Computational and Applied Mathematics (2019) 38:185
https://doi.org/10.1007/s40314-019-0934-1

A study on PGEP to evolve heuristic rules for FJSSP considering the total cost of energy consumption and weighted tardiness

Shaqing Zhang[1] · Junrui Zhong[2] · Haidong Yang[3] · Zhantao Li[3] · Guosheng Liu[1]

Received: 20 October 2018 / Revised: 19 May 2019 / Accepted: 24 September 2019 /
Published online: 10 October 2019

Abstract
Performance indicators such as makespan, flow time and tardiness are considered to be optimisation objectives in the traditional flexible job shop scheduling problem (FJSSP). However, the cost of energy consumption or environmental problems should not be ignored. This paper addresses the FJSSP by minimising the sum of the cost of energy consumption and the weighted tardiness. First, a mathematical model of the problem and a heuristic algorithm for the problem are presented. Second, a parallel gene expression programming (PGEP) method with a migration scheme is put forward to evolve rules for the proposed heuristic algorithm to solve the problem. To speed up the system learning process, a parallel and distributed computing framework is also designed. Finally, the performance of the proposed PGEP approach is evaluated through extensive simulations. The time-of-use electricity pricing, due date tightness and tardiness penalty weight are considered when evaluating the effect of the heuristic rules. Experimental results show that the proposed PGEP approach can significantly improve the quality of the heuristic rules, and the PGEP-evolved rules can fast and effectively solve FJSSP.

Keywords Job shop scheduling · Flexible manufacturing · Heuristics · Simulation · Parallel gene expression programming (PGEP) · Energy consumption cost

Mathematics Subject Classification 90-08 · 90B30 · 90B35

Communicated by Hector Cancela.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s40314-019-0934-1) contains supplementary material, which is available to authorized users.

✉ Junrui Zhong
ecis@163.com

[1] School of Management, Guangdong University of Technology, Guangzhou 510520, China

[2] The First Affiliated Hospital of Jinan University, Guangzhou 510630, China

[3] School of Electromechanical Engineering, Guangdong University of Technology, Guangzhou 510006, China

1 Introduction

Energy is one of indispensable basic resources for manufacturing. With the global development of industry, the amount of world's energy consumption has been doubled in the last 50 years. At present, industry consumes about half of the world's energy (Mouzon et al. 2007). The rising costs of energy are one of the most vital factors for the development of manufacturing enterprises. In the US, about 34% of all energy is consumed by the industrial sector. Electricity in industrial consumption accounts for a tenth in total energy consumption (Energy Information Administration 2005). Energy costs for the manufacturers were annually 100 billion dollars in 2006, which is even larger today (Solar Energy International 2015). In China, the manufacturing industry just completes about 21.5% of the global manufacturing tasks annually, but it consumes an equivalent of 18 billion tons of standard coal. In the US, the manufacturing sector consumes about one-third of the global energy, contributing to about 28% of greenhouse gas emissions (Mouzon 2008). In China, carbon emissions from energy consumption in manufacturing accounts for approximately 60% of total industrial carbon emissions (Ya 2013). Electricity is one of the most important forms of energy for manufacturing. Emissions of pollutants and greenhouse gases from electricity generation account for a significant portion of world's greenhouse gas emissions. For example, every year in China, around 50% of the electricity is consumed by manufacturing (Tang et al. 2006), and at least 26% of carbon dioxide emission is generated (Liu et al. 2014). With the increasing energy price, energy shortage and environment deterioration, more and more manufacturing companies are forced to reduce energy consumption as well as environmental pollution to save energy costs and become more environmentally friendly.

At present, research on minimising the energy consumption of manufacturing system is divided into three categories: the machine level, the product level and the manufacturing system level. First, from the machine level perspective, many researchers have tried to develop and design more energy-efficient machine equipment to reduce power and energy demands of machine components (Duflou et al. 2012; Fang et al. 2011a; Li et al. 2011; Mori et al. 2011; Neugebauer et al. 2011). In the field of discrete part manufacturing, reviews on the state-of-the-art technologies that increase energy and resource efficiency are provided by Duflou et al. (2012) and Fysikopoulos et al. (2013). However, in contrast with over 30% of input energy demand for background of machining (i.e., spindle, jog, coolant pump, computers and fans, etc.), the energy requirement for the active removal of material is quite small (Dahmus and Gutowski 2004; Drake et al. 2006). Most energy is consumed by functions that have no direct relevance to the production of components (Gutowski et al. 2005). Second, from the product design perspective, some existing research has focused so far on the modelling framework for embodied energy of a product (Rahimifard et al. 2010; Seow and Rahimifard 2011; Kara et al. 2010). However, to facilitate the analysis and evaluation of the embodied energy of a product, powerful commercial simulation software needs to be developed, which requires enormous capital investment. It cannot be easily applied to most of the manufacturing companies, especially to those small- and medium-sized enterprises (SMEs). Third, from the manufacturing system level perspective, an operational decision method is used to optimise shop floor planning and scheduling and decision strategies, which is relatively low cost and applicable to existing systems compared to the first two categories. This suggests that energy saving on system level may have a significant (perhaps a bigger) opportunity than solely on updating individual machines or processes (Fang et al. 2011a). So it can be employed as a promising energy saving approach.

Facing the challenge from an intensely competitive market, more and more companies, especially SMEs, have adopted flexible layouts of workshops to realise the transformation from traditional large-scale continuous production to modern small-scale discrete production with multispecies. However, to the best knowledge of the authors, most of the current energy-conscious scheduling research focuses on single machine, parallel machine and flow shop. A typical job shop type of manufacturing system has not been well investigated from the perspective of energy consumption reduction (Liu et al. 2014). For example, in the mould manufacturing industry, operations such as milling, drilling, boring and cutting, consume a large quantity of electricity while these operations can be usually processed on different machines with different processing time. At the same time, the mould enterprise usually takes on manufacturing tasks of dozens or even more than one hundred sets of moulds but with limited equipment, delay delivery with a large amount of penalty may be impossible to be avoided.

The remainder of the paper is organized as follows. In Sect. 2, a literature review is presented. In Sect. 3, the mathematical problem is formally defined. Then, in Sect. 4, the presented heuristic algorithm for solving the FJSSP is described. In Sect. 5, the proposed PGEP approach to evolve scheduling rules (SRs) for the heuristic algorithm is elaborated. Section 6 is dedicated to a computational study for evaluating the feasibility of the model and the effectiveness of the method. Finally, conclusions and further research subjects are presented in Sect. 7.

2 Literature review

As mentioned in Sect. 1, the operational decision method is regarded as a feasible and effective approach for manufacturing companies to reduce the total cost of energy consumption. The research on floor shop scheduling with the objective of reducing the total cost of energy consumption is relatively less but increasing.

The approach which breaks down the total energy use of machining processes has been employed as the bases for modelling power input of machine tools at the workshop level. According to this approach, electricity consumption for a machine tool in a feasible schedule can be divided into two types: non-processing electricity consumption (NPE) and processing electricity consumption (PE). NPE includes a machine's starting up, shutting down and idling. JPE is defined as job-related processing electricity consumption (JPE) which means electricity is consumed when a job is processed on a specific machine. Thus, PE is the sum of all the JPE on a specific machine (Liu et al. 2014).

From the workshop scheduling perspective, sequencing, turning off/turning on and process route selection (PRS) are considered as typical operational decision methods which are used for electricity saving. Mouzon adopts the sequencing method to reduce the total NPE in both single machine environment and parallel machine environment (Mouzon 2008). Dai et al. propose an energy-efficient model for flexible flow shop scheduling (Dai et al. 2013). A sequencing method based on genetic-simulated annealing algorithm is adopted to make a significant trade-off between the makespan and the total energy consumption (including PE and NPE) to implement feasible scheduling. Fang et al. address a scheduling problem in a multiple-machine system where the computing speed of the machines is allowed to be adjusted during the course of execution (Fang and Lin 2013). A particle swarm optimisation (PSO) algorithm is adopted to optimise the decision by allocating the jobs to the machines as well as determining the job sequence and processing speed of each machine with the objective function involving the total weighted job tardiness and the power consumption (PE) cost. Fang

et al. present a new mathematical programming model of the flow shop scheduling problem considering peak power load, energy consumption (including NPE and PE), and associated carbon footprint in addition to the cycle time (Fang et al. 2011b). Apart from the processing order of the jobs, the operation speed is also considered as an independent variable in the proposed scheduling problem, which can be changed to affect the peak load and energy consumption. Zhang et al. develop a time-indexed integer programming formulation to identify manufacturing schedules that minimise electricity cost (PE) and the carbon footprint under time-of-use tariffs without compromising the production throughput. And the approach is demonstrated using a flow shop with eight process steps to operate on a typical summer day (Zhang et al. 2014). To deal with the production and energy efficiency of the unrelated parallel machine scheduling problem, Moon et al. focus on two optimisation objectives: minimising the makespan of production and minimising the time-dependent electricity costs (PE) (Moon et al. 2013). A hybrid genetic algorithm with a blank job insertion algorithm is proposed and its performance is demonstrated in simulation experiments. Zeng et al. present a PSO-based approach to solve the dynamic scheduling problem of multi-task for hybrid flow shop with the objective of minimising the energy consumption (PE) (Zeng et al. 2009). According to the characteristic of hybrid flow shop scheduling problem (HFSP) in practice, Liu et al. present a mixed integer nonlinear programming model for HFSP to minimise the energy consumption (including NPE and PE), and develop an improved genetic algorithm to solve the problem (Liu et al. 2008). Liu et al. address a fuzzy flow shop scheduling problem in a production system where the machine setup times depend on their prior states, and develop an enhanced genetic algorithm to minimise the energy consumption and tardiness penalty (Liu et al. 2017). Mokhtari et al. present an energy-efficient scheduling problem in a flexible job shop floor industrial environment and design an improved simulated annealing genetic algorithm to solve the multi-objective optimization problem (Mokhtaria and Hasani 2017). Yin et al. propose a new low-carbon mathematical scheduling model for the flexible job shop environment to optimise productivity, energy efficiency and noise reduction, and develop a multi-objective genetic algorithm based on a simplex lattice design to solve the model (Yin et al. 2017).

Each idle period of a machine and the total amount of the idle time are closely related to the order of jobs which are processed on that machine (Liu et al. 2014). Large quantities of energy is consumed when non-bottleneck machines are lying idly and significant amounts of energy are consumed when a machine is turned on or turned off (Drake et al. 2006). So for electricity saving, a machine tool can be turned off when it becomes idle (Mouzon 2008). Based on the previous work, more and more research about turning on/turning off methods has been conducted. For example, by scheduling a machine's starting up and shutting down, Chen et al. investigate energy consumption reduction in serial production lines with finite buffers and machines having Bernoulli reliability mode (Chen et al. 2013). Shrouf et al. propose a mathematical model to minimise the total cost of energy consumption for a single machine production scheduling during production processes. The genetic algorithm has been utilized to optimise the sequencing of job processing and switching of machines (turning on/turning off) (Shrouf et al. 2014). Zanoni et al. analyse the effects of energy cost in production on the optimal production policy of a two-stage production system with controllable production rates. They propose two different classes of policies, namely continuous and interrupted batch production. In the former case, the first machine produces continuously until the end of the production cycle, while in the latter case, the first machine will turn into an idle state after a batch has been shipped to the second stage, and it is turned on again when it is necessary to refill the buffer stock before the second stage. In addition, they suggest different policies for each class by assuming that both machines may either stay idle or be switched off and on between two production cycles (Zanoni et al. 2014).

PRS is widely used in flexible workshop scheduling problems with traditional optimisation objectives, such as makespan, tardiness, and cost. With regard to reducing the total cost of energy consumption, the related research is relatively less. Mouzon adopts the PRS to reduce total PE as well as total NPE for a parallel machine scheduling problem (Mouzon 2008). According to He et al., energy consumption dynamically depends on the flexibility and variability of task flow in production processes, and the PRS can be adopted to decrease both total PE and total NPE in a flexible job shop environment. The PRS is only effective in systems which have alternative routes with different energy characteristics for the same job (He et al. 2012).

Based on the above literature review, several observations can be obtained. First, from the workshop scheduling perspective, most research on reducing the total cost of energy consumption by employing operational decision method has focused on the single machine environment, the parallel machine environment, or the flow shop environment. However, few references concentrate upon the job shop environment. Especially, the literature on energy-efficient flexible job shop scheduling is relatively rare. Second, from the perspective of optimisation techniques, methods or tools of reducing the total cost of energy consumption mainly include dispatching rules (Mouzon et al. 2007), a greedy randomised adaptive search procedure (Mouzon 2008), constraint programming algorithm (Moon and Park 2014), mixed integer programming solver (Moon and Park 2014; Bruzzonea et al. 2012), genetic algorithm (Dai et al. 2013; Moon et al. 2013; Liu et al. 2017; Mokhtaria and Hasani 2017; Yin et al. 2017) and PSO (Fang and Lin 2013; Zeng et al. 2009). However, different environments and characteristics of problems limit the applicability of these methods to a wider range of production scheduling for reducing energy consumption. So it is of considerable significance for both academia and industry to explore the flexible job shop model considering reducing the energy consumption and its related scheduling optimisation methods.

In this paper, derived from a cooperative enterprise which produces a variety of metal moulds, a FJSSP is proposed from the system level perspective to minimise the sum of the total weighted tardiness penalty (TWTP) of all jobs and the total energy cost (TEC). The modelling method for this problem is presented as below.

3 Problem definition and notations

The FJSSP is prevalent in manufacturing industry, especially in SMEs. In our research, the FJSSP is the static one which means the number of jobs is deterministic and all of them are available at time zero. The objective function for the FJSSP can be expressed as below:

$$\text{minimise} F(s) = \text{TWTP}(s) + \text{TEC}(s) s \in S \tag{1}$$

The formal mathematical definition of the problem has been described in detail in the following sections.

3.1 Notations

The following notations will be used for problem statement throughout the paper.

Indices and sets

J	The set of jobs
i, i'	Jobs ($i, i' \in J$)

O	The set of operations
O_i	A finite set of ordered operations of job i
j, j'	Operations ($j, j' \in O$)
M	The set of machines
M_j	The set of alternative machines on which operation j can be processed,$M_j \subseteq M$
$M_j \cap M_{j'}$	The set of machines on which operations j and j' can be processed
k	Machines ($k \in M$)

Parameters

d_i	Due date of job i
g_i	The number of operations of job i
w_i	Weighted importance of job i, which indicates the tardiness penalty degree. Although different jobs have the same delay time, the tardiness penalty cost may vary for these jobs
S	A finite set of all feasible schedule plans
s	A feasible schedule plan,$s \in S$
L	A very large number
$cpup$	Cost of per unit power
utp	Unit of tardiness penalty
O_{ij}	Operation j of job i
O'_{rk}	r-th operation processed on machine k within a feasible schedule s
S'_{rk}	Starting time of O'_{rk} on machine k
$C_i(s)$	Completion time of job i in schedule s
C'_{rk}	Completion time of O'_{rk} on machine k
$T_i(s)$	Tardiness of job i, defined as $T_i(s) = \max\{0, C_i(s) - d_i\}$
t_{ijk}	The cutting time of operation O_{ij} on machine k
P_k	Input power of machine k
p_{ijk}	The processing time of operation O_{ij} on machine k
P_k^{idle}	Idle power of machine k
$P_{ijk}^{runtime}$	The increase in power based on P_k^{idle} when executing the runtime operations for processing O_{ij} on machine k
P_{ijk}^{cutting}	The increase in power based on P_{ijk}^{runtime} when actually executing cutting for O_{ij} on machine k
E_{ijk}	JPE of O_{ij} on machine k
E_{ijk}^{idle}	Electricity consumed by machine k with idle power level during p_{ijk}
E_{ijk}^{runtime}	Electricity consumed by machine k when it executes the runtime operations for processing O_{ij}
E_{ijk}^{cutting}	Electricity consumed by machine k when it actually executes cutting for O_{ij}

Decision variables

C_i	The completion time of job i
C_{ijk}	The completion time of operation O_{ij} on machine k
$C\text{max}$	Maximum completion time of all jobs (makespan)
S_{ijk}	The starting time of operation O_{ij} on machine k
X_{ijk}	1, if operation O_{ij} is processed on machine k; 0, otherwise
$Y_{iji'j'}$	1, if operation O_{ij} precedes operation $O_{i'j'}$ on machine k; 0, otherwise

3.2 The TWTP of flexible job shop model

Flexible job shop scheduling problem is formally defined as FJSSP according to the work of Özgüven et al. (2010) and Driss et al. (2015). A finite set of n independent jobs is processed on a finite set of m machines M. A number of g_i ordered operations $(O_{i1},O_{i2},\ldots,O_{i,g_i})$ is performed to complete the job i. The operation j of job i can be processed by any machine in a given set $M_j \subseteq M$ for a given processing time p_{ijk}. Each job has a due date d_i as well as a weighted importance factor w_i. The FJSSP is a routing and sequencing problem which means that each operation O_{ij} is assigned to a selected machine from the set M_j and operations on the machines are sequenced to optimise one or more objectives. All jobs are available at time zero. A machine can process only one operation at a time, and no pre-emption is allowed. In this paper, we will construct a feasible schedule s of all jobs so as to minimise the TWTP $(s) = \text{utp} \times \sum_{i=1}^{n} w_i \times \max\{0, C_i(s) - d_i\}$ with the following constraints:

$$\sum_{k \in M_j} X_{ijk} = 1, \forall i \in j, j \in O_i \tag{2}$$

$$S_{ijk} + C_{ijk} \leq X_{ijk} \times L, \ \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{3}$$

$$C_{ijk} \geq S_{ijk} + p_{ijk} - \left(1 - X_{ijk}\right) \times L, \ \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{4}$$

$$S_{ijk} \geq C_{i'j'k} - Y_{iji'j'} \times L, \ \forall i < i', \forall j \in O_i, \forall j \in O_i, \forall k \in M_j \cap M_{j'}, \tag{5}$$

$$S_{i'j'k'} \geq C_{ijk} - \left(1 - Y_{iji'j'}\right) \times L, \ \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'}, \tag{6}$$

$$\sum_{k \in M_j} S_{ijk} \geq \sum_{k \in M_j} C_{i,j-1,k}, \ \forall i \in J, \forall j \in O_i - \left\{O_{i,1}\right\}, \tag{7}$$

$$C_i \geq \sum_{k \in M_j} C_{i,O_{i,g_i},k}, \forall i \in J, \tag{8}$$

$$C_{max} \geq C_i, \forall i \in J, \tag{9}$$

where

$$X_{ijk} \in \{0, 1\}, \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{10}$$

$$S_{ijk} \geq 0, \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{11}$$

$$C_{ijk} \geq 0, \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{12}$$

$$Y_{iji'j'k} \in \{0, 1\}, \ \forall i \in i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j,$$

$$\forall i \in i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j, \tag{13}$$

$$C_i \geq 0, \forall i \in J. \tag{14}$$

Constraints (2) ensure that operation O_{ij} is assigned to only one machine. If the operation O_{ij} is not allocated to machine k, the constraints (3) set the starting time and completion time of it on machine k equal to zero. Otherwise, the constraints (4) make sure that the difference between the starting time and completion time is equal in the least to the processing time on machine k. If O_{ij} has not been processed on machine k, the symbol L is used to indicate the unpredictable completion time of O_{ij}. Constraints (5) and (6) require that operation O_{ij} and operation $O_{i'j'}$ cannot be done at the same time on any machine in the set $M_j \cap M_{j'}$. Constraints (7) guarantee that the precedence orders among the operations of a job are not violated, i.e., the operation O_{ij} cannot start until its previous operation $O_{i,j-1}$ is completed.

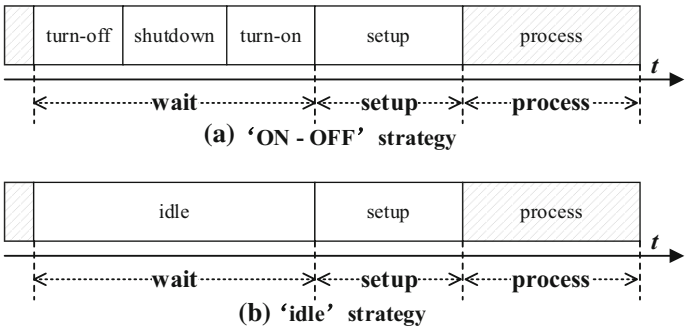

Fig. 1 Machine state transition for two different strategy

Constraints (8) determine the completion time (of the final operation defined in terms of number) of the jobs, and constraints (9) determine the makespan which limits the maximum completion lead time (of all jobs) to be greater than or equal to the completion time of job i. Finally, the other constraints (10), (11), (12), (13) and (14) denote the conditions on decision variables.

3.3 The TEC of electricity consumption model

At any time, each machine can only be in one of six states, namely, turn off, shutdown, turn on, setup, process, idle. These states can be divided into three phases, that is, waiting, setup, and process. Figure 1 depicts the two alternative state transition strategies for machine in the waiting phase. Indicators for determining the optimal strategy during the waiting phase are applied by some researchers (Mouzon et al. 2007; Dai et al. 2013), so we will not address this topic here. In this paper, 'idle strategy' is employed. Both the setup time and setup energy are negligible.

The electricity consumption model is based on the existing research work on environmental analysis of machining (Liu et al. 2014; Diaz et al. 2010). In the waiting phase, the idle power of machine k is defined by P_k^{idle}, while in processing phase, each machine k has two constant levels of power consumption: during the runtime mode and when carrying out the actual operation, presented as cutting operation in this paper. The increase in power during runtime is given by P_{ijk}^{runtime}, and the further additional power requirement for cutting is expressed as P_{ijk}^{cutting}. The whole processing time p_{ijk} can be defined as the time interval between coolant switching on and off. The t_{ijk} often has a slightly shorter time interval during which the highest power level $P_k^{\max}$ is required, and the $P_k^{\max}$ is defined as: $P_k^{\max} = P_k^{\text{idle}} + P_{ijk}^{\text{runtime}} + P_{ijk}^{\text{cutting}}$.

The job-related processing electricity consumption (JPE) required to accomplish operation O_{ij} on machine k is E_{ijk}, which is given as follows: $E_{ijk} = E_{ijk}^{\text{idle}} + E_{ijk}^{\text{runtime}} + E_{ijk}^{\text{cutting}} = P_k^{\text{idle}} \times p_{ijk} + P_{ijk}^{\text{runtime}} \times p_{ijk} + P_{ijk}^{\text{cutting}} \times t_{ijk}$. Because of the flexibility of the research problem, E_{ijk} is a variable which is affected by different scheduling plans. Thus, the processing electricity consumption required for completing s is $\text{PE}(s) = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{g_i} (E_{ijk} \times X_{ijk})$, which is the sum of all the JPE on a specific machine.

Similarly, the NPE of machine k, defined by $NPE_k(s)$ (only including the idle power consumption of machine k) is also a function of the scheduling plan, which can be calculated as follows:

$$\mathrm{NPE}_k(s) = P_k^{\mathrm{idle}} \times \left(\max\left(C_{rk}^{'}\right) - \min\left(S_{rk}^{'}\right) - \sum_{r} (C_{rk}^{'} - S_{rk}^{'}) \right). \tag{15}$$

By carrying out a feasible schedule s of all jobs, we obtain the amount of total NPE by $\mathrm{NPE}(s) = \sum_{k=1}^{m} \mathrm{NPE}_k(s)$. Therefore, the total energy cost is given as, $\mathrm{TEC}(s) = (\mathrm{PE}(s) + \mathrm{NPE}(s)) \times \mathrm{cpup}$.

4 Heuristic algorithm for solving FJSSP

It is well known that it is difficult to find the optimal solution to the FJSSP due to its inherent complexity. Thus, we propose a heuristic algorithm to solve the FJSSP.

Because of its flexibility, we will decompose the FJSSP into two sub-problems: the routing problem and the sequencing problem, where the routing problem means that each job is assigned to an appropriate machine for waiting to be processed, while, the sequencing problem means that operations on the machines are prioritised and then executed.

Except the notations defined earlier, the ones to be used in the heuristic algorithm are defined as follows:

CT	Current time
$\emptyset$	Empty set
Φ	The set of machines that is idle at current time
Ω	The set of operations unprocessed
Y	The set of jobs that have not been scheduled
Z	A temporary variable (a set of jobs)
M_{ij}	The set of alternative machines on which operation O_{ij} can be processed, $M_{ij} \subseteq M$
OW_k	The set of operations waiting for processing by machine k.

4.1 Heuristic for FJSSP

In this subsection, we will construct a heuristic algorithm for the FJSSP to find a feasible solution. The algorithm flow chart is illustrated in Fig. 2.

Step 1 Initialise variables:$\mathrm{CT} = 0$, $Y = J$, $\Omega = O$, $\varphi = M$;
Step 2 If $Y = \emptyset$, the algorithm is finished; otherwise, go to Step 3;
Step 3 Set$Z = Y$;
Step 4 If $Z = \emptyset$, go to Step 5; otherwise randomly select a job i from set Z, and then execute the following procedure:

1. If there is no operation of job i being executed at current time, the operation of job i that can be processed currently is selected and defined as O_{ij}; otherwise,$Z = Z \backslash i$, go to Step 4;
2. IfO_{ij} has been assigned to a machine,$Z = Z \backslash i$, go to Step 4;
3. Calculate priority values for all the machines in set M_{ij} according to a MAR;
4. Choose the machine k whose priority value is max; add O_{ij} to the operation set OW_k, $Z = Z \backslash i$, go to Step 4.

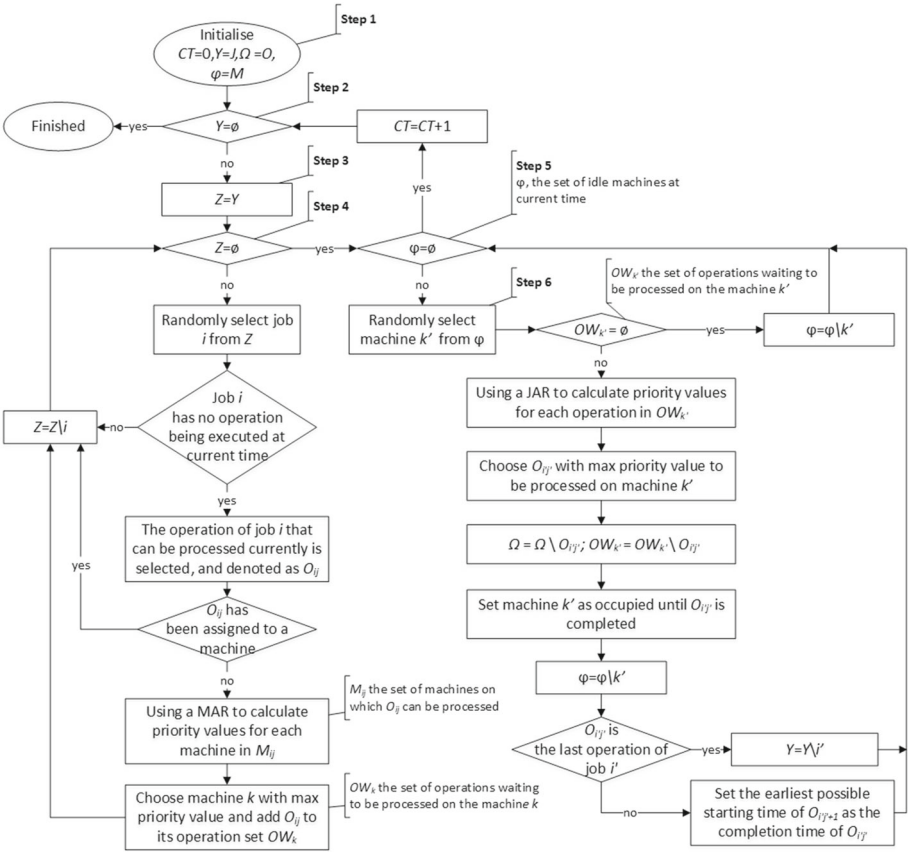

Fig. 2 Workflow of heuristic algorithm for FJSSP

Step 5 If $\varphi = \emptyset$, CT = CT + 1, go to Step 2; otherwise, go to Step 6;

Step 6 Randomly choose a machine k' from set φ and determine the processing operation for machine k' by performing the following steps:

1. If $OW_{k'} = \emptyset$, $\varphi = \varphi \backslash k'$, go to Step 5; otherwise, executes the following procedure (2)-(6);
2. Calculate priority values for all the operations in set $OW_{k'}$ according to a JAR;
3. Choose the operation $O_{i'j'}$ with max priority value to be processed on machine k', $OW_{k'} = OW_{k'} \backslash O_{i'j'}$, $\Omega = \Omega \backslash O_{i'j'}$;
4. Set the status of machine k' as occupied until $O_{i'j'}$ is completed,$\varphi = \varphi \backslash k'$;
5. If $O_{i'j'}$ is the last operation of job i',$Y = Y \backslash i'$; otherwise, set the earliest possible starting time of $O_{i',j'+1}$ as soon as $O_{i'j'}$ is completed;
6. Go to Step 5.

In above procedure of the heuristic algorithm, the machine allocation rule (MAR) and the job allocation rule (JAR) are employed as SRs for the routing problem and the sequencing problem, respectively. As for SRs, we can refer to Su et al. (2013), Doh et al. (2013), Gema and Rafael (2014) and Melo and Ronconi (2015), etc. However, there are few papers to construct universal SRs which can dominate all FJSSPs with different workshop features, constraint

conditions and optimisation objectives. Most research on SRs for the FJSSP only focuses on how to select a given rule from a number of candidates, rather than exploring and discovering a new and potentially more effective rule (Nie et al. 2013a). However, a better solution to the FJSSP should not be ignored. To explore efficient rules for a given scheduling environment for the FJSSP may be a significant and interesting job. Some researchers have gradually paid more attention to automatically construct SRs based on artificial intelligence approaches, such as, Tay and Ho employed suitable parameters and operator spaces to evolve composite dispatching rules which are generated by GP (genetic programming) framework (Tay and Ho 2008). Nie et al. proposed a GEP-based (gene expression programming) approach which automatically constructs reactive scheduling policies for the dynamic FJSSP with job release dates, and its effectiveness is verified by simulation experiments (Nie et al. 2013b). However, the GP or GEP-based artificial intelligence approach usually costs a lot of computation time which limits its wide use.

Based on the above discussion, we propose a PGEP method to quickly and automatically construct SRs in the given FJSSP environment in the following sections.

5 PGEP approach to evolve SRs

GEP, as an evolutionary algorithm which was first proposed by Ferreira (2001), uses populations of individuals, selects them in terms of fitness and introduces genetic variation by using one or more genetic operators with a fixed length and linear strings of chromosomes (genome) representing expression trees (ETs) of different shapes and sizes (phenome).

Based on these previous researches, the flow of PGEP is proposed and then the implementation of PGEP to evolve SRs for FJSSP with optimising objective of minimising the sum of TWTP and TEC is stated in details.

5.1 Flow of PGEP

As shown in Fig. 3, the flow of PGEP comprises two modules, parallel learning module and distributed simulation module. In the parallel learning module, a set number of CPU threads all start at the same time, each thread initialises and then autonomously performs evolution of a population. Each population is comprised of a number of chromosomes (i.e., candidate SRs) and each chromosome has fixed length genes that are randomly generated. All chromosomes in populations are sent to a master server of distributed simulation module via network, and the master server distributes chromosomes to a set number of slaves. Each slave simulates a production environment and evaluates the fitness value using a quantitative performance measure for chromosomes. Then, the fitness value of each chromosome passes back from slave to the master and finally back to the parallel learning module. The optimal fitness for all chromosomes in each population (i.e., each population's optimal fitness) is picked out by the main thread of parallel learning module, respectively, and they are compared with each other in pairs. Some individuals in the population whose currently best fitness is preferable to the other one are migrated to the latter from which the same number of worst individuals are removed. After that, for each population, its next population of chromosomes is formed through reproducing and modifying its current excellent individuals using evolutionary search operators such as selection with elitism strategy, replication, mutation, transposition and recombination. One of the next populations of individuals is then evaluated again by distributed simulation module, simultaneously with the other populations. This cycle is repeated until the termination condition is satisfied.

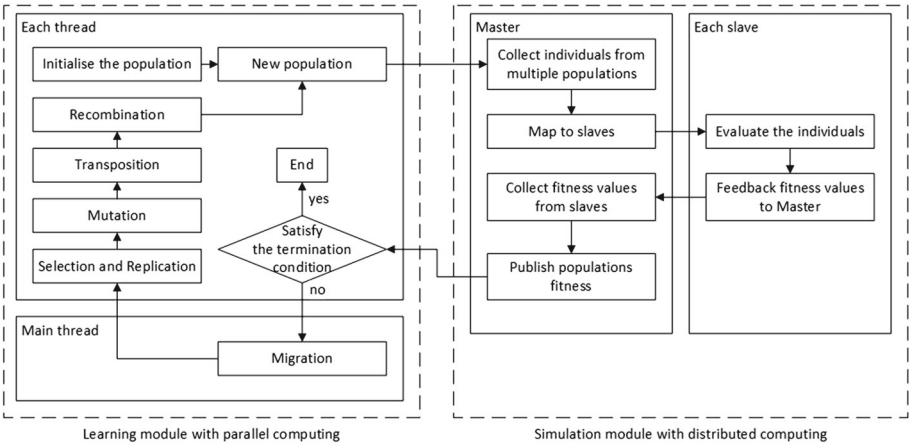

Fig. 3 PGEP workflow

5.2 Application of PGEP to evolve SRs for FJSSP

5.2.1 Designing of FS and TS

The most essential thing required for implementing PGEP is to design appropriate functions set (FS) and terminal set (TS) relevant to a particular problem domain (i.e., FJSSP with optimising objective of minimising the sum of TWTP and TEC in this paper). Each chromosome in PGEP is constructed using the elements from FS and TS and modified during parallel evolutionary progress. A priori is defined by the set of available elements which are used to discover possible solutions to the problem (Nie et al. 2013a). Therefore, choosing the proper elements for FS and TS is vital for the implementation of optimization process. The FS and TS used to construct SRs in PGEP are defined in Table 1. TS can be divided into two subsets, TS-R and TS-S. TS-R is adopted to be combined with FS to construct MARs for the routing problem, while TS-S is combined with FS to construct JARs for the sequencing problem.

It should be noted that in the FJSSP an operation of a job can be processed on any machine in a determinate set of machines. Therefore, the average processing time of O_{ij} is defined as p_{ij}^{avg}, which is given as follows: $p_{ij}^{\text{avg}} = \left(\sum_{k \in M_j} p_{ijk}\right) \Big/ h_{ij}$. If O_{ij} has been completed at current time, then the remaining processing time (pl, in Table 1) of job i is calculated as the following formula: $\text{pl} = \sum_{h=j+1}^{g_i} p_{ij}^{\text{avg}}$. The remaining cutting time of job i is calculated in the same way.

5.2.2 Mapping mechanism between chromosomes in PGEP and SRs

Generally, a chromosome comprises one or more genes, each of which consists of a symbolic string with fixed length selected form FS or TS. Since two types of problems (routing problem and sequencing problem) have to be solved for FJSSP, the symbolic string with fixed length in each chromosome can be divided into two parts, the routing part and the sequencing part. They are, respectively, used to map into a MAR and a JAR. Every gene in each part of a chromosome is composed of a head and a tail of symbolic string. It is stipulated that: (1) in the routing part, the head of a gene may contain symbols from both FS and TS-R, whereas

Table 1 Definitions of the functions and terminals that are used in this research

	Meaning
FS	
$+, -, \times$	The corresponding arithmetic functions, respectively
/	The protected division which returns 1 when the denominator is equal to 0
TS-R	
no	Number of operations that have been processed on the same machine
p	Processing time of an operation on a machine
P	Input power of a machine
P^{idle}	Idle power of a machine
P^{runtime}	The increase in power based on P^{idle} when executing the runtime operations for processing on a machine
P^{cutting}	The increase in power based on P^{runtime} when actually executing cutting for an operation on a machine
t	Cutting time of an operation on a machine
ti	Sum of idle time on a machine
tp	Sum of processing time of operations that have been processed on the same machine
wno	Number of operations waiting to be processed on the same machine
wp	Sum of processing time of operations waiting to be processed on the same machine
TS-S	
d	Due date of a job
nl	Number of unprocessed operations of a job
p	Processing time of an operation on a machine
pl	Remaining processing time of a job
qt	Interval between starting time of current operation of a job and completing time of its immediate predecessor operation
sl	Slack time of a job($\max\{0, d - \text{CT} - \text{pl}\}$)
t	Cutting time of an operation on a machine
tl	Remaining cutting time of a job
w	Weighted importance of a job
wt	Interval between starting time of current operation on a machine and completing time of the previous operation on the same machine

the tail of a gene consists only of symbols in TS-R; in the sequencing part, the head may contain symbols from both FS and TS-S, whereas the tail consists only of symbols in TS-S (Nie et al. 2013b); (2) assume the symbolic string has lh symbols in the head and lt symbols in the tail, then lh and lt satisfy the equation $\text{lt} = \text{lh} \times (\text{ma} - 1) + 1$, where *ma* is the maximum number of arguments for all the operators in FS (Ferreira 2001). The stipulation ensures the validity of the computer program's output (Hardy and Steeb 2002). For example, if we set $\text{lh} = 6$ and $\text{ma} = 2$, then $\text{lt} = 6 \times (2 - 1) + 1 = 7$, and the total gene length is 13. Assume the routing part and the sequencing part both consist of two genes. Thus, a chromosome comprises four genes and its length is $4 \times 13 = 52$. Figure 4 describes the mapping between a chromosome and a solution to FJSSP. As shown in Fig. 4a, in a typical chromosome, "||" is used to separate routing part and sequencing part, "|" is used to separate different genes, and "." is used to separate different symbols. The tail of each gene is indicated with an underline.

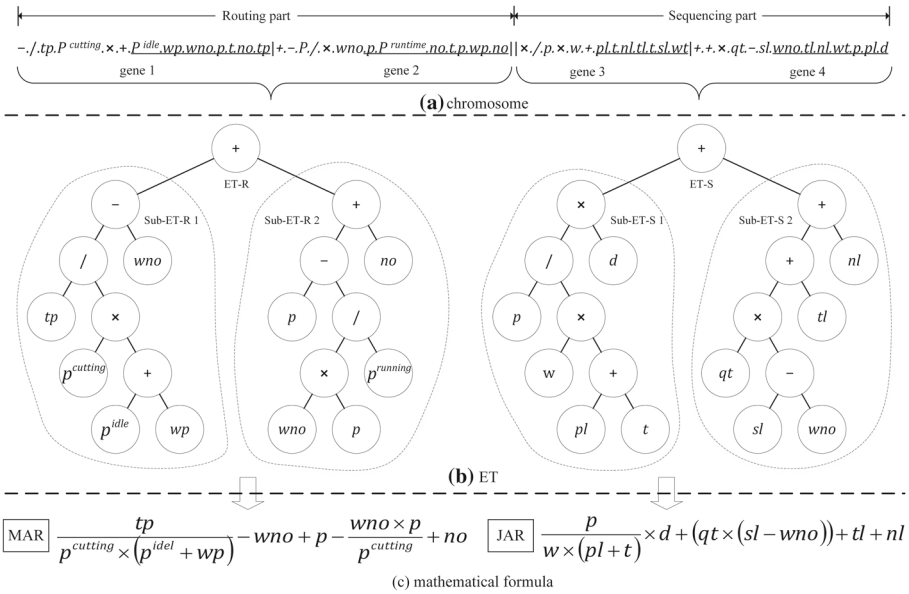

Fig. 4 Mapping between a chromosome and a solution to FJSSP

The routing part of the chromosome, consisting of gene 1 and gene 2, are mapped into an expression tree for the routing problem (ET-R) in depth-first fashion. Gene 1 and gene 2 are linked with the add function shown in Fig. 4b. Similarly, the sequencing part with gene 3 and gene 4 are mapped into the expression tree for the sequencing problem (ET-S). A node of the expression tree is attached when it indicates a function and stops being attached when it indicates a terminal. The branch number of a node is as many as the number of arguments of the function which the node corresponds to. The expression tree for the routing or sequencing problem can be further interpreted into the mathematical form as shown in Fig. 4c, which are MAR and JAR. In ET-R, the sub-ET-R 1 corresponding to gene 1 indicated by the dash line interacts with the sub-ET-R 2 corresponding to gene 2 in a way of addition. Subtrees interact with each other in the same way in ET-S.

5.2.3 Evolutionary search operators in PGEP

The evolutionary operators employed in PGEP are described as follows.

Selection and replication A set of individuals is selected according to the fitness through the roulette wheel method and is invariably replicated to the next generation, which means the better the fitness is, the greater the chance to be chosen will be.

Mutation Mutation can occur anywhere in the chromosome. Owing to the difference between symbols in the routing part and the sequencings part of chromosomes, it is stipulated that: (1) in the head of the routing part, any symbol can change into any other symbol in the set FS ∪ TS − R, while symbols in the tail can only change into terminals in TS-R; (2) in the head of the sequencing part, any symbol can change into any other symbol in set

FS ∪ TS − S, while symbols in the tail can only change into terminals in TS-S. The following equation is utilized to adaptively adjust the mutation probabilities.

$$\mathrm{mp} = \begin{cases} \rho_1 - \frac{(\rho_1-\rho_2)\times(\mathrm{fit\ (min)}+\mathrm{fit}\ (i))}{\mathrm{fit\ (min)}+\mathrm{fit\ (avg)}}, & \mathrm{fit}(i) \geq \mathrm{fit(avg)} \\ \rho_1, & \mathrm{fit}(i) < \mathrm{fit(avg)} \end{cases} \tag{16}$$

where fit(min) and fit(avg) denote the minimum, average of fitness of individuals at a sub-populations, respectively, and fit(i) represents the fitness of individual to be mutated. ρ_1, ρ_2 are both constants, and $\rho_1 > \rho_2$, where $\rho_1, \rho_2 \in (0, 1)$.

Transposition Randomly choose a fragment of a chromosome that usually consists of several successive symbols and transpose it to another site of the chromosome. In this paper, two types of transposition operator are adopted for each part of the chromosome: (1) transposition of the insertion sequence (IS), i.e., a fragment of a chromosome is randomly chosen and inserted in any site of the head of a gene except the beginning site of the head, while the same number of symbols are removed from the end of the head to make room for the inserted string in order not to break the parting line between the head and the tail; (2) transposition of the root insertion sequence (RIS), i.e., randomly select a fragment which begins with a function (called RIS elements) and insert it into the head of a gene in the same part. As same as the IS transposition, the same number of symbols are removed from the end of the head.

Recombination Randomly choose two parent chromosomes, and exchange some of their materials between them to form two new daughter chromosomes. Two types of recombination are used in this paper: (1) one-point recombination, i.e., splitting the chromosomes into halves and swapping the corresponding sections; (2) two-point recombination, i.e., splitting the chromosomes into three portions and swapping the middle one.

5.2.4 Fitness function

The focus of this paper is to learn effective SRs for FJSSP to minimise sum of TWTP and TEC. Due to the large difference among the objective values obtained by SRs for each instance, the quality of an obtained schedule will be measured by the relative deviation of its objective value from its corresponding reference objective value as shown in the following formula (Nguyen et al. 2013).

$$\mathrm{dev}(A; B, I_n) = \frac{\mathrm{Obj}(A; \mathrm{B}, I_n) - \mathrm{Ref}(I_n)}{\mathrm{Ref}(I_n)}. \tag{17}$$

In this formula, Obj(A; B, I_n) is the objective value obtained using rule (A; B) to solve instance I_n, while A and B are, respectively, one of MARs and JARs. Ref(I_n) is the reference objective value for instance I_n. The fitness value of rule (A; B) on the data set is expressed as follows:

$$\mathrm{dev}_{\mathrm{average}}(A; B) = \frac{\sum_{I_n \in \{I_1, I_2, \ldots, I_{ni}\}} \mathrm{dev}(A; B, I_n)}{\mathrm{ni}}. \tag{18}$$

The fitness value $\mathrm{dev}_{\mathrm{average}}(A; B)$ is used to measure the average performance of rule (A; B) on the data set with ni instances.

Since rule EDD is one of the effective SRs commonly used in solving FJSSP to minimise job tardiness or makespan and rule LWP is the only MAR presented in our research that is directly related to energy consumption, we will use the objective function values obtained by rule (LWP; EDD) (seen in Table 3) as the reference objective values for all instances in

the data set. Moreover, preliminary experimental results show that (LWP; EDD) performed better than many other SRs. Choosing a relatively good rule of (LWP; EDD) for reference, we can more intuitively analyse and depict the performance of PGEP-evolved SRs. Obviously, if the value of formula (18) is negative, it means that the average performance of rule (A; B) on the data set $(I_1, I_2, \ldots, I_{\text{ni}})$ is better than that of (LWP; EDD). The smaller the fitness value is, the better the individual is.

5.2.5 Migration scheme

To increase the performance of the parallel evolutionary algorithm, a migration scheme is proposed in PGEP. A population (defined by popSet) comprises N_{pop} equal sized sub-populations, each of which has N_{ind} individuals. A threshold is adopted to control the migration between any two sub-populations, and the penetration theory is used to set the migration interval, migration rate and migration direction adaptively. Suppose pop_x and pop_y are two randomly chosen sub-populations from popSet, $\text{fit}(x)$ and $\text{fit}(y)$ are the best fitness of individuals in pop_x and pop_y, respectively, and $\Delta\text{fit}(x, y) = \text{fit}(x) - \text{fit}(y)$. Then, the migration rate between pop_x and pop_y is defined by λ as follows:

$$\lambda = \begin{cases} \left| \frac{\Delta\text{fit}(x,y)}{\max\{\text{fit}(x),\text{fit}(y)\}} \right|, & |\Delta\text{fit}(x, y)| > \theta \\ 0, & |\Delta\text{fit}(x, y)| \leq \theta \end{cases} \tag{19}$$

where $\theta(\theta \geq 0)$ is the threshold by which the migration is determined.

The migration direction depends on the sign of $\Delta\text{fit}(x, y)$. If $\Delta\text{fit}(x, y) < 0$, the migration from pop_x to pop_y takes place and vice versa. Furthermore, if $\Delta\text{fit}(x, y) < 0$ and $\lambda > 0$, then $\text{int}(\lambda \times N_{\text{ind}})$ individuals with best fitness in pop_x migrates to pop_y and then $\text{int}(\lambda \times N_{\text{ind}})$ individuals with worst fitness in pop_y are removed. If $\lambda = 0$, no migration occurs regardless of $\Delta\text{fit}(x, y) < 0$ or not. The symbol "int" indicates the rounding operation.

6 Experiments and results

6.1 Design of the experiments

This subsection discusses the configuration of the PGEP method and the data sets used for training and testing. And also, the benchmark SRs are described in detail.

6.1.1 PGEP parameter settings

Based on the results from previous studies and data attained from extensive preliminary experiments, the parameters' setting in PGEP is shown in Table 2.

The population size and the length of the head of chromosome are, respectively, set to 50 and 20. Of course, the larger population size and length of the head are, the larger the algorithm search space is, and also the longer calculation time of algorithm is. The reason for this setting is mainly to reduce the computational times of PGEP and to facilitate the analysis of the performance of PGEP-evolved SRs.

The influence of three important parameters of np, lc, and θ on the performance of PGEP is one of the research focuses of this paper. We set these parameters to three different levels, respectively, as shown in bold in Table 2. There are $3 \times 3 \times 3 = 27$ combinations in total.

Table 2 Parameters' setting in PGEP

Parameters' setting in PGEP	
Number of populations (***np***)	**2, 4, 8**
Population size (N_{ind})	50
Length of chromosome (lc)	**4, 8,12**
Length of head	20
Selection strategy	Roulette wheel sampling
Mutation probability	0.05
IS transposition probability	0.1
RIS transposition probability	0.1
Gene transposition probability	0.1
One-point recombination probability	0.2
Two-point recombination probability	0.5
Gene recombination probability	0.1
Termination condition	Iteration generations > 150 or no improvement with consecutive 20 generations
Threshold for migration (θ)	**0.05, 0.1, 100***

*No migration between any two sub-populations

6.1.2 Benchmark heuristics

To verify the effectiveness of PGEP, some frequently used classical human-made SRs (Doh et al. 2013; Nie et al. 2013b), are selected as benchmarks to be compared with the PGEP-evolved SRs. These benchmark SRs are summarised in Table 3.

6.1.3 Data generation and selection

Our experimental data are based on a library of flexible job shop problem instances with different features from http://people.idsia.ch/~monaldo/fjsp.html. Machine's power, job's weight and deadline, operation's cutting time on the machine, power cost and tardiness cost for each instance are appended by referring to the data from cooperative enterprise, a metal moulds maker in South China. The idle power of the machine is uniformly distributed between 1500 and 1800. The running power of the machine is η times of its idle power, where η is uniformly distributed between 1.3 and 2. This cutting power of the machine is ζ times of its running power, where ζ is uniformly distributed between 1.2 and 1.8. The weight of the job is uniformly distributed between 1 and 10.

The deadline of the job is calculated from:

$$d_i = \left(1 + R \times \frac{n}{m}\right) \times \sum_{j=1}^{g_i} \left(\sum_{k \in M_{ij}} p_{ijk} \right) / h_{ij} \tag{20}$$

where n is the number of jobs, m is the number of machines, and h_{ij} is the number of machines that can process the operation O_{ij}. R denotes the tightness factor of the due date, the larger the value of R, the looser the job's due date. R is set to be 0.1, 0.3 and 0.5.

It is assumed that the cutting time of the operation is 50–80% of the whole processing time, and the ratio is uniformly distributed.

Table 3 Benchmark scheduling rules

	Meaning
Operation machine selection rules	
LWP	Least waiting power, select a machine with the least power (sum of running power and cutting power required of waiting operations)
LWT	Least waiting time, select a machine with the least waiting time (sum of processing times of waiting operations)
SP	Smallest processing, select a machine with the smallest processing time of the imminent operation
SQ	Shortest queue, select a machine with the smallest number of waiting jobs
Job sequencing rules	
ATC	Apparent tardiness cost, select a job with the maximum apparent tardiness cost
COVERT	Cost over time, select a job with the maximum COVERT value
CR	Critical ratio, select a job with the minimum CR value
EDD	Earliest due date, select a job with the earliest due date
FIFO	First in first out, select a job that arrived the earliest at the queue of the machine
HPP	Highest processing power, select an operation with the highest processing power required (include running power and cutting power)
LJPT	Largest job processing time, select a job with the largest job processing time
LOPR	Least operation remaining, select a job with the least number of remaining operations
LOPT	Largest operation processing time, select a job with the largest operation processing time
LPP	Least processing power, select a operation with the least processing power required (include running power and cutting power)
LWKR	Least work remaining, select a job with the least work remaining
MDD	Modified due data, select a job with the minimum modified due date
Mon	Montagne, select a job with the minimum ratio defined as $p_{ijk}/\left(\left(\sum_{O_{ij}\in OW_k} p_{ijk}\right) - d_i\right)$
MST	Minimum slack time, select a job with the minimum slack time
MST/LWKR	Select a job with the minimum value of MST/LWKR
SJPT	Shortest job processing time, select a job with the shortest job processing time
SOPT	Shortest operation processing time, select a job with the shortest operation processing time
SOPT/LWKR	A rule combined SOPT and LWKR, select a job with the minimum value of SOPT/LWKR

The cpup in the experiment is based on the time-of-use electricity pricing in a city in South China, as shown in Table 4. Without loss of generality, the cpup of off-peak is set to be 1 unit per KWh, the cpup of mid-peak and on-peak are 2.68 and 4.23 times of those of off-peak, respectively, and the utp is set to be 10, 30 and 50 units per hour.

In the whole, there are three different due date tightness and three different tardiness cost level settings, making a total number of nine simulation experiment settings. The 1026 problem instances with different features, including levels of flexibility (the degree of freedom for operations of jobs to select machine) and energy cost ratio are generated and all the problem instances are divided into two groups. One is used as the training set (including 522 problem instances, Online Resource 1) to evolve SRs through PGEP, and the other (named

Table 4 Time-of-use electricity pricing

From	To	Price
00:00	06:00	Off-peak, 1.00 unit/kWh
06:00	08:00	Mid-peak, 2.68 units/kWh
08:00	11:00	On-peak, 4.23 units/kWh
11:00	13:00	Mid-peak, 2.68 units/kWh
13:00	15:00	On-peak, 4.23 units/kWh
15:00	18:00	Mid-peak, 2.68 units/kWh
18:00	21:00	On-peak, 4.23 units/kWh
21:00	22:00	Mid-peak, 2.68 units/kWh
22:00	24:00	Off-peak, 1.00 unit/kWh

test set below, including 504 problem instances, Online Resource 2) is applied to valid the best individual of evolved SRs.

6.1.4 Parallel simulating environment construction

The proposed PGEP learning module and job shop simulation module have been implemented in C# with .Net Framework 4.5. Based on shared nothing architecture, parallel simulation on multiple physical servers can be conveniently implemented with the tools of Task Parallel Library (TPL) and Parallel LinQ (PLINQ) (Microsoft Docs 2017). The PGEP learning module runs on a PC server with Intel Xeon CPU 2.4 GHz processor, 8 CPU threads, 8 GB of RAM. The job shop simulation module runs on 12 distributed PC servers. Each PC server has an Intel Xeon CPU 2.0 GHz processor and 32 GB of RAM. 16 CPU threads are started on each PC server. So, a total of 192 CPU threads are used for the simulation computing.

6.2 Analysis of the results

6.2.1 Effect of different parameter settings on PGEP

In this subsection, we will use PGEP with different parameter settings to evolve SRs. As described in 6.1.1, we need to conduct a total of $3 \times 3 \times 3 = 27$ experiments. The triple $\langle \text{np}, \text{lc}, \theta \rangle$ indicates the setting of the PGEP parameters in a specific experiment. For example, $\langle 8, 12, 0.05 \rangle$ represents the experiment in which np is 8, and lc, θ are 12 and 0.05, respectively. To reduce the computational times of PGEP, experiments were carried out on part of problem instances in the training set and the test set with setting the parameters R and utp to 0.3 and 30, respectively, for all instances, which means that 58 instances in the training set (called the training set 1) and 56 instances in the test set (called the test set 1) were used to conduct experiments. For each experiment, ten independent runs of evolutionary learning were performed using PGEP on each instance in the training set 1 with different random seeds, and then ten optimal SRs were obtained. Then, the means of fitness values of the best evolved SRs from each run obtained from all experiment on the training set 1 and the test set 1 were attained.

Using np, lc and θ as independent variables, and $\text{dev}_{\text{average}}(A; B)$ as the dependent variable, multiple linear regression analysis was performed on the training set 1 and the test set 1,

Table 5 Analysis of the two multiple linear regression models

	Training set 1	Test set 1
Estimate of variable		
np	− 5.052E − 03	− 3.819E − 03
lc	2.944E − 03	2.421E − 03
θ	1.054E − 04	6.973E − 05
Intercept	− 3.312E − 01	− 2.893E − 01
p value for variable		
np	< 2E − 16	< 2E − 16
lc	< 2E − 16	< 2E − 16
θ	< 2E − 16	7.52E − 07
Intercept	< 2E − 16	< 2E − 16
Degree of freedom	266	266
Multiple R-squared	0.8832	0.5949
Adjusted R-squared	0.8819	0.5903
F-statistic	670.8	130.2
Prob (F-statistic)	< 2.2E − 16	< 2.2E − 16

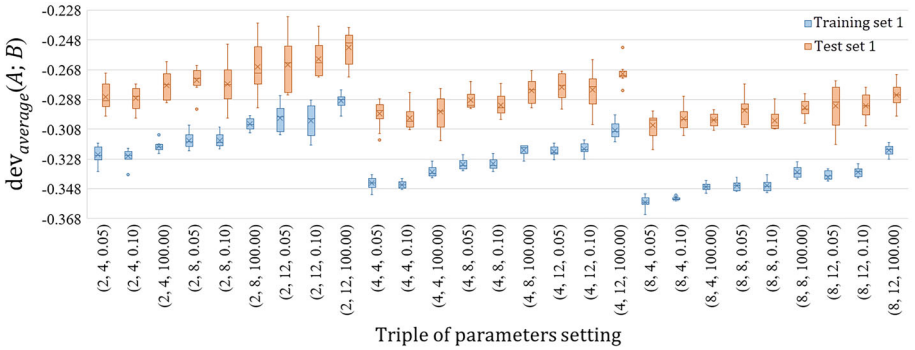

Fig. 5 Box plot for the effect of different parameter settings on PGEP

respectively. The analysis and test results of the two multiple linear regression models are shown in Table 5 and additional data are given in Online Resource 3.

It can be seen from Table 5, the setting of parameters *np*, *lc* and θ has a significant influence on the performance of PGEP (all p values $\leq$ 7.52E − 07). Figure 5 represents the box plot representations of the performance of PGEP-evolved SRs on the training set 1 and the test set 1 with different settings.

The statistical tests discussed in the subsequent subsections are the equal variance t tests and they are considered significant if the obtained p value is less than 0.05.

As shown in Fig. 5, the PGEP-evolved SRs with larger np are generally better than those evolved with small np both on the training set 1 and the test set 1. It also can be seen from Fig. 5, the PGEP-evolved SRs with small lc generally outperform those evolved with larger lc both on the training set 1 and the test set 1. Moreover, the obvious difference is observed between the experiment with migration and the experiment without migration, regardless of

whether on the training set 1 or on the test set 1. Overall, the performance of PGEP with migration is better than that of PGEP without migration.

To summarise, triple $\langle 8, 4, 0.05 \rangle$ gives the best performance both on the training set 1 and the test set 1. Therefore, in the subsequent experiments, we set np, lc and θ to 8, 4, 0.05, respectively.

6.2.2 Comparison with human-made SRs

In this subsection, we will compare the performance of PGEP-evolved SRs with those of human-made SRs. First, a total of 72 (4×18) human-made SRs were applied to solve the instances in the training set and the test set, respectively. Then, ten independent runs of evolutionary learning were performed using PGEP on all instances in the training set with different random seeds, and ten optimal SRs were obtained, denoted as PGEP1, PGEP2, …, PGEP10, respectively.

Subsequently, these ten SRs were, respectively, used to solve instances in the training set and the test set. Table 6 shows the statistics of relative deviations using formula (17). The values of mean can be calculated using formula (18) to measure the average performance of a rule on a given set of instances, while min value and max value show the best-case performance and worst-case performance, respectively (additional data are given in Online Resource 4).

As shown in Table 6, on the training set, (LWT; SJPT) performs best among all human-made SRs, while (LWT; EDD) beats all other human-made SRs on the test set. Although (LWT; MST) cannot perform as well as (LWT; SJPT) on the training set, in the best case, it can provide a better schedule than that obtained by (LWT; SJPT). However, on the test set, in the worst case, (LWT; EDD) cannot provide better schedules than those obtained by (LWP; EDD), (LWP; MDD), (LWP, MST), (LWP; SJPT) and (LWT; SMT). It is not easy to find a human-made rule that can totally dominate others in most cases.

Obviously, the ten PGEP-evolved SRs all show better average, best-case and worst-case performance than all human-made SRs on the training set. On the test set, these ten SRs also outperform all human-made SRs in terms of the average and best-case performance. The ten PGEP-evolved SRs beat all human-made SRs in about 95.9% of instances in whole data set. However, in the worst case, some of these ten SRs cannot provide better schedules than those obtained by several excellent human-made SRs. For example, they are beaten by (LWP, EDD) on some instances which are derived from the "Mk04" and "Mk06" problems with different settings of parameters R and utp. Figure 6 represents the histogram representations of the performance of these ten SRs on solving instances mentioned above. It can be seen from Fig. 6, overall, the performance of these ten PGEP-evolved SRs on solving the Mk04 problem instances is better than that on solving the Mk06 problem instances. Only when R is 0.5, can (LWP, EDD) beat some of these ten PGEP-evolved SRs on the Mk04 problem instances. In the worst case, when R and utp are set to 0.5 and 50, respectively, 6 of 10 PGEP-evolved SRs are beaten by (LWP, EDD) on the Mk04 problem instances, while (LWP, EDD) outperforms 8 of 10 PGEP-evolved SRs on the Mk06 problem instances.

6.2.3 Comparison with PGEP without migration and GEP

In this subsection, the performance of PGEP with migration (the proposed PGEP) will be compared with those of GEP and PGEP without migration.

Table 6 Relative deviations obtained by PGEP-evolved SRs and human-made SRs

SRs	Training set			Test set			SRs	Training set			Test set		
	Min	Mean	Max	Min	Mean	Max		Min	Mean	Max	Min	Mean	Max
LWP, ATC	− 0.453	0.580	14.624	− 0.518	0.618	10.302	SP, ATC	− 0.424	0.717	13.785	− 0.737	0.812	13.937
LWP, COVERT	− 0.616	0.301	6.405	− 0.735	0.328	5.049	SP, COVERT	− 0.485	0.519	9.498	− 0.737	0.543	9.589
LWP, CR	− 0.559	0.460	8.570	− 0.453	0.477	8.381	SP, CR	− 0.482	0.635	11.678	− 0.324	0.825	11.625
LWP, EDD	0.000	0.000	0.000	0.000	0.000	0.000	SP, EDD	− 0.660	0.180	5.258	− 0.702	0.233	7.531
LWP, FIFO	− 0.616	0.496	8.736	− 0.735	0.554	9.511	SP, FIFO	− 0.485	0.784	16.189	− 0.737	0.786	13.343
LWP, HPP	− 0.320	0.838	12.473	− 0.361	0.857	12.873	SP, HPP	− 0.482	1.187	25.648	− 0.269	1.365	18.115
LWP, LJPT	− 0.493	0.837	14.821	− 0.637	0.787	12.302	SP, LJPT	− 0.424	1.222	20.431	− 0.737	1.463	24.297
LWP, LOPR	− 0.423	0.319	6.358	− 0.637	0.292	6.373	SP, LOPR	− 0.485	0.558	15.172	− 0.737	0.621	12.414
LWP, LOPT	− 0.320	0.842	13.870	− 0.361	0.836	12.873	SP, LOPT	− 0.482	1.181	25.648	− 0.269	1.418	16.844
LWP, LPP	− 0.520	0.125	3.600	− 0.735	0.185	3.181	SP, LPP	− 0.468	0.253	6.729	− 0.737	0.237	6.198
LWP, LWKR	− 0.507	0.070	2.152	− 0.471	0.142	3.009	SP, LWKR	− 0.485	0.334	9.891	− 0.544	0.505	8.611
LWP, MDD	− 0.237	0.012	0.809	− 0.244	0.025	0.651	SP, MDD	− 0.660	0.208	6.689	− 0.702	0.267	4.717
LWP, Mon	− 0.616	0.507	8.736	− 0.584	0.557	9.511	SP, Mon	− 0.485	0.797	16.189	− 0.587	0.813	13.343
LWP, MST	− 0.645	0.105	1.474	− 0.717	0.133	1.593	SP, MST	− 0.482	0.364	7.243	− 0.737	0.523	11.570
LWP, MST/LWKR	− 0.633	0.402	5.452	− 0.717	0.449	7.751	SP, MST/LWKR	− 0.485	0.721	17.903	− 0.737	0.746	14.024
LWP, SJPT	− 0.348	0.001	0.354	− 0.285	0.013	0.903	SP, SJPT	− 0.660	0.190	5.258	− 0.702	0.239	7.531
LWP, SOPT	− 0.572	0.080	3.958	− 0.735	0.196	3.869	SP, SOPT	− 0.570	0.228	6.776	− 0.737	0.272	4.255
LWP, SOPT/LWKR	− 0.625	0.542	9.535	− 0.735	0.618	10.809	SP, SOPT/LWKR	− 0.485	0.834	19.557	− 0.737	0.832	18.167
LWT, ATC	− 0.650	0.537	11.601	− 0.737	0.557	9.863	SQ, ATC	− 0.493	0.624	11.485	− 0.522	0.708	9.667
LWT, COVERT	− 0.685	0.266	5.432	− 0.737	0.274	6.764	SQ, COVERT	− 0.542	0.423	7.197	− 0.522	0.422	6.023
LWT, CR	− 0.663	0.339	7.473	− 0.650	0.416	6.670	SQ, CR	− 0.440	0.475	5.191	− 0.476	0.476	6.126
LWT, EDD	− 0.668	− 0.037	1.266	− 0.681	− 0.019	1.608	SQ, EDD	− 0.518	0.101	2.657	− 0.714	0.122	4.045

Table 6 continued

SRs	Training set			Test set			SRs	Training set			Test set		
	Min	Mean	Max	Min	Mean	Max		Min	Mean	Max	Min	Mean	Max
LWT, FIFO	− 0.685	0.462	10.007	− 0.737	0.510	9.538	SQ, FIFO	− 0.542	0.619	8.869	− 0.522	0.664	9.937
LWT, HPP	− 0.614	0.661	10.951	− 0.650	0.737	11.285	SQ, HPP	− 0.440	0.758	9.898	− 0.476	0.873	11.766
LWT, LJPT	− 0.573	0.726	13.590	− 0.737	0.748	11.556	SQ, LJPT	− 0.478	0.915	15.604	− 0.522	0.901	11.118
LWT, LOPR	− 0.611	0.217	8.629	− 0.737	0.248	6.219	SQ, LOPR	− 0.478	0.372	7.398	− 0.693	0.354	5.029
LWT, LOPT	− 0.614	0.680	10.622	− 0.650	0.715	11.285	SQ, LOPT	− 0.389	0.809	12.867	− 0.476	0.864	11.766
LWT, LPP	− 0.649	0.070	2.374	− 0.737	0.121	2.711	SQ, LPP	− 0.486	0.240	3.667	− 0.522	0.258	4.880
LWT, LWKR	− 0.630	0.050	4.066	− 0.681	0.103	3.307	SQ, LWKR	− 0.618	0.136	2.052	− 0.705	0.193	4.521
LWT, MDD	− 0.668	− 0.033	1.266	− 0.681	− 0.005	1.608	SQ, MDD	− 0.520	0.110	2.153	− 0.714	0.146	4.458
LWT, Mon	− 0.685	0.465	10.007	− 0.650	0.518	9.538	SQ, Mon	− 0.542	0.625	8.869	− 0.634	0.661	9.937
LWT, MST	− 0.790	0.085	1.014	− 0.737	0.081	1.264	SQ, MST	− 0.683	0.217	1.671	− 0.714	0.236	3.984
LWT, MST/LWKR	− 0.685	0.340	4.379	− 0.737	0.429	6.605	SQ, MST/LWKR	− 0.478	0.522	6.346	− 0.545	0.571	6.139
LWT, SJPT	− 0.668	− 0.040	1.266	− 0.681	− 0.018	2.015	SQ, SJPT	− 0.518	0.096	1.408	− 0.714	0.092	2.458
LWT, SOPT	− 0.649	0.077	4.159	− 0.737	0.116	3.187	SQ, SOPT	− 0.486	0.203	3.570	− 0.583	0.236	3.248
LWT, SOPT/LWKR	− 0.685	0.535	10.444	− 0.737	0.579	10.793	SQ, SOPT/LWKR	− 0.542	0.671	9.460	− 0.522	0.702	10.536
PGEP 1	− 0.873	− 0.329	− 0.015	− 0.751	− 0.292	0.553	PGEP 6	− 0.876	− 0.329	− 0.016	− 0.75	− 0.288	0.175
PGEP 2	− 0.857	− 0.328	− 0.021	− 0.774	− 0.286	0.451	PGEP 7	− 0.872	− 0.328	− 0.015	− 0.75	− 0.287	0.444
PGEP 3	− 0.852	− 0.329	− 0.017	− 0.751	− 0.293	0.302	PGEP 8	− 0.872	− 0.33	− 0.017	− 0.749	− 0.293	0.217
PGEP 4	− 0.873	− 0.331	− 0.013	− 0.768	− 0.287	0.279	PGEP 9	− 0.854	− 0.33	− 0.015	− 0.754	− 0.284	0.556
PGEP 5	− 0.853	− 0.329	− 0.015	− 0.749	− 0.288	0.348	PGEP 10	− 0.881	− 0.329	− 0.017	− 0.76	− 0.288	0.313

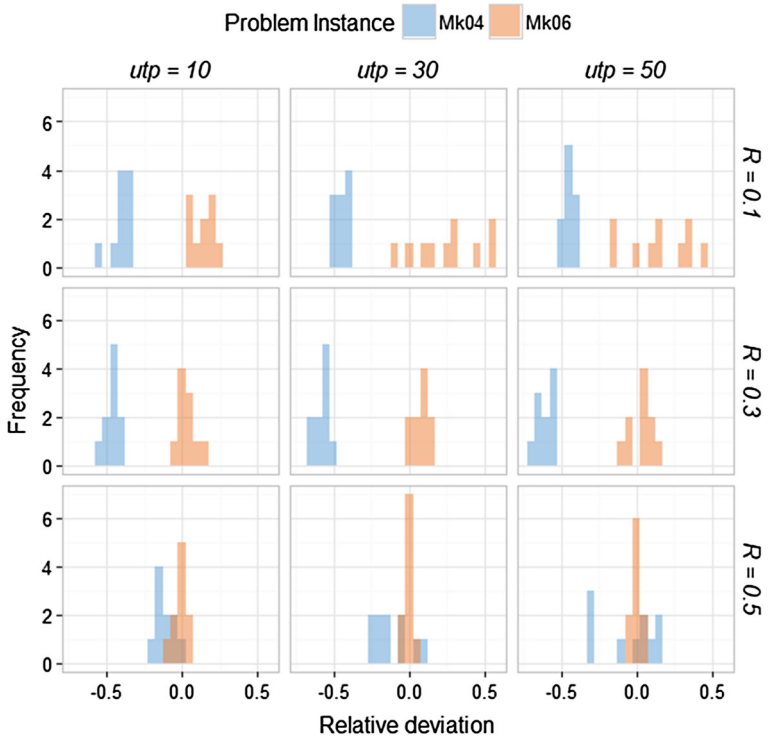

Fig. 6 Performance of the ten PGEP-evolved SRs on solving the Mk04 and Mk06 problem instances

The experiment is divided into two parts. In part one, we set population size of GEP to 400 (50×8), which is equal to the total number of chromosome individuals in the 8 sub-populations of the proposed PGEP. In a serial computing environment, ten independent runs of evolutionary learning based on GEP were performed on the training set and ten optimal SRs were obtained. Then, these ten SRs were, respectively, applied to solve instances in the training set and the test set. In part two, we removed the migration scheme from the proposed PGEP, that is, we set θ to a relatively large value (represented by 100 in this paper, indicating that no migration occurs between any two sub-populations), and then conducted the experiment similar to that in part one in a parallel computing environment.

Figure 7 represents the box plot representation of the achieved results of the PGEP with migration, PGEP without migration and GEP on the training set and the test set (additional data are given in Online Resource 5). The statistical tests show that PGEP with migration is significantly better than PGEP without migration and GEP on the training set (p values are, respectively, 5.917E−12 and 7.065E−19). On the test set, PGEP with migration is also significantly superior to PGEP without migration and GEP (p values are, respectively, 1.788E−02 and 1.035E−14). From experimental results, it can be concluded that migration among multiple sub-populations enhances the genetic diversity of populations, with which the learning precocity is avoided and the convergence speed of proposed PGEP is also accelerated. The effectiveness of migration scheme proposed in this paper was verified again on the whole data set.

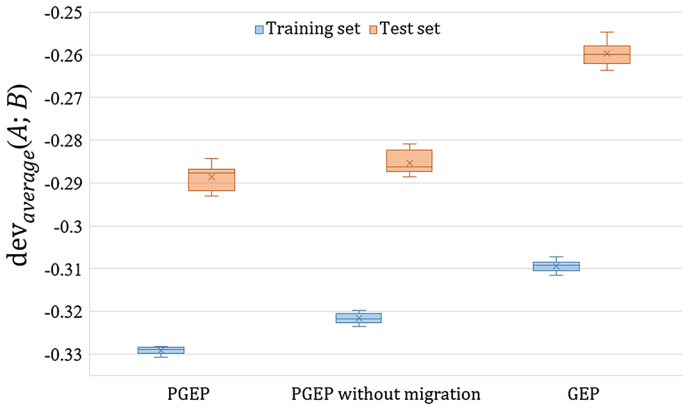

Fig. 7 Comparison among PGEP with migration, PGEP without migration and GEP

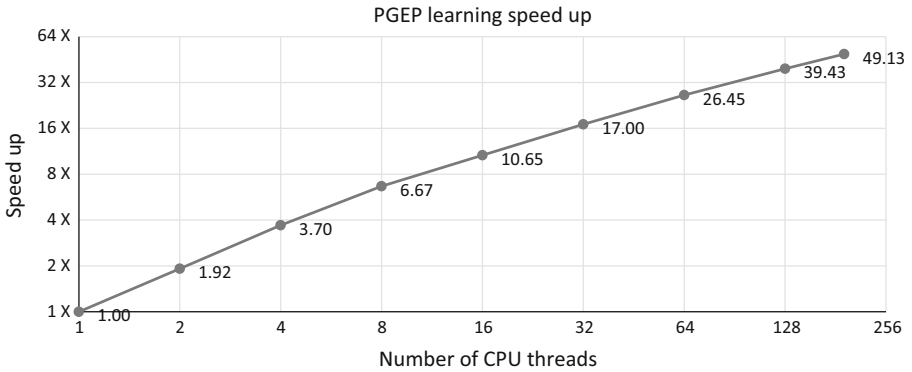

Fig. 8 PGEP learning speed up

Moreover, it should also be noted that PGEP without migration is significantly better than GEP both on the training set and the test set (both p values $\leq$ 5.465E−14). It can be inferred that although the chromosomal genotype has not changed, a relatively large change in genetic frequency took place during the evolution of small populations, which resulted in small populations evolving faster than large populations. So, multiple small populations parallel search policy can expedite the convergence of PGEP.

As for time efficiency, PGEP is undoubtedly better than GEP. In the proposed PGEP, the simulation module with distributed computing performed about 759 times simulations per second. About 0.81 s was consumed in average to evaluate 400 individuals of a new generation of all populations, including the communication time between simulation module and learning module. Compared with the serial computing, calculation speed of PGEP is improved by about 49.13 times through parallel computing. As 192 CPU threads are used for the computing tasks and the inter-thread communication consumes lots of time, the efficiency of PGEP is approximately 25.59%. The trends in relationships between calculation speed in PGEP and number of CPU threads are summarised in Fig. 8.

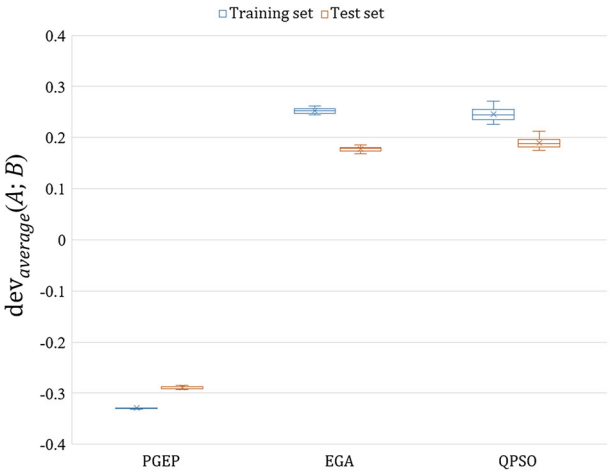

Fig. 9 Comparison among PGEP-evolved SRs, EGA and QPSO

6.2.4 Comparison with meta-heuristic methods

This subsection will compare the performance of the PGEP-evolved SRs to two meta-heuristic methods. Although it seems unfair to the proposed PGEP method, since most meta-heuristic methods were specially developed for the static environment and expected to obtain better solutions through iteratively exploring the solution search space of each instance, at least the comparison can provide an indicator about what level of performance the GPEP-evolved SRs can achieve (Nguyen et al. 2013).

In this subsection, we compared the performance of the PGEP-evolved SRs with those of a genetic algorithm coupling elitism strategy (EGA) (Xie and Chen 2018) and a quantum behaved particle swarm optimization (QPSO) (Singh and Mahapatra 2016). According to the authors' conclusions (Xie and Chen 2018), population size, max iteration generations in EGA are set to 100, 150, respectively. Since the authors had not described the setting of population size and max iteration generations in QPSO in their paper, for the sake of fairness, population size and max iteration generations in QPSO are set to be same as those in EGA, respectively.

Using EGA and QPSO, respectively, ten independent runs of solving each instance in the training set and the test set were completed. As shown in Fig. 9, on the training set, GPEP-evolved SRs are significantly better than EGA and QPSO (both p values $\leq$ 9.66E−29), and they can provide good average, best-case and worst-case performance. On the test set, GPEP-evolved SRs also significantly outperforms EGA and QPSO (both p value $\leq$ 4.44E−28), and shows better average and worst-case performance than EGA and QPSO. However, in the best case, EGA and QPSO can provide better schedules than those obtained by GPEP-evolved SRs. The fact that the GPEP-evolved SRs can beat EGA and QPSO in most cases shows the effectiveness of these SRs.

The experimental results show that it is not easy for EGA and QPSO to find a satisfactory solution within relatively less time under the condition that the problems optimization search space is large. Although EGA and QPSO have achieved good results in some instances, their overall performances are not as good as those of some excellent human-made SRs, such as (LWP; EDD).

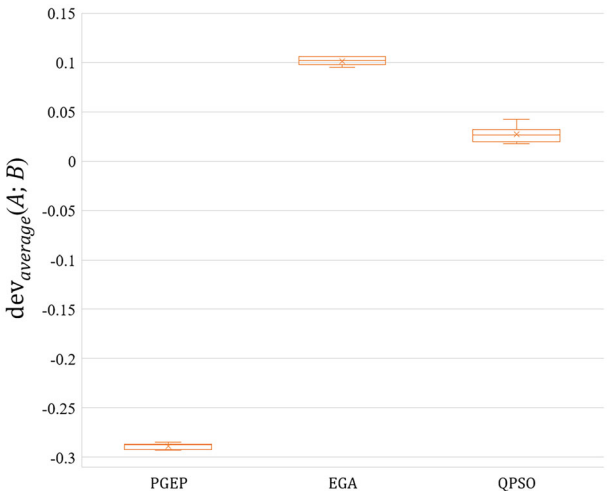

Fig. 10 Comparison among PGEP-evolved SRs, EGA and QPSO on the test set (max iteration generations in EGA and QPSO is 600)

With the consideration that PGEP requires about 60000($8 \times 50 \times 150$) evaluations to evolve and construct an optimal rule when populations number, population size and max iteration generations are set to 8, 50 and 150, respectively, we increase the parameter of max iteration generations in EGA and QPSO to 600. So, 60000(100×600) evaluations will also be needed for EGA and QPSO to solve an instance. Then, EGA and QPSO are, respectively, used to solve the 504 instances in the test set. Similarly, for each instance, ten independent runs of solving were performed with different random seeds.

Figure 10 represents the box plot representation of the achieved results of the PGEP-evolved SRs, EGA and QPSO. As shown in Fig. 10, the PGEP-evolved SRs are still significantly better than EGA and QPSO (both p values $\leq 1.64\text{E} - 27$).

To analyse the performance of PGEP, EGA and QPSO on each problem instance, the symbol $\text{dev}_{\text{average}}(I_n)$ is used to denote the average of relative deviations of ten runs of solving instance I_n using EGA or QPSO. For the PGEP approach, $\text{dev}_{\text{average}}(I_n)$ is the average of relative deviations obtained using the ten PGEP-evolved SRs to solve the instance I_n respectively. Figure 11 represents the histogram representations of frequency distribution of different solutions for each problem obtained by GPEP-evolved SRs, EGA and QPSO, respectively.

Among 504 instances, PGEP-evolved SRs outperforms EGA and QPSO in 337 instances, respectively. It should also be noted that EGA and QPSO, respectively, beat (LWP; EDD) in 289 and 300 instances through extensive searches. The performance rankings of the PGEP-evolved SRs, EGA, QPSO and (LWP; EDD) are presented in a bar graph, as shown in Fig. 12 (additional data are given in Online Resource 6).

Moreover, while both EGA and QPSO need 60, 000 evaluations to solve an instance, GPEP-evolved SRs can solve an instance in less than 0.52 s. The advantages of performance and efficiency make the GPEP-evolved SRs especially suitable for those application scenarios with high real-time requirements.

Another superiority of the proposed PGEP method is that it can easily incorporate the features and constraints of different flexible job shop manufacturing environments into the SRs without revising or redesigning the algorithm. Therefore, it is very suitable for con-

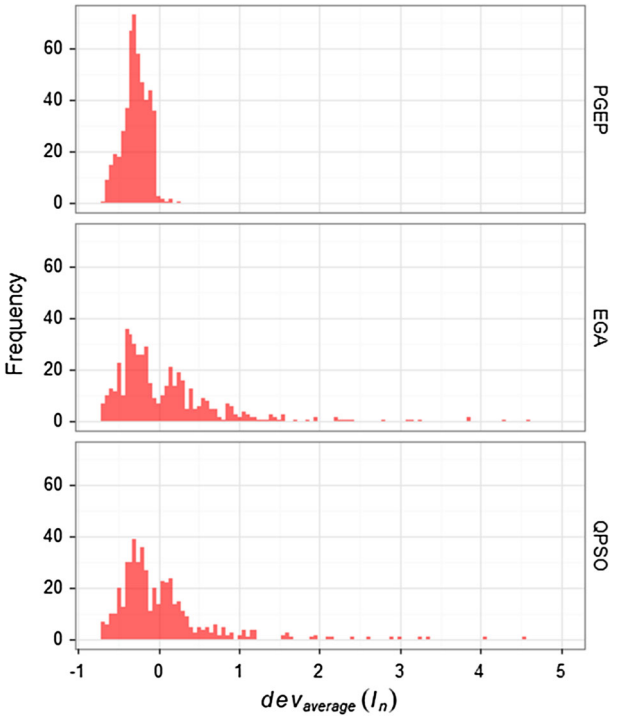

Fig. 11 Histogram of the performance of PGEP, EGA and QPSO on each instance in the test set

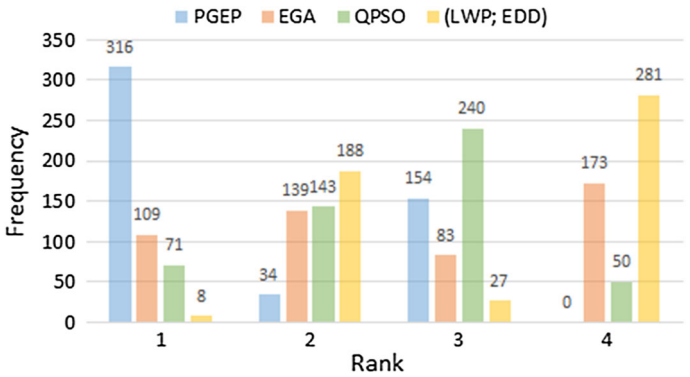

Fig. 12 Performance rankings of the PGEP-evolved SRs, EGA, QPSO and (LWP; EDD)

stantly exploring better SRs from new instances through incremental learning in a dynamic environment. This is very meaningful for solving those NP-hard scheduling problems, such as FJSSP.

7 Conclusion and future work

In this paper, the authors proposed a PGEP method to automatically discover new SRs to solve FJSSP considering total cost of energy consumption and weighted tardiness. In the proposed PGEP method, a framework for parallel learning and distributed simulation was designed to reduce the optimisation time and improve the solution quality. Multi-population evolution strategies with a migration scheme were also put forward to enhance the genetic diversity, which is vital for constructing better SRs. Extensive simulations have been carried out over a series of problem instances that represent various operation conditions generated by different levels of due date tightness and tardiness penalty weight. The experimental results show that the PGEP method can construct more efficient SRs for the complex FJSSP than the empirical SRs based on the experience of human schedulers, and that the proposed PGEP is significantly superior to PGEP without migration and GEP. Moreover, the PGEP-evolved SRs were also shown to outperform EGA and QPSO both on performance and efficiency. Compared to EGA and QPSO, PGEP-evolved SRs are more suitable for those application scenarios with high real-time requirements or dynamic systems. The PGEP method provides a convenient way to incorporate features of a particular system in a real environment into adaptive SRs. Therefore, it can be easily extended to other manufacturing systems as a tool to automatically discover effective SRs.

However, only a preliminary research on PGEP method to evolve SRs for FJSSP considering energy consumption and job tardiness was carried out in this paper. There are still many potential issues worth exploring further. Among them, it is very useful to delve into the design of terminal sets and function sets to discover more potential system properties and explore their impact on different types of manufacturing environments. For example, by adding the probability distribution variable of uncertain factors such as job durations changing, machines breakdown and recovery to the terminal set, we could use the proposed PGEP to evolve SRs to solve FJSSPs under uncertainty. These interesting studies are subject to further collection of experimental data and validation in the real manufacturing environment.

Acknowledgements The authors would like to thank the support from the National Natural Science Foundation of China (NSFC) (Nos. 51475096, 51675107, and 71571050), the NSFC-Guang Dong Collaborative Fund (no. U1501248), and the New Pearl River Star Program of Guangzhou City (201610010035).

References

Bruzzonea AAG, Anghinolfib D, Paoluccib M, Tonellia F (2012) Energy-aware scheduling for improving manufacturing process sustainability: a mathematical model for flexible flow shops. CIRP Ann Manuf Technol 61(1):459–462. https://doi.org/10.1016/j.cirp.2012.03.084

Chen GR, Zhang L, Arinez J, Biller S (2013) Energy-efficient production systems through schedule-based operations. IEEE Trans Autom Sci Eng 10(1):27–37. https://doi.org/10.1109/TASE.2012.2202226

Dahmus JB, Gutowski TC (2004) An environmental analysis of machining. In: ASME 2004 international mechanical engineering congress and exposition. https://doi.org/10.1115/IMECE2004-62600

Dai M, Tang D, Giret A, Salido MA, Li WD (2013) Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robot Comput Integr Manuf 29(5):418–429. https://doi.org/10.1016/j.rcim.2013.04.001

Diaz N, Choi S, Helu M, Chen Y, Jayanathan S, Yasui Y, Kong D et al (2010) Machine tool design and operation strategies for green manufacturing. Laboratory for Manufacturing & Sustainability, Berkeley

Doh YH, Yu JM, Kim JS, Lee DH, Nam SH (2013) A priority scheduling approach for flexible job shops with multiple process plans. Int J Prod Res 51(12):3748–3764. https://doi.org/10.1080/00207543.2013.76-5074

Drake R, Yildirim MB, Twomey J, Whitman L, Ahmad J, Lodhia P (2006) Data collection framework on energy consumption in manufacturing. In: Institute of industrial engineering research conference. http://hdl.handle.net/10057/3422. Accessed 10 May 2018

Driss I, Mouss KN, Laggoun A (2015) A new genetic algorithm for flexible job-shop scheduling problems. J Mech Sci Technol 29(3):1273–1281. https://doi.org/10.1007/s12206-015-0242-7

Duflou JR, Sutherland JW, Dornfeld D, Herrmann C, Jeswiet J, Kara S, Hauschild M et al (2012) Towards energy and resource efficient manufacturing: a processes and systems approach. CIRP Ann Manuf Technol 61(2):587–609. https://doi.org/10.1016/j.cirp.2012.05.002

Energy Information Administration (2005) Annual energy review 2004. http://www.eia.gov/totalenergy/data/annual/archive/038404.pdf. Released 19 Aug 2005

Fang KT, Lin BMT (2013) Parallel-machine scheduling to minimize tardiness penalty and power cost. Comput Ind Eng 64(1):224–234. https://doi.org/10.1016/j.cie.2012.10.002

Fang K, Uhan N, Zhao F, Sutherland JW (2011a) A new shop scheduling approach in support of sustainable manufacturing. Glocal Solut Sustain Manuf. https://doi.org/10.1007/978-3-642-19692-8_53

Fang K, Uhan N, Zhao F, Sutherland JW (2011b) A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. J Manuf Syst 30(4):234–240. https://doi.org/10.1016/j.jmsy.20-11.08.004

Ferreira C (2001) Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst 13(2):87–129. https://www.gene-expression-programming.com/webpapers/GEPfirst.pdf. Accessed 1 Jan 2018

Fysikopoulos A, Papacharalampopoulos A, Pastras G, Stavropoulos P, George C (2013) Energy efficiency of manufacturing processes: a critical review. Proced CIRP 7:628–633. https://doi.org/10.1016/j.procir.2013.06.044

Gema C, Rafael P (2014) A dispatching algorithm for flexible job-shop scheduling with transfer batches: an industrial application. Prod Plan Control 25(2):93–109. https://doi.org/10.1080/09537287.2013.782846

Gutowski T, Murphy C, Allen D, Bauer D, Bras B, Piwonka T, Sheng P et al (2005) Environmentally benign manufacturing: observations from Japan, Europe and the United States. J Clean Prod 13(1):1–17. https://doi.org/10.1016/j.jclepro.2003.10.004

Hardy Y, Steeb WH (2002) Gene expression programming and one dimensional chaotic maps. Int J Mod Phys C 13(1):13–24. https://doi.org/10.1142/S0129183102002912

He Y, Liu B, Zhang X, Gao H, Liu X (2012) A modeling method of task-oriented energy consumption for machining manufacturing system. J Clean Prod 23(1):167–174. https://doi.org/10.1016/j.jclepro.2011.10.033

Kara S, Manmek S, Herrmann C (2010) Global manufacturing and the embodied energy of products. CIRP Ann Manuf Technol 59(1):29–32. https://doi.org/10.1007/BF01719451

Li W, Zein S, Kara S, Herrmann C (2011) An investigation into fixed energy consumption of machine tools. Glocal Solut Sustain Manuf. https://doi.org/10.1007/978-3-642-19692-8_47

Liu X, Zou FX, Zhang XP (2008) Mathematical model and genetic optimization for hybrid flow shop scheduling problem based on energy consumption. In: 2008 Chinese control and decision conference (CCDC 2008). https://doi.org/10.1109/CCDC.2008.4597463

Liu Y, Dong HB, Lohse N, Petrovic S, Gindy N (2014) An investigation into minimising total energy consumption and total weighted tardiness in job shops. J Clean Prod 65(4):87–96. https://doi.org/10.1016/j.jclepro.2013.07.060

Liu GS, Zhou Y, Yang HD (2017) Minimizing energy consumption and tardiness penalty for fuzzy flow shop scheduling with state-dependent setup time. J Clean Prod 147(5):470–484. https://doi.org/10.1016/j.jclepro.2016.12.044

Melo ELD, Ronconi DP (2015) Efficient priority rules that explore flexible job shop characteristics for minimizing total tardiness. Production 25(1):79–91. https://doi.org/10.1590/S0103651320140050000016

Microsoft Docs (2017) Parallel programming in.Net. https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming. Released 30 Mar 2017

Mokhtaria H, Hasani A (2017) An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. Comput Chem Eng 104(9):339–352. https://doi.org/10.1016/j.compchemeng.2017.05.004

Moon JY, Park J (2014) Smart production scheduling with time-dependent and machine-dependent electricity cost by considering distributed energy resources and energy storage. Int J Prod Res 52(13):3922–3939. https://doi.org/10.1080/00207543.2013.860251

Moon JY, Shin K, Park J (2013) Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. Int J Adv Manuf Technol 68(1):523–535. https://doi.org/10.1007/s00170-013-4749-8

Mori M, Fujishima M, Inamasu Y, Oda Y (2011) A study on energy efficiency improvement for machine tools. CIRP Ann Manuf Technol 60(1):145–148. https://doi.org/10.1016/j.cirp.2011.03.099

Mouzon G (2008) Operational methods and models for minimization of energy consumption in a manufacturing environment. Dissertation, Wichita State University

Mouzon G, Yildirim MB, Twomey J (2007) Operational methods for minimization of energy consumption of manufacturing equipment. Int J Prod Res 45(18–19):4247–4271. https://doi.org/10.1080/00207540701450013

Neugebauer R, Wabner M, Rentzsch H (2011) Structure principles of energy efficient machine tools. CIRP J Manuf Sci Technol 4(2):136–147. https://doi.org/10.1016/j.cirpj.2011.06.017

Nguyen S, Zhang MJ, Johnston M, Tan KC (2013) 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. https://doi.org/10.1109/TEVC.2012.2227326

Nie L, Gao L, Li PG, Shao XY (2013a) Reactive scheduling in a job shop where jobs arrive over time. Comput Ind Eng 66(2):389–405. https://doi.org/10.1016/j.cie.2013.05.023

Nie L, Gao L, Li PG, Li XY (2013b) A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. J Intell Manuf 24(4):763–774. https://doi.org/10.1007/s10845-012-0626-9

Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 34(6):1539–1548. https://doi.org/10.1016/j.apm.2009.09.002

Rahimifard S, Seow Y, Childs T (2010) Minimising embodied product energy to support energy efficient manufacturing. CIRP Ann Manuf Technol 59(1):25–28. https://doi.org/10.1016/j.cirp.2010.03.048

Seow Y, Rahimifard S (2011) A framework for modelling energy consumption within manufacturing systems. CIRP J Manuf Sci Technol 4(3):258–264. https://doi.org/10.1016/j.cirpj.2011.03.007

Shrouf F, Meré JO, Sánchez AG, Mier MO (2014) Optimizing the production scheduling of a single machine to minimize total energy consumption costs. J Clean Prod 67(6):197–207. https://doi.org/10.1016/j.jclepro.2013.12.024

Singh MR, Mahapatra SS (2016) A quantum behaved particle swarm optimization for flexible job shop scheduling. Comput Ind Eng 93:36–44. https://doi.org/10.1016/j.cie.2015.12.004

Solar Energy International (2015) Energy fact. http://www.solarenergy.org/resources/energyfacts.html. Accessed 2 March 2015

Su ZL, Yuan JL, Chen W (2012) Flexible job-shop scheduling analysis and its heuristic algorithm. Comput Eng Appl 48(10):233–237. https://doi.org/10.3778/j.issn.1002-8331.2012.10.053

Tang DC, Li LS, Du K (2006) On the development path of chinese manufacturing industry based on resource restraint. Jiangsu Soc Sci 4:51–58. https://doi.org/10.13858/j.cnki.cn32-1312/c.2006.04.013

Tay JC, Ho NB (2008) Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput Ind Eng 54(3):453–473. https://doi.org/10.1016/j.cie.2007.08.008

Xie NM, Chen NL (2018) Flexible job shop scheduling problem with interval grey processing time. Appl Soft Comput 70:513–524. https://doi.org/10.1016/j.asoc.2018.06.004

Ya K (2013) Empirical study of China's manufacturing enterprise development and carbon emissions. Dissertation, Tianjin University of Technology

Yin LJ, Li XY, Gao L, Liu C, Zhang Z (2017) A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. Sustain Comput Inform Syst 13(1):15–30. https://doi.org/10.1016/j.suscom.2016.11.002

Zanoni S, Bettoni L, Glock CH (2014) Energy implications in a two-stage production system with controllable production rates. Int J Prod Econ 149(149):164–171. https://doi.org/10.1016/j.ijpe.2013.06.025

Zeng LL, Zou FX, Xu XH, Gao Z (2009) Dynamic scheduling of multi-task for hybrid flow-shop based on energy consumption. In: Proceedings of the 2009 IEEE international conference on information and automation. https://doi.org/10.1109/ICINFA.2009.5204971

Zhang H, Zhao F, Fang K, Sutherland JW (2014) Energy-conscious flow shop scheduling under time-of-use electricity tariffs. CIRP Ann Manuf Technol 63(1):37–40. https://doi.org/10.1016/j.cirp.2014.03.011