Proceedings of the 2019 IEEE/SICE
International Symposium on System Integration
Paris, France, January 14-16, 2019

Automatic Design of Dispatching Rules for Real-time Optimization of Complex Production Systems*

Zhongshun Shi, Siyang Gao, Jianzhong Du, Hang Ma and Leyuan Shi

Abstract— **Dispatching rules have proven to be very useful in practice for real-time optimization of production systems. Due to the complex and dynamic nature of manufacturing environment, there are no universal rules that are superior in all situations. The task of manually designing dispatching rules is not trivial and time consuming. Recently, genetic programming becomes a popular method to facilitate the design task by automatically generating dispatching rules. This paper develops an efficient hybrid method, which achieves an automatic design of dispatching rules for real-time optimization of complex production systems. We first formulate the evaluation problem as a ranking and selection problem, and then propose a new evaluation method for genetic programming by simulation optimization. The proposed method can be applied for evaluation of candidate heuristics to enhance the search efficiency of genetic programming. Numerical results show the efficiency and effectiveness of the proposed method.**

I. INTRODUCTION

Job shop environment is stochastic and dynamic. Unexpected events make it almost impossible to determine the optimality of a schedule until all arriving jobs have been processed, which suggests that the classical mathematical optimization models should not be appropriate for dynamic job shop scheduling (DJSS). As an alternative, researchers have tried to find various heuristic methods for DJSS problems. A dispatching rule is a low-level heuristic that determines the next job to be processed by a machine when it becomes available at a decision point. Dispatching rules can be applied for both static and dynamic JSS problems. Unfortunately, there are no universal rules that are better performances in all environments. Due to the complex and dynamic nature of scheduling problems, designing effective dispatching rules usually requires many times of trial-and-error and extensive problem domain knowledge. Recent advances in machine learning and artificial intelligence have offered some novel perspectives to facilitate the design process. Genetic programming (GP), an evolutionary computation (EC) method, is a promising method for generating complicated and robust dispatching rules for different scheduling problems.

With its ability of learning executable data structures, flexible representations and powerful search ability, GP is capable of evolving effective heuristics and thus dealing with various types of scheduling problems. The basic idea of this approach is not complicated. In GP, a population of individuals (dispatching rules), represented by the sets of terminals (i.e., attributes of jobs/machines/shop floor) and functions (e.g., arithmetic/logical/mathematical operators), is initialized. Discrete event simulation (DES) is then used to assess the performance of the candidate dispatching rules. The fitness values obtained by the rules in the population determine the probability that each rule survives and reproduces in the next generation.

The hyper-heuristics generation of dispatching rules can be summarized as dispatching rule *generation* and *evaluation*. Previous studies have shown the superiority of the dispatching rules evolved by GP. Most of the existing research focuses on the representations and search mechanisms of GP and the experimental results show that they can play an important role for the quality of the evolved rules. Another important issue in GP is the rule evaluation. Generally, evaluation of the dispatching rules can be treated as a simulation optimization problem as the fitness of heuristics is stochastic. Due to the lack of exploitation in early stages of GP, some good programs can easily be ignored if not evaluated accurately. In this research, we treat the evaluation problem as a ranking and selection (R&S) problem. Under this formulation, a new evaluation method is proposed to improve the accuracy by assessing the relative ranking as correctly as possible. Next, we propose a new GP framework for automatic design of dispatching rules. The proposed framework improves the efficiency of GP and can be used as a general framework for other hyper-heuristic generation methods.

II. BACKGROUND

Job shop scheduling is a classical operations research problem, which is NP-hard in the strong sense. Even for a very small instance, an optimal solution cannot be guaranteed. The first formulations of the problem can be found in Akers and Friedman [1]. The basic job shop scheduling problem can be described as follows. There is a job set J to be processed on a machine set M. Each job j is composed of several operations O_j that must be processed on the machines. Let $O = \{(j,i)|j \in J, i \in O_j\}$. Each operation uses one of the machines for a specified processing time, p_{ji}. Each machine k can process at most one operation at a time. Once an operation starts on a given machine, it must be completed on that machine without interruption. The operations of a given job have to be processed in a

Z. Shi and H. Ma are with the Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA (e-mail: zhongshun.shi@wisc.edu, hma66@wisc.edu).

S. Gao and J. Du are with the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China (e-mail: siyangao@cityu.edu.hk, jianzhodu2-c@my.cityu.edu.hk).

L. Shi is with the Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI, 53706 USA, and the Department of Industrial Engineering and Management, Peking University, Beijing 100871, China (e-mail: leyuan.shi@wisc.edu).

given order. Let $A_j = \{(O_{ji_1}, O_{ji_2}) | j \in J, i_1 < i_2\}$ denote the precedence relationship between operations of job j. Let $E_k = \{O_{ji^k} | k \in M\}$ denote the set of operations processed on machine k. C_{ji} and S_{ji} are the start time and stop time, respectively. The objective is to find a schedule of the operations on the machines to minimize/maximize a certain objective function.

Over the past few decades, a large number of methods have been applied to JSP, ranging from simple heuristics to artificial intelligence and mathematical techniques. Given the complexity of JSP, it is very time-consuming to find its optimal solution, especially for large-scale problems. The research on meta-heuristic, on the other hand, has been very active. The common meta-heuristic methods are Genetic algorithm, Estimation of distribution algorithm, Firefly algorithm, Biogeography-based optimization algorithm, Shuffled frog-leaping algorithm, Tabu search, Differential evolution, Particle swarm optimization, Simulated annealing [2], Iterated greedy [3] and Hybrid algorithm.

Although there have been a lot of breakthroughs in exact and approximate approaches for JSP, these methods primarily focused on static problems and simplified job shop environments, and cannot be applied to a dynamic setting, which is usually the case in practice. In addition, real-world problems have more complex constraints and a large number of jobs which cause low computation efficiency when adopting both exact and meta-heuristic methods.

As a solution technique for handling the challenge, GP has become one of the most popular methods in hyper-heuristics for production scheduling and other fields [4]. Since GP is capable of evolving complex program or rules, it can be used to solve different types of production scheduling problems, e.g., [5], [6], [7].

In the implementation of GP, a major problem is that it might not achieve sufficient exploitation when dealing with complex job shop scheduling problems, and this problem can be alleviated by efficiently utilizing the computing resources and accurately estimating the ranking of the evolved rules in a population. In this paper, we will follow this direction and develop a new ranking and selection method for evaluating the design of dispatching rules generated by GP.

III. PROPOSED APPROACH

In this section, we first discuss the issue of fitness evaluation in GP. We treat the fitness evaluation as a simulation optimization problem. Then, we propose a new ranking and selection mechanism to improve the accuracy of fitness evaluation. Finally, we propose a new framework of GP and discuss its application.

A. Fitness Evaluation

Genetic programming is an evolutionary method which iteratively produces new populations of individuals to find good individuals. The first step is to choose proper representation for GP. It determines the whole search space of hyper-heuristics in GP. Then, the main generational loop of a run of genetic programming consists of fitness evaluation, Darwinian selection, and the genetic operations. They determine the search capability of GP in identifying better individuals.

Fitness evaluation plays a key role in genetic programming execution. The quality of the solutions generated by the rules determines its fitness, which in turn guides the search behavior of GP. Previous studies on automatic design of dispatching rules (DRs) have proposed some methods for fitness evaluation. Branke et al. [8] suggested that it is beneficial to use the same 10 random number seeds to evaluate all individuals in GP optimization to reduce the impact of stochasticity. However, current research on evaluation needs to be extended for automatic design of dispatching rules.

In this paper, we propose a new GP framework which automatically designs dispatching rules for DJSS problems. The basic idea is to utilize the simulation optimization technique to guide the search of GP. We first propose a ranking-and-selection method to adaptively allocate the simulation budget to candidate DRs, in order to accurately assess their relative ranking. Next, we will present a framework of GP for presentation.

B. A New R&S method for Evaluating Dispatching Rules

Suppose that there are n simulation samples available at each generation of GP. We consider how to efficiently allocate these n samples to the k dispatching rules so as to rank them as correctly as possible.

We first define the quality of rules by the relative deviation of its objective value from its corresponding reference objective value by the following equation:

$$dev(H_i, I_n) = \frac{Obj(H_i, I_n) - Ref(I_n)}{Ref(I_n)},$$

which is the same as the definition in [7]. In this equation, $Obj(H_i, I_n)$ is the objective value when we apply heuristic H_i to instance I_n, and $Ref(I_n)$ is the reference objective value for instance In. The fitness of heuristic H_i on the training set is determined by the following equation. The fitness function $dev_{average}(H_i)$ will measure the average performance of H_i

$$dev_{average}(H_i) = \frac{1}{T} \sum_{I_n \in \{I_1, \cdots, I_T\}} dev(H_i, I_n).$$

The input instance I_n is stochastic, i.e., $I_n \sim \mathcal{D}$, where $\mathcal{D}$ denote the input parameter distribution.

Suppose there is a total of n simulation samples and k DRs. $X_{i,j}$ is the j-th simulation replication for DR i. μ_i and σ_i^2 are the mean and variance of $X_{i,j}$, and $\bar{X}_i$ and S_i^2 are the sample mean and sample variance. n_i is the number of simulation samples received by DR i in the selection procedure. DR t is the best DR. δ is an indifference parameter. We let $S_0 := \{i \in \{1, 2, ..., k\}; \mu_i < \mu_t + \delta\}$, $S_1 := S_0 \backslash \{t\}$ and $S_2 := \{i \in \{1, 2, ..., k\}; \mu_i \geq \mu_t + \delta\}$.

Assumption 1: There is no DR i whose mean performance satisfies $\mu_i = \mu_t + \delta$, $i = 1, 2, ..., k$.

Assumption 2: The simulation output samples are normally distributed and are independent across different DRs and replications.

Here, we introduce an indifference parameter δ to adjust the candidate subset of DRs. The goal of this sample allocation rule is to correctly select all the DRs in S_0 and correctly rank all the DRs in S_0. Suppose there are k_0 DRs in S_0. Without loss of generality, we assume $\mu_1 < \mu_2 < ... < \mu_{k_0}$ for simplicity of notation. Note that the best DR becomes $t = 1$.

A correct selection happens when $\forall i \in S_0$, $\bar{X}_1 < \bar{X}_2 < ... < \bar{X}_{k_0} < \mu_1 + \delta$, and $\forall j \in S_2$, $\bar{X}_j \geq \mu_1 + \delta$. Therefore, the probability of correct selection (PCS) is given by $PCS = P\big(\bigcap_{i=1}^{k_0-1}(\bar{X}_i < \bar{X}_{i+1}) \cap (\bar{X}_{k_0} < \mu_1 + \delta) \cap \bigcap_{j \in S_2}(\bar{X}_j \geq \mu_1 + \delta)\big)$. To facilitate the theoretical development and the computation, in this research, we identify a lower bound of PCS as an approximation of it. Note that there exist parameters $c_1, c_2, ..., c_{k_0}$ such that $\mu_1 < c_1 < \mu_2 < c_2 < ... < c_{k_0-1} < \mu_{k_0} < c_{k_0}$. For the purpose of selection, we let $c_{k_0} = \mu_1 + \delta$, and c_i can be any real number between μ_i and μ_{i+1} for $i = 1, 2, ..., k_0 - 1$, but for the convenience and effectiveness of the implementation of the allocation rule, we let $c_i = \frac{1}{2}(\mu_i + \mu_{i+1})$, $i = 1, 2, ..., k_0 - 1$.

We have,

$$\begin{aligned} PCS =& P\Bigg(\bigcap_{i=1}^{k_0-1}(\bar{X}_i < \bar{X}_{i+1}) \cap (\bar{X}_{k_0} < \mu_1 + \delta) \\ & \qquad \cap \bigcap_{j \in S_2}(\bar{X}_j \geq \mu_1 + \delta)\Bigg) \\ \geq& 1 - P(\bar{X}_1 \geq c_1) - \sum_{i=2}^{k_0} P(\bar{X}_i \leq c_{i-1}) \\ & - \sum_{i=2}^{k_0} P(\bar{X}_i \geq c_i) - \sum_{j \in S_2} P(\bar{X}_j < c_{k_0}). \end{aligned} \tag{1}$$

We call the lower bound in (1) APCS†, and consider the following sample allocation problem

$$\max_{n_1,...,n_k} APCS^\dagger \quad s.t. \sum_{i=1}^{k} n_i = n, \; n_i \geq 0, \; i = 1, 2, ..., k. \tag{2}$$

To solve the optimization problem (2), we introduce two lemmas.

Lemma 1 ([9]): Consider $u_i, v_i, w_j, z_j \in \mathbb{R}$ with $u_i, w_j > 0$ and $v_i, z_j < 0$, $i = 1, 2, ..., m_1$ and $j = 1, 2, ..., m_2$. $v_i \neq v_{i'}$ for $i, i' \in \{1, 2, ..., m_1\}$ and $z_j \neq z_{j'}$ for $j, j' \in \{1, 2, ..., m_2\}$. If

$$\sum_{i=1}^{m_1} u_i \exp(v_i x) = \sum_{j=1}^{m_2} w_j \exp(z_j x) \tag{3}$$

as $x \to +\infty$, then, $v_{i_v} = z_{j_z}$, where $i_v = \arg\max_{i \in \{1,...,m_1\}} v_i$ and $j_z = \arg\max_{j \in \{1,...,m_2\}} z_j$.

Lemma 2: Problem (2) is a convex optimization problem.

Proof: In order to show the convexity of the optimization problem (2), we only need to show that $APCS^\dagger$ is a concave function, i.e., $P(\bar{X}_1 \geq c_1)$, $P(\bar{X}_i \leq c_{i-1})$, $P(\bar{X}_i \geq c_i)$ and $P(\bar{X}_j < c_{k_0})$ are convex for all $i \in \{2, 3, ..., k_0\}$ and $j \in \mathcal{S}_2$.

Let Φ be the standard normal cumulative distribution function and $\nu_1 = \sigma_1/(\mu_1 - c_1)$. Note that $\nu_1 < 0$. We consider the term $P(\bar{X}_1 \geq c_1)$.

$$\frac{\partial P(\bar{X}_1 \geq c_1)}{\partial n_1} = \frac{1}{2\sqrt{2\pi}} \exp\left\{-\frac{n_1}{2\nu_1^2}\right\} \frac{1}{\nu_1\sqrt{n_1}}, \tag{4}$$

$$\begin{aligned} \frac{\partial^2 P(\bar{X}_1 \geq c_1)}{\partial n_1^2} &= \frac{-1}{4\sqrt{2\pi}\nu_1} \exp\left\{\frac{-n_1}{2\nu_1^2}\right\} \left(\frac{1}{\nu_1^2\sqrt{n_1}} + n_1^{-\frac{3}{2}}\right) \\ &> 0. \end{aligned} \tag{5}$$

Then, $P(\bar{X}_1 \geq c_1)$ is a convex function. The convexity of $P(\bar{X}_i \leq c_{i-1})$, $P(\bar{X}_i \geq c_i)$ and $P(\bar{X}_j < c_{k_0})$ can be similarly obtained by investigating their derivatives as in (4) and (5). ■

Lemma 1 captures the asymptotic behavior for equations with summation of exponential terms. Lemma 2 shows that (2) is a convex optimization problem. As a result, this problem can be solved by checking the KKT conditions of it.

Theorem 1: Problem (2) is asymptotically optimized if

$$\frac{n_i}{n_j} = \frac{\beta_i}{\beta_j}, \quad i, j \in \{1, 2, ..., k\} \text{ and } i \neq j, \tag{6}$$

where

$$\beta_i = \begin{cases} \frac{\sigma_1^2}{(\mu_1 - c_1)^2}, & \text{if } i = 1; \\ \frac{\sigma_i^2}{\min\{(\mu_i - c_{i-1})^2, (\mu_i - c_i)^2\}}, & \text{if } i = 2, 3, ..., k_0; \\ \frac{\sigma_i^2}{(\mu_i - c_{k_0})^2}, & \text{if } i \in \mathcal{S}_2. \end{cases} \tag{7}$$

Proof: Let $\mathcal{F}^\dagger$ be the Lagrangian relaxation of (2) with Lagrange multiplier γ. That is,

$$\begin{aligned} \mathcal{F}^\dagger =& 1 - P(\bar{X}_1 \geq c_1) - \sum_{i=2}^{k_0} P(\bar{X}_i \leq c_{i-1}) - \\ & \sum_{i=2}^{k_0} P(\bar{X}_i \geq c_i) - \sum_{j \in \mathcal{S}_2} P(\bar{X}_j < c_{k_0}) - \gamma(\sum_{i=1}^{k} n_i - n). \end{aligned}$$

The KKT conditions of (2) are given by

$$\frac{\partial \mathcal{F}^\dagger}{\partial n_1} = -\frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(\mu_1 - c_1)^2}{2\sigma_1^2} n_1\right) \frac{c_1 - \mu_1}{\sigma_1\sqrt{n_1}} - \gamma = 0; \tag{8}$$

$$\begin{aligned} \frac{\partial \mathcal{F}^\dagger}{\partial n_i} =& -\frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(\mu_i - c_i)^2}{2\sigma_i^2} n_i\right) \frac{\mu_i - c_i}{\sigma_i\sqrt{n_i}} - \\ & \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(\mu_i - c_{i-1})^2}{2\sigma_i^2} n_i\right) \frac{c_{i-1} - \mu_i}{\sigma_i\sqrt{n_i}} - \gamma = 0, \\ & i = 2, 3, ..., k_0; \end{aligned} \tag{9}$$

$$\begin{aligned} \frac{\partial \mathcal{F}^\dagger}{\partial n_j} =& \frac{-1}{2\sqrt{2\pi}} \exp\left(-\frac{(\mu_j - c_{k_0})^2}{2\sigma_j^2} n_j\right) \frac{c_{k_0} - \mu_j}{\sigma_j\sqrt{n_j}} - \gamma = 0, \\ & j \in \mathcal{S}_2. \end{aligned} \tag{10}$$

We consider the following two cases.

Case 1. Comparison of DR 1 and DR i, $i \in \{2, 3, ..., k_0\}$. We analyze the number of samples for DR 1 and DR i. From (8) and (9), for $i = 2, 3, ..., k_0$,

$$\exp\left(-\frac{(\mu_1 - c_1)^2}{2\sigma_1^2} n_1\right) \frac{c_1 - \mu_1}{\sigma_1 \sqrt{n_1}} = \exp\left(-\frac{(\mu_i - c_i)^2}{2\sigma_i^2} n_i\right) \cdot \frac{\mu_i - c_i}{\sigma_i \sqrt{n_i}} + \exp\left(-\frac{(\mu_i - c_{i-1})^2}{2\sigma_i^2} n_i\right) \frac{c_{i-1} - \mu_i}{\sigma_i \sqrt{n_i}}. \quad (11)$$

From Lemma 1, (11) asymptotically holds when

$$\frac{(\mu_1 - c_1)^2}{2\sigma_1^2} n_1 = \min\left(\frac{(\mu_i - c_i)^2}{2\sigma_i^2} n_i, \frac{(\mu_i - c_{i-1})^2}{2\sigma_i^2} n_i\right),$$

for $i = 2, 3, ..., k_0$. That is,

$$\frac{n_1}{n_i} = \frac{\sigma_1^2/(\mu_1 - c_1)^2}{\sigma_i^2/\min((\mu_i - c_i)^2, (\mu_i - c_{i-1})^2)}, \quad i = 2, 3, ..., k_0. \quad (12)$$

Case 2. Comparison of DR 1 and DR j, $j \in \mathcal{S}_2$. We next analyze the number of samples for DR 1 and DR j. From (8) and (10),

$$\frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(\mu_1 - c_1)^2}{2\sigma_1^2} n_1\right) \frac{c_1 - \mu_1}{\sigma_1 \sqrt{n_1}} = \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{(\mu_j - c_{k_0})^2}{2\sigma_j^2} n_j\right) \frac{c_{k_0} - \mu_j}{\sigma_j \sqrt{n_j}}, \quad j \in \mathcal{S}_2. \quad (13)$$

From Lemma 1, (13) asymptotically holds when

$$\frac{(\mu_1 - c_1)^2}{2\sigma_1^2} n_1 = \frac{(\mu_j - c_{k_0})^2}{2\sigma_j^2} n_j, \quad j \in \mathcal{S}_2.$$

That is,

$$\frac{n_1}{n_j} = \frac{\sigma_1^2/(\mu_1 - c_1)^2}{\sigma_j^2/(\mu_j - c_{k_0})^2}, \quad j \in \mathcal{S}_2. \quad (14)$$

Combine (12) and (14), and the result in (6) and (7) follows. ■

C. Basic Procedure of RSGP framework

The proposed ranking and selection-based GP (RSGP) is shown in Figure 1. GP starts its evolution process with a randomly generated population. The most commonly used approach to generating tree is called the *ramped-half-and-half* method [10], combining two of the simplest tree generation approaches grow and full with equal probability. The GP system then evaluates the current population of candidate rules. The performance of each rule will be estimated by simulation samples. Here, we apply the proposed R&S method to provide relative ranking of the current population. Each DR is initially simulated with n_0 replications in the first iteration. Additional Δ replications are increased in each iteration and assigned to selected DR until the total computing budget is exhausted. After that, we add an additional step before genetic operations are executed. We use memory to record the evolved rules and their statistical performance. Those DRs with different mean of objectives will be inserted into the candidate set of DRs. Then, tournament selection is performed to select parent DR based on their relative ranking for the following genetic operations: crossover, mutation, and reproduction. GP runs until a termination condition holds, which is normally when a predefined number of generations is met. In the final stage, we perform the proposed R&S procedure to select a subset of the top dispatching rules as the output of GP. The set of evolved rules can be applied to DJSS problems.

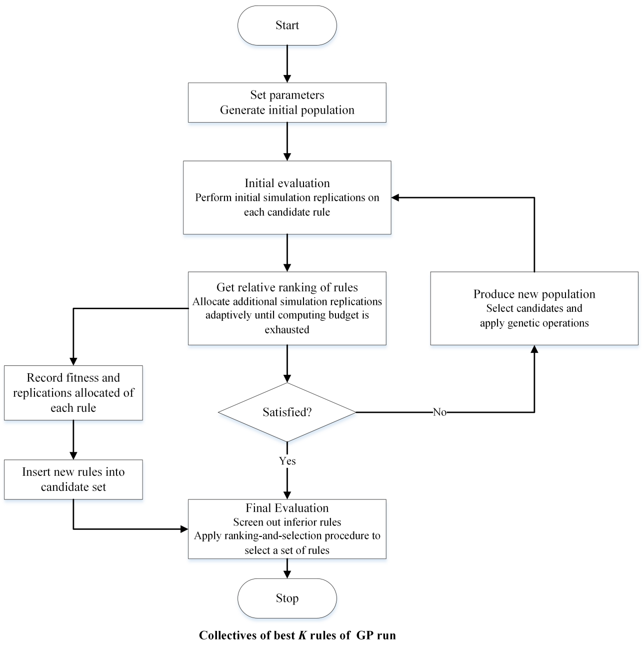
Start
Set parameters
Generate initial population
Initial evaluation
Perform initial simulation replications on each candidate rule
Get relative ranking of rules
Allocate additional simulation replications adaptively until computing budget is exhausted
Produce new population
Select candidates and apply genetic operations
Record fitness and replications allocated of each rule
Satisfied?
No
Yes
Insert new rules into candidate set
Final Evaluation
Screen out inferior rules
Apply ranking-and-selection procedure to select a set of rules
Stop
Collectives of best K rules of GP run

Fig. 1. The proposed procedure of RSGP framework

RSGP can also be used as a general method for other hyper-heuristic generation problems. This approach provides a new way to increase the search efficiency of GP under a fixed simulation budget.

IV. Experiment Design

In this section, we present the simulation model to train and test GP. Then, we discuss the configuration of the GP system.

A. Simulation model

All experiments in this paper are based on simulation of the dynamic job shop environment, which has been widely used in the literature of dispatching rules [11] [12] [8]. In our simulation configuration, 10 machines are on the shop floor. A random routing is assigned to each job entering the shop. The processing times follow a discrete uniform distribution U[1, 49], ranging from 1 to 49 minutes. Job arrivals follow a Poisson process. Job arrivals follow a Poisson process. Utilization of the shop is set to 95%. There are machine break-down and preemption is not allowed.

At first the shop is empty, and the period until the 500th job is used as warm-up. We collect data of jobs numbering from 501 to 2500, which will be used to calculate the performance. The objective function is the mean flowtime.

TABLE I
TERMINAL AND FUNCTION SETS OF GP

Symbol	Descriptions
PT	processing time of current operation
NPT	processing time of the next operation
WINQ	work in next queue, i.e., sum of processing times of all jobs at the next machine
RPT	sum of processing times of all unfinished operations
OpsLeft	number of all unfinished operations within a job
TIQ	time since arrival in current queue (t - *ready time*)
TIS	time since arrival in system of job (t - *release time*)
Constant	0, 1
Function set	+, -, *, /, max, if-then-else

TABLE II
PARAMETER SETTING FOR PROPOSED GP SYSTEM

Population size	200
Generations	20
Crossover rate	95%
Mutation rate	5%
Elite	10
Selection Method	Tournament Selection (size 7)
Initialization	Ramped Half-and-Half (Min:2 Max:6)
Max-depth	17

B. Parameter Setting

The GP system is developed using the ECJ23 library. Table IV-B shows the terminal set and function set of GP, which were found to be the best settings for the tree representation [8]. All attributes in the terminal set have a natural lower bound, but it is not clear if they all have an upper bound. The upper bounds for related attributes were set to be the 90% quantiles encountered in preliminary simulation runs using the Holthaus rule $-(2PT + WINQ + NPT)$ [13].

The parameter setting is shown in Table II. Duplicate removal mechanism is applied during the whole GP run. Branke *et al.* [8] proposed this mechanism to efficiently suppress trees causing duplicate objective values. It increases the diversity in a GP run. We also use it to satisfy the assumption, which assumes that all the heuristics are not unique, to implement the ranking and selection method in our GP system.

For the fitness function, we choose the Holthaus rule as the reference rule, which is the best dispatching rule manually developed so far for the objective of mean flowtime in the dynamic job shop environment.

C. Benchmark methods

In order to show the effectiveness our RSGP system, we use the following benchmark dispatching rules.

1) *FIFO*: the operations are sequenced *first-in-first-out*.
2) *SPT*: the job with the shortest processing time of its current operation is processed first.
3) *ERD*: the job with the earliest release time of its current operation is processed first.
4) *EDD*: the job with the earliest due date of its current operation is processed first.
5) *SRPT*: the operation belonging to the job that has the shortest remaining processing time is processed first.
6) *WINQ*-Work In Next Queue first: jobs are ranked in the order of a (rather worst case) estimation of their waiting time before processing on the next machine can start
7) *PT+WINQ*: this rule uses the sum of WINQ and the processing time of current operation. The job where this sum is least gets the highest priority.
8) *PT+WINQ+SL*: this rule uses the sum of WINQ, the processing time of current operation and the slack of the job. The job where this sum is least gets the highest priority.
9) *2PT+WINQ+NPT*: this rule uses the sum of WINQ, twice the processing time of current operation and the processing time of a job's next operation. The job where this sum is least gets the highest priority.

Meanwhile, we will compare the proposed RSGP method with TGP [8] and TGP-OCBA to show its effectiveness. TGP is traditional tree-based GP method proposed by Branke *et al.* [8] in the literature. In TGP, each candidate DR uses fixed number of simulation replications with common random number for evaluation. TGP-OCBA method integrates optimal computing budget allocation (OCBA) rule [14] for selecting the best candidate DR in the evaluation step of TGP. We try to check if classical ranking and selection mechanism is beneficial for improving search behavior of GP. TGP, TGP-OCBA and RSGP use the same terminal and function sets as shown in table IV-B. The parameter settings for the three methods are shown in Table II. In the experiment, we set a maximum of 80000 simulation budget in a whole GP run. It means that there are 4000 replications for evaluation in each generation.

V. NUMERICAL RESULTS AND ANALYSIS

The proposed RSGP method is applied to evolve new dispatching rules under different settings. In order to compare TGP, TGP-OCBA and RSGP, 30 independent runs of each method for dynamic job shop simulation scenario are performed. In each run, the average performance of 100 testing replications of best obtained dispatching rule is recorded to evaluate the effectiveness of the above methods.

We use the tuple <minop, maxop, u> to represent the different settings of simulation scenarios. Here, *minop* is the minimum number of operations and *maxop* is the maximum number of operations of a job entering the system, i.e., the number of operations for each new job is randomly generated from the discrete uniform distribution U[minop, maxop]. The utilization level of the shop is u. We use 4 standard simulation scenarios to test performance. They are <2, 10, 85%>, <2, 10, 95%>, <10, 10, 85%>, <10, 10, 95%>, respectively. In order to check the generalization abilities of the rules, we add two additional simulation scenarios. The first one is <50, 50, 95%>, where the minimum and maximum number of operations and the number of machines

TABLE III

MEAN FLOWTIME PERFORMANCE IN DIFFERENT SIMULATION SCENARIOS

Name	<2,10,0.85>	<10,10,0.85>	<2,10,0.95>	<10,10,0.95>	<50,50,0.95>	<20,200,0.95>
FIFO	654.065(1.4566)	1031.5988(1.4616)	1427.0016(1.7943)	2273.0886(1.8344)	8850.4392(1.5839)	10396.6334(1.2524)
SPT	457.1567(1.0724)	721.6083(1.0705)	835.4561(1.1923)	1313.2966(1.1826)	6004.6565(1.1835)	8586.2644(1.2402)
ERD	626.5758(1.3954)	976.7027(1.3838)	1262.4431(1.5874)	1861.9991(1.5027)	6981.9385(1.2495)	9583.5009(1.1545)
EDD	610.4482(1.3594)	943.5266(1.3368)	1222.8437(1.5376)	1829.0099(1.476)	6948.4736(1.2435)	7735.8085(0.9319)
SRPT	618.1563(1.3766)	962.2576(1.3633)	1040.0674(1.3078)	1853.1782(1.4955)	6671.4116(1.1939)	7141.5651(0.8603)
WINQ	517.532(1.1525)	806.4785(1.1426)	980.775(1.2332)	1534.6867(1.2385)	6254.685(1.1193)	8780.4471(1.0577)
PT+WINQ	457.2701(1.0183)	721.8638(1.0227)	844.397(1.0618)	1329.0128(1.0725)	5921.5165(1.0597)	8474.6748(1.0209)
WINQ+PT+SL	478.8852(1.0665)	729.9787(1.0342)	984.9514(1.2385)	1419.2727(1.1454)	5596.1998(1.0015)	8382.2065(1.0098)
2PT+WINQ+NPT	449.0426(1.0)	705.8107(1.0)	795.2861(1.0)	1239.1405(1.0)	5587.854(1.0)	8301.2114(1.0)
TGP	426.3034(0.9494)	674.0662(0.955)	700.7196(0.8811)	1110.5003(0.8962)	5073.6556(0.908)	6923.0481(0.834)
TGP-OCBA	426.6772(0.9502)	675.9139(0.9576)	697.4559(0.877)	1107.0938(0.8934)	5071.0709(0.9075)	6717.3681(0.8092)
RSGP	**425.5625(0.9477)**	**670.7116(0.9503)**	**672.9382(0.8462)**	**1093.5101(0.8825)**	**5045.0224(0.9029)**	**6416.0669(0.7729)**

is up to 50. Another complex scenario is <20, 200, 95%>, where the number of machines is up to 200 and the number of operations is between 20 and 200. These two harder job shop scenarios could give us some sense of robustness of the evolved rules from GP.

The experimental results are shown in Table III. Here, the real mean flowtime value (minutes) and relative performance (compared to rule *2PT+WINQ+NPT* and given in brackets) are compared in 6 different simulation scenarios. In given 4 standard scenarios, rule 2PT+WINQ+NPT is the best manually designed dispatching rule. Therefore, it is a good choice as our reference rule in GP optimization. From the experimental results, we can see that the evolved rules obtained by different GP methods dominate all the benchmark dispatching rules. It means that the GP method is very effective and can be used to evolve competitive dispatching rules. Meanwhile, it is very clear that RSGP method achieves the best performance in all simulation scenarios. It improves the reference rule by 11.75% in scenario <10, 10, 95%> and 15.38% in scenario <2, 10, 95%>, respectively. TGP-OCBA performs better than TGP in those two scenarios with utilization level of 95%, but worse than TGP in those two scenarios with utilization level of 85%.

In two additional simulation scenarios of <50, 50, 95%> and <20, 200, 95%>, we can get similar results as standard scenarios. It shows that the evolved rules by GP can maintain their good performance. An interesting observation is that the RSGP method achieves the best improvement on the 2PT+WINQ+NPT rule by 22.71% in scenario <20, 200, 95%>. However, the EDD and SRPT rules perform better than the reference rule in this simulation scenario.

All the results above indicate that RSGP is capable of discovering very effective rules for each simulation scenario. Also, the results show that RSGP can indeed discover more effective rules than TGP and TGP-OCBA.

VI. CONCLUSION

This paper proposed a new evaluation approach for genetic programming. The new approach treats the evaluation of the dispatching rules as a ranking and selection problem, and provides a novel way to identify relative ranking of individuals in each generation of GP. The proposed closed-form allocation rule is integrated with GP to solve the dynamic job shop scheduling problem. The numerical results indicate that the proposed RSGP approach has the best performance among the compared methods in all simulation scenarios. Also, RSGP performs better than the traditional GP and GP with OCBA rules within finite computing budget. In general, this approach provides a new way to increase the search efficiency of GP and can be applied to solve other GP applications.

REFERENCES

[1] S. B. Akers Jr and J. Friedman, "A non-numerical approach to production scheduling problems," *Journal of the Operations Research Society of America*, vol. 3, no. 4, pp. 429–442, 1955.

[2] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operational Research*, vol. 113, no. 1, pp. 123–136, 1999.

[3] M. Pranzo and D. Pacciarelli, "An iterated greedy metaheuristic for the blocking job shop scheduling problem," *Journal of Heuristics*, vol. 22, no. 4, pp. 587–611, 2016.

[4] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*. Springer, 2009, pp. 177–201.

[5] K. Miyashita, "Job-shop scheduling with genetic programming," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2000, pp. 505–512.

[6] D. Jakobović and L. Budin, "Dynamic scheduling with genetic programming," *Genetic Programming*, pp. 73–84, 2006.

[7] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[8] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: A comparison of rule representations," *Evolutionary Computation*, vol. 23, no. 2, pp. 249–277, 2015.

[9] S. Gao and W. Chen, "Efficient feasibility determination with multiple performance measure constraints," *IEEE Transactions on Automatic Control*, vol. 62, pp. 113–122, 2017.

[10] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[11] C. Rajendran and O. Holthaus, "A comparative study of dispatching rules in dynamic flowshops and jobshops," *European journal of operational research*, vol. 116, no. 1, pp. 156–170, 1999.

[12] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2010, pp. 257–264.

[13] O. Holthaus and C. Rajendran, "Efficient jobshop dispatching rules: further developments," *Production Planning & Control*, vol. 11, no. 2, pp. 171–178, 2000.

[14] C. H. Chen, J. Lin, E. Ycesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems*, vol. 10, no. 3, pp. 251–270, 2000.