
Ramasesh, R., 1990. Dynamic job shop scheduling: a survey of simulation research. Omega 18 (1), 43–57.
Rossi, A., Dini, G., 2007. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. Rob. Comput. Integr. Manuf. 23 (5), 503–516.
Schaller, J., Valente, J.M.S., 2019. Heuristics for scheduling jobs in a permutation flow shop to minimize total earliness and tardiness with unforced idle time allowed. Expert Syst. Appl. 119, 376–386.
Sculli, D., 1980. Priority dispatching rules in job shops with assembly operations and random delays. Omega 8 (2), 227–234.

Siebert, M., Bartlett, K., Kim, H., Ahmed, S., Lee, J., Nazzal, D., Nemhauser, G., Sokol, J., 2018. Lot targeting and lot dispatching decision policies for semiconductor manufacturing: optimisation under uncertainty with simulation validation. Int. J. Prod. Res. 56 (1-2), 629–641.

Sweeney, K.D., Sweeney, D.C., Campbell, J.F., 2019. The performance of priority dispatching rules in a complex job shop: A study on the Upper Mississippi River. Int. J. Prod. Econ. 216, 154–172.

Tay, J.C., Ho, N.B., 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Ind. Eng. 54 (3), 453–473.

Thiagarajan, S., Rajendran, C., 2005. Scheduling in dynamic assembly job-shops to minimize the sum of weighted earliness, weighted tardiness and weighted flowtime of jobs. Comput. Ind. Eng. 49 (4), 463–503.

Thürer, M., Stevenson, M., Silva, C., Land, M., Filho, M.G., 2013. Workload control and order release in two-level multi-stage job shops: An assessment by simulation. Int. J. Prod. Res. 51 (3), 869–882.

Whigham, P.A., Dick, G., 2010. Implicitly controlling bloat in genetic programming. IEEE Trans. Evol. Comput. 14 (2), 173–190.

Xanthopoulos, A.S., Koulouriotis, D.E., 2018. Cluster analysis and neural network-based metamodeling of priority rules for dynamic sequencing. J. Intell. Manuf. 29 (1), 69–91.

Xiong, H., Fan, H., Jiang, G., Li, G., 2017. A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. Eur. J. Oper. Res. 257 (1), 13–24.

Zhang, Z., Zheng, L.i., Weng, M.X., 2007. Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning. Int. J. Adv. Manuf. Technol. 34 (9-10), 968–980.

Zhou, Y., Yang, J.-J., Huang, Z., 2020. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. Int. J. Prod. Res. 58 (9), 2561–2580.