A Hyper-heuristic-Based Ensemble Genetic Programming Approach for Stochastic Resource Constrained Project Scheduling Problem

HaoJie Chen [1], Guofu Ding [1], Shengfeng Qin[2], Jian Zhang[1]

HaoJie Chen
e-mail: chenhaojie12138@163.com
Guofu Ding
e-mail: dingguofu@163.com
Shengfeng Qin
e-mail: sheng-feng.qin@northumbria.ac.uk
Jian Zhang(✉Correspondence author)
e-mail:jerrysmail@263.net

1. **School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China**
2. **Department of Design, Northumbria University, Newcastle upon Tyne NE1 8ST, UK**

Abstract: In project scheduling studies, to the best of our knowledge, the hyper-heuristic collaborative scheduling is first-time applied to project scheduling with random activity durations. A hyper-heuristic-based ensemble genetic programming (HH-EGP) method is proposed for solving stochastic resource constrained project scheduling problem (SRCPSP) by evolving an ensemble of priority rules (PRs). The proposed approach features with (1) integrating the critical path method into the resource-based policy class to generate schedules; (2) improving the existing single hyper-heuristic project scheduling research to construct a suitable solution space for solving SRCPSP; and (3) bettering genetic evolution of each subpopulation from a decision ensemble with three different local searches in corporation with discriminant mutation and discriminant population renewal. In addition, a sequence voting mechanism is designed to deal with collaborative decision-making in the scheduling process for SRCPSP. The benchmark PSPLIB is performed to verify the advantage of the HH-EGP over heuristics, meta-heuristics and the single hyper-heuristic approaches.
Key words: Ensemble decision; Genetic programming; Hyper-heuristics; Priority rule; Stochastic resource constrained project scheduling

1 Introduction

As one of the most important problems in project management, resource constrained project scheduling problem (RCPSP) involves planning a set of activities while considering various resource and precedence constraints. It is an NP-hard problem (Blazewicz, Lenstra, & Kan, 1983) with a wide range of industrial application scenarios, such as semiconductor wafer fabrication (Wang, Zhang, & Wang, 2018), software development (Chen & Zhang, 2013), and aircraft assembly (Shan et al., 2017). And most often, the objective of RCPSP is to minimise the project makespan (Arkhipov, Battaïa, & Lazarev, 2019; Hartmann & Briskorn, 2010), i.e., the completion time of the last activity in the project.

There is a body of research on the problem solving methods and optimisation strategies. In general, they can be categorised into the exact algorithms and heuristics-based approaches. The exact algorithms (Brucker, Knust, Schoo, & Thiele, 1998; Chakrabortty, Sarker, & Essam, 2015; Moukrim, Quilliot, & Toussaint, 2015) can effectively solve small-scale RCPSP, but, generally speaking, their scalability is poor; in other words, their solution time will become unacceptable when the problem scale increases. For solving large-scale RCPSP problems, heuristics-based approaches have been widely used to obtain approximate optimal solutions with a rapid scheduling process. In general, the heuristics-based methods can be divided into constructive and meta-heuristic methods. The former relies on schedule generation scheme (SGS), which is often classified as serial and parallel (Kolisch, 1996a), and determines an execution sequence of activities through a large number of PRs, such as the most total successors and the latest finish time (Kolisch, 1996b). In contrast, the latter iteratively optimises an initial population produced randomly or according to a certain strategy to obtain an optimal solution. The meta-heuristic methods usually have powerful global search capabilities, including simulated annealing (Boctor, 1996), genetic algorithm (Kadri & Boctor, 2018; Mendes, Gonçalves, & Resende, 2009), particle swarm optimisation (Jia & Seo, 2013), and hybrid algorithm (Bettemir & Sonmez, 2015; Chen et al., 2010).

An important common assumption of the above studies is that the duration of an activity is

deterministic and unvaried during the project implementation, although this assumption is difficult to maintain in the real world. During an industrial production process, there exist many disturbance factors on a planned activity, including machine faults, resource shortages and the most common factor - human influences, which lead to some deviations of an activity's duration from its expected ideal value. For example, in an assembly production process, workers, tools and other resources required by an assembly operation are fixed, but the duration of this operation can be varied due to factors such as human emotion and proficiency. In order to deal with this duration time uncertainty, there are two common strategies for RCPSP, namely proactive/reactive scheduling (Lamas & Demeulemeester, 2016) and stochastic scheduling (Chen et al., 2018). The former does not violate the concept of activity duration as a value, but either by adding activity buffers based on estimation (proactive scheduling), or repairing illegal schedule caused by dynamic factors through the updated information (reactive scheduling) (Davari & Demeulemeester, 2019). The latter solves a stochastic resource constrained project scheduling problem (SRCPSP) extended from RCPSP, and it adopts a distribution to replace a fixed activity duration and has gained more and more attention in recent years. In SRCPSP, the most commonly used objective function becomes the minimum expected makespan, leading its solution as a scheduling policy rather than a schedule itself (Möhring, Radermacher, & Weiss, 1985).

Currently, most existing SRCPSP methods are mainly based on meta-heuristics or priority heuristics. To our best knowledge, there is a lack of hyper-heuristic method in SRCPSP, which incorporates learning mechanisms into search methods for selecting or generating heuristics to solve combinatorial optimisation problems effectively in a heuristics computational time (Burke et al., 2013). The hyper-heuristics approach is supposed to have better optimisation ability and fast response ability, which are very important for most common scheduling scenarios, especially when solving dynamic problems. And its effectiveness has been verified in job shop scheduling (Nguyen et al., 2012) and static project scheduling (see Section 2.3 for details).

In this paper, a new hyper-heuristic-based ensemble genetic programming framework (HH-EGP) is proposed for generating a decision ensemble that contains a set of evolved PRs. It controls the evolution of multiple subpopulations (single hyper-heuristics) for selecting optimal PRs to form a decision ensemble and to achieve collaborative decision scheduling. Thus, it can achieve better scheduling than traditional PRs, meta-heuristics and the single hyper-heuristic methods by evolving PRs under a heuristic computing time. Compared with the existing research, the main contributions of this study are as follows:

1. An HH-EGP framework to effectively evolve a decision ensemble with multiple PRs for solving SRCPSP and realise the benefits of applying the hyper-heuristic scheduling and ensemble scheduling technology in the SRCPSP problem for the first time.
2. Design of a sequence voting mechanism to achieve collaborative decision-making in SRCPSP.
3. Modification of the basic components in existing hyper-heuristic studies and improvement of the subpopulation evolution process.

Section 2 first shows the problem description and related mathematical definition of SRCPSP. and then more specifics on the related work, the motivation of this study and its novelties and differences compared with related work are described. Section 3 introduces some modified basic components in HH-EGP, including policy class, PR coding structure, attribute/terminal set and function set. The evolution process of decision ensemble and subpopulations in HH-EGP and the

designed collaborative decision mechanism are described in Section 4. In Section 5, experiments and results are given to verify the effectiveness and superiority of this method. In Section 6, conclusions are drawn with discussions on the potential merit and future research directions of HH-EGP.

2 Problem definition and related work

2.1 Definition and problem description of SRCPSP

The description of SRCPSP is based on RCPSP, which contains a project represented by a directed acyclic graph $G(V, E)$, and its structure is shown in Fig.1 with 12 activities as an example. The nodes $V = \{0, 1, \ldots, n+1\}$ represent $n+2$ activities, in which activity 0 and activity $n+1$ are dummy activities to indicate the beginning and end of the project. The arrowed connection (E) between two activities represents the precedence relationships between activities. In the process of project implementation, $|K|$ renewable resources are required and each resource has a constant maximum supply R_k. Each activity i of SRCPSP has two important attributes: a fixed set of resource requirements and an activity duration subject to a known distribution over a range. Suppose r_{ik} and d_i are the demand of activity i for resource k and its duration, respectively. $P(B)$ represents the occurrence probability of event B to measure the duration randomness. Let S_i and PS_i represent the successors and the predecessors of activity i, and st_i refers to the start time of activity i. The longest path from the beginning to the end of the project is called the critical path, as shown by the red dotted arrow in Fig.1. It is worth mentioning that the critical path in the same $G(V, E)$ may change due to the duration uncertainty. The mathematical model of SRCPSP is as follows:

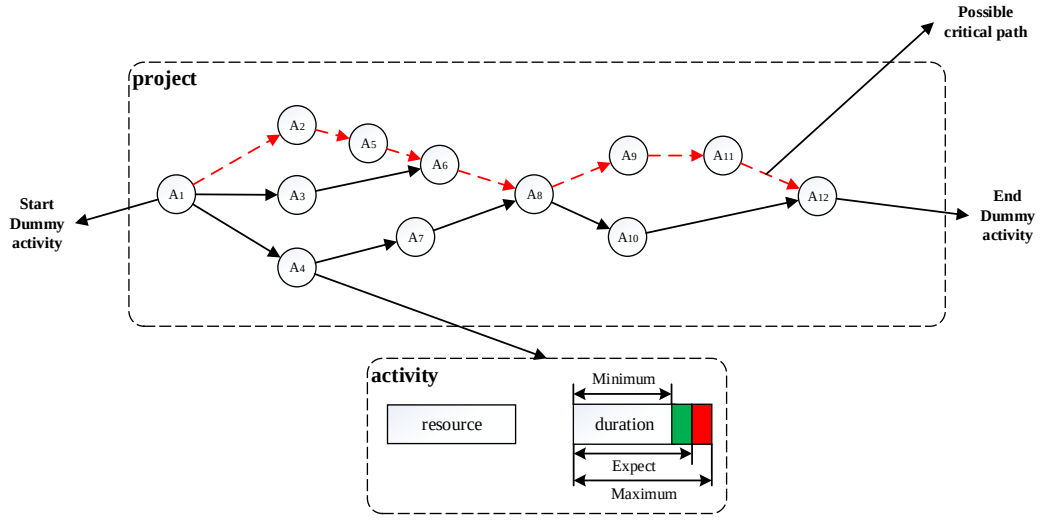


Fig.1 The structure of SRCPSP

Objective:

$$\min \ devi = \left(\sum_{ins=1}^{totalIns} \frac{E(\boldsymbol{d})_{ins} - CP^{*}_{ins}}{CP^{*}_{ins}} \right) / totalIns \times 100\% \tag{1}$$

S.t:

$$st_j - st_i \geq d_i \quad \forall i \in V, \forall j \in S_i \quad or \quad st_i - st_j \geq d_j \quad \forall i \in V, \forall j \in PS_i \tag{2}$$

$$\sum_{i \in A_t} r_{ik} \leq R_k \quad \forall k \in K, \forall t \in T \tag{3}$$

$$r_0 = r_{n+1} = 0; P(d_0 = 0) = P(d_{n+1} = 0) = 1 \tag{4}$$

$$P(d_i \leq 0) = 0 \tag{5}$$

Referring to Chen et al. (2018), the objective of this paper is to minimise the average percentage deviation of $E(\boldsymbol{d})$ (expected makespan) and CP^* (the critical path using the expected durations d^*) over multiple instances shown in Eq.(1), where *totalIns* represents the total number of instances to be scheduled, and *ins* is the *ins*th instance. In addition, Eq.(2) represents the precedence constraint, i.e. all activities cannot start until their predecessors are completed. Eq.(3) is the resource constraint, that is, the sum of the required resources of all activities in the execution state cannot exceed the maximum supply at any time, where t is the time point, A_t is the set of the executed activities at time t, and T is a maximum allowed time point. Eq.(4) indicates that the duration of the dummy activity is 0 and no resources are required, and Eq.(5) shows that any activity duration cannot be negative.

2.2 Existing scheduling approach for SRCPSP

In the existing SRCPSP methods, the precise algorithm is very rare, only including, to our knowledge, a branch-and-bound (Stork, 2001) and a Markov chain (Creemers, 2015), and most studies focus on priority heuristics or meta-heuristics scheduling, which are divided into two parts. In the first part, an activity sequence will be produced through their priority mechanism, also known as a solution. In priority heuristics, an activity sequence is generated based on activity priorities, while, in meta-heuristics, it is produced by iteration. Then, in the second part, a policy is adopted to transform this solution into activity start time without violating precedence and resource constraints. Thus, how to sort activities and implement policies is very important in solving SRCPSP.

2.2.1 How to sort activities with heuristics or meta-heuristics

The first key point to solve SRCPSP is how to sort activities, i.e., the ordering of activity scheduling. A lot of methods have been proposed, and most of them are more inclined to meta-heuristic method. Golenko-Ginzburg and Gonik (1997) proposed a meta-heuristic algorithm composed of three sub-algorithms, whose functions are control, calculation and selection, respectively. Tsai and Gemmill (1998) described a tabu search algorithm combining multiple tabu lists and randomised short-term memory for this purpose. Ballestin and Leus (2009) designed a greedy randomised adaptive search with descriptive sampling. Chakrabortty, Sarker, and Essam (2017) combined six heuristics into a robust optimisation model to find higher quality solutions. Fang et al. (2015) proposed an estimation of distribution algorithm using permutation-based local search, and their experiments show that the algorithm has a clear dominance in the middle or high distribution. In addition, evolutionary algorithm and swarm intelligence are gradually applied.

Ashtiani, Leus, and Aryanezhad (2011) introduced a genetic algorithm and Tahooneh and Ziarati (2011) applied the artificial bee colony algorithm to SRCPSP. In recent years, due to its fast response and good solving ability, PRs-based heuristics have been considered and applied. Chen et al. (2018) summarised the existing 12 PRs and designed five PRs based on statistics. Their experiments analysed the pros and cons of different PRs, and proved that the optimal PRs have the stronger solving ability under the fast response by comparing with different meta heuristics, especially in the middle and high distribution. Wang et al. (2017) analysed the performance and characteristics of 20 PRs for SRCPSP, and based on this study, Chen et al. (2019) considered the dynamic factor of new project arrival, and further analysed the advantages and disadvantages of the same 20 PRs.

2.2.2 Scheduling policies

Another key point is how to convert the activity sequence into scheduling with scheduling policies. Scheduling policies can be classified as static policies and dynamic policies (Chen et al., 2018). In the static policy class, applied policies will not be changed during the whole execution process and their relevant information is known before decision time, while policies in the dynamic class will be applied dynamically with seeking a best policy to apply step by step in the process, thus the solution with dynamic policies may be just a temporary solution, which needs to be updated or repaired in an dynamic way with the activity completion or other new information acquisition. Although the dynamic policies are more flexible and more adaptable, applying them takes a lot of computing time. This gives the reason why the use of static policies is more widely adopted (Chen et al., 2018; Li & Womer, 2015).

Based on the current research (Ashtiani, Leus, & Aryanezhad, 2011; Chen et al., 2018; Rostami, Creemers, & Leus, 2018), the static policies are divided into the following six sub-classes: resource-based policy class (RB-policies), activity-based policy class (AB-policies), earliest-start policy class (ES-policies), pre-selective policy class (PS-policies), pre-processor policy class (PP-policies) and generalised preprocessor policy class (GP-policies). The characteristics of these polices can be summarised into three categories.

(1) Direct policy category

RB-policies and AB-policies fall into this category because they can be directly applied without invoking additional decision-making strategies. The function of RB-policies (Ashtiani et al.,2011) is similar to that of parallel SGS in RCPSP, that is to say, for an ordering L of all activities, a decision maker should start as many activities as possible without violating resource constraints and precedence constraints according to the order in L at each decision time. RB-policies are neither monotone nor continuous (Radermacher, 1981), so they are associated with Graham anomalies (Graham, 1969). Graham anomaly is an abnormal phenomenon that the decrease of an activity duration will lead to the increase of the makespan. On the basis of RB-policies, AB-policies (Ballestín, 2007) add a start-start (SS) side constraint, that is, any activity in ordering L cannot start before its predecessors start, so AB-policies are often continuous and monotonous (Chen et al., 2018).

(2) Policy category with minimal forbidden set

By introducing a minimum forbidden set, Radermacher (1981) and Igelmund and Radermacher (1983) proposed ES-policies and PS-policies, respectively. A forbidden set is a subset of all activities in which the sum of resource requirements (one or more resources) between any two activities is greater than the resource availability, so they cannot be executed at the same time. If any subset of the forbidden set breaks this condition, it is called the minimum forbidden set (Stork, 2001). If a finish-start (FS) constraint is added between activities in the minimum forbidden set, that is, one activity must start after another activity is completed, then the original logical relationship between activities will be extended to calculate the earliest start time without resource limitation. ES-policies and PS-policies are two different expressions based on this idea. An ES-policy is a set of FS constraints while a PS-policy is an activity sequence.

(3) Policy category with extra constraint

Consistent with ES-policies and PS-policies, PP-polices proposed by Ashtiani et al. (2011) firstly have the logical relationship of original activities by adding extra FS constraints. The extra FS constraints are added through the extra meta-heuristic. Inspired by this, Rostami, Creemers, and Leus (2018) found that the extension of activity logical relationship would rely not only on extra FS constraints, but also on SS constraints. Thus, GP-policies are proposed and a meta-heuristic algorithm is designed to find these constraints. Their experiments show that the extra FS and SS are effective for optimising the expected makespan.

2.3 Application of hyper-heuristics in project scheduling

Although many approaches have been proved to be effective, both PRs and meta-heuristics have some defects. For meta-heuristics, their optimisation needs to be iterative, resulting in a lot of computing time, which limits their applicability and practicability as the problem scale increases. At the same time, meta-heuristics always need a lot of random operations, such as crossover and mutation in genetic algorithm, which reduce their stability, especially in the case of a problem with randomness, thus weakening their solving ability. On the contrary, PRs can achieve rapid response and effective scheduling, but have no optimisation ability. From relevant research (Chen et al., 2018, 2019; Wang et al., 2017), we can see that the pros and cons of PRs are related to data characteristics and objective functions.

In order to overcome these defects, hyper-heuristic methods attract a growing number of attentions and are gradually used in project scheduling in recent years, since their optimisation process involves selection, combination and generation of heuristics. At the beginning, only greedy search (Anagnostopoulos & Koulinas, 2012) or threshold accepting (Koulinas & Anagnostopoulos, 2012) was used to realise heuristic control. Then, meta-heuristics such as evolutionary algorithm and swarm intelligence are incorporated into hyper-heuristic optimisation. In addition to a hyper-heuristic framework based on particle swarm optimisation proposed by Koulinas, Kotsikas, and Anagnostopoulos (2014), genetic programming becomes the main way for hyper-heuristic project scheduling. Lin, Zhu, and Gao (2020) constructed a genetic programming framework, which incorporated 10 low-level heuristics, for solving RCPSP by considering multi-skills and employing the design-of-experiment method to investigate the effect of relevant parameters. Chand et al. (2018) proposed a genetic programming algorithm to solve RCPSP by evolving and generating new PRs and verified its superiority by comparing with traditional PRs.

On this basis, Chand, Singh, & Ray. (2019) further considered dynamic resource disruptions and validated the effectiveness of genetic programming with evolutionary PRs.

2.4 Motivation and contribution

Based on the reviewing of the above existing research, our research motivations and contributions are summarised as follows:

1. As a new way of project scheduling, hyper-heuristic has been paid more and more attention in recent years, but the research is still at its infant stage. To the best of our knowledge, there is no research to explore the effect of hyper-heuristic scheduling on SRCPSP. Generally speaking, single hyper-heuristic scheduling may have the risk of insufficient stability under dynamic scheduling problems (Hart & Sim, 2016). Therefore, our motivation is to explore the idea of ensemble learning-based scheduling for better than the single hyper-heuristic scheduling. It is expected to achieve better results through multiple PRs collaborative scheduling, which is a common idea in classification and regression (Polikar, 2006). Thus, in this paper, an HH-EGP framework is proposed to apply hyper-heuristic and ensemble learning for solving SRCPSP for the first time.

2. In order to implement HH-EGP, some basic components have been first modified (see Section 3) to allow combining the critical path method with RB-policies to obtain the excellent schedule. This is because that the evaluation of decision ensemble or PRs depends on policy classes as evidenced in (Chen et al., 2018) and (Villafáñez et al., 2019).Secondly, according to the characteristics of SRCPSP and to guarantee a complete solution space, we design a coding structure with discriminant and modify the function set and terminal/attribute set to form a new PR structural representation.

3. Due to the aforementioned changes, the existing voting or other cooperative mechanisms in scheduling field are not suitable for SRCPSP. Therefore, we design a new sequence voting mechanism, through the cooperation between PRs to generate a final priority sequence at each decision time for achieving RB-policies collaborative scheduling. Meanwhile, we design and integrate three local searches, discriminant mutation and discriminant population renewal to improve the evolutionary process of each subpopulation (see Section 4).

Based on the benchmark PSPLIB (Sprecher & Kolisch, 1996), experiments are carried out under five distributions to verify the superiority of the proposed method and further analyse the influence of the number of subpopulation individuals and decision PRs on its performance.

3 The framework of HH-EGP

The framework of HH-EGP is shown in Fig.2, which includes its evolution process and contribution. As can be seen from the left of Fig.2, HH-EGP is to generate a better decision ensemble through continuous iteration on the basis of initialisation. Some improvements are needed in the key steps to achieve better evolution and collaborative decision-making, as shown on the right of Fig.2. These improvements include two aspects: basic components displayed in Yellow and the ensemble evolution in Green. In the basic components, two parts need to be improved as the basis of initialisation and scheduling generation in HH-EGP. The first is the new representation of each PR to complete individual initialisation, including the encoding structure

with discriminant, the modified function and the terminal/attribute set needed to form this structural expression. The second is the selection of policy classes needed for PRs or ensemble scheduling to achieve the evaluation. In the ensemble evolution, three local searches, the discriminant mutation and the discriminant renewal are partnered with crossover to improve the subpopulation evolutionary process, resulting in better representative PRs to update the decision ensemble. Meanwhile, a sequence voting mechanism is designed to evaluate the decision ensemble with collaborative decision as the basis for the ensemble evolution.

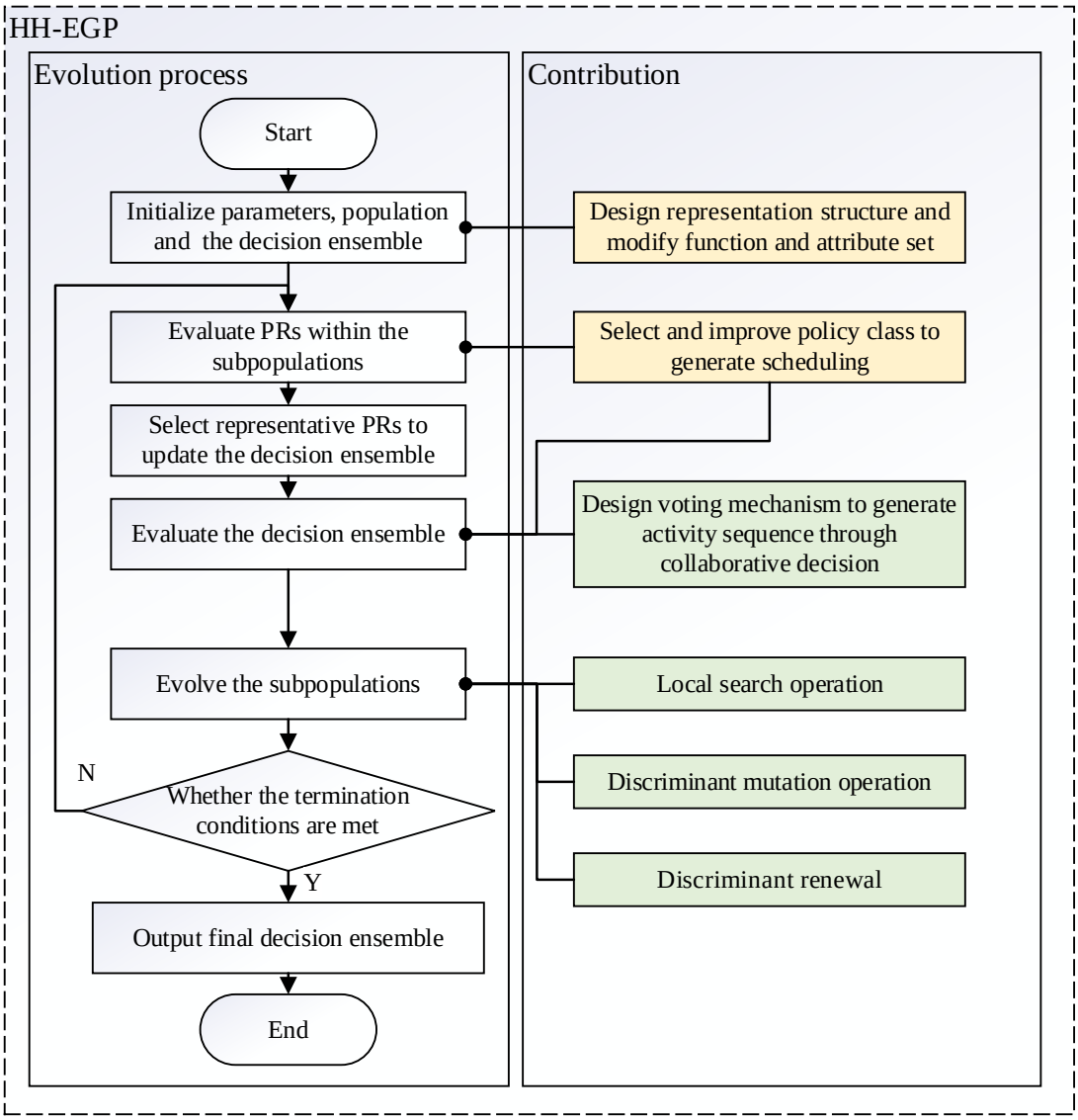


Fig.2 The framework of HH-EGP

3.1 New structural representation of PR

3.1.1 The encoding structure of PR

In the existing hyper-heuristic research of evolution PRs, the PR structure usually uses an arithmetic representation, which is a tree structure shown in Fig.3. The nodes in the bottom layer

are continuously transported through the functional symbols in the upper layer to form a priority expression, and the result is (*LS-EF*)×*DT*. However, according to the performance analysis of PRs by Kolisch (1996b) and Browning and Yassine (2010) in project scheduling, the outstanding PRs may be to maximise or minimise some priority expression, such as the rule of the most total successors and the early finish time rule. In addition the existing structure (Chand et al., 2018; Chand, Singh, & Ray, 2019) only contains priority expressions. Thus in this paper, the discriminant is added to the PR structure as shown in Fig.3. When the discriminant at the top level is "fall", the priority expression is minimised, and the maximisation is "rise". Therefore, there are more combinations, because the same priority expression can form two PRs through different discrimination. The layer number corresponding to each node in the tree is called its depth and the depth of the discriminant in the top layer is 0. Thus, the maximum depth of Fig.3 is 3.

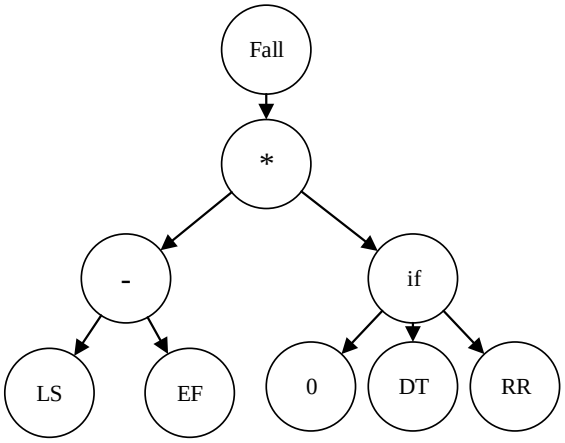


Fig.3 The example of PR structure

3.1.2 Modifying the terminal/attribute set

Chand, Singh, and Ray (2019) summarised some attributes in the deterministic duration in RCPSP, but due to the characteristics of SRCPSP, we need to modify the existing attribute set in two aspects. First, the RB-policies used in this paper will build a temporary scheduling and filter out an eligible set at every decision time (see Section 4.2). The early start time for each activity in the collection is consistent, so the early start in the static attribute is not needed. At the same time, when the activity duration is uncertain, referring to the PRs summarised by Chen et al. (2018), we take the duration as a priority attribute, then the original attribute set adds two attributes, *Duration* and *Total Successor Duration.* Secondly, the calculation of all-time related attributes in the training process depends on the expected duration d^*, and the normalisation only calculates the activities in the eligible set. The modified attribute set (newly added in Italic) and its calculation formula are shown in Table 1.

Table 1 The attribute set of activity

Attribute	Calculation formula
Early Finish (EF_i)	$\frac{EF_i}{\max EF_j} \quad i, j \in AE_t$
Late Start (LS_i)	$\frac{LS_i}{\max LS_j} \quad i, j \in AE_t$

Late Finish (LF_i)	$\frac{LF_i}{\max LF_j} \quad i,j \in AE_t$
Total Successor (TS_i)	$\frac{\lvert S_i \rvert}{\lvert V \rvert - 1} \quad i \in AE_t$
Total Successor Duration (TSD_i)	$\frac{1}{\sum_{j \in V} d_j^*} \sum_{j \in S_i} d_j^* \quad i \in AE_t$
Duration (DT_i)	$\frac{d_i^*}{\max d_j^*} \quad i,j \in AE_t$
Resources Required (RR_i)	$\frac{1}{\lvert K \rvert} \sum_{k=1}^{\lvert K \rvert} \begin{cases} 1 & if\ r_{ik} > 0 \\ 0 & otherwise \end{cases} \quad i \in AE_t$
Average Resource Requirement ($AvgRR_i$)	$\frac{1}{\lvert K \rvert} \sum_{k=1}^{\lvert K \rvert} \frac{r_{ik}}{R_k} \quad i \in AE_t$
Maximum Resource Requirement ($MaxRR_i$)	$\max \frac{r_{ik}}{R_k} \quad k \in \{1,2,...,\lvert K \rvert\}, i \in AE_t$
Minimum Resource Requirement ($MinRR_i$)	$\min \frac{r_{ik}}{R_k} \quad k \in \{1,2,...,\lvert K \rvert\}, i \in AE_t$

AE_t is the eligible set at time t

The attributes in modified attribute set are described as follows:

- EF_i: The earliest finish time of activity i in the eligible set, of which calculation ignores resource constraints.
- LS_i/LF_i: The latest start/finish time of activity i in the eligible set, and the resource constraints are ignored when calculating.
- TS_i: The total number of all immediate and non-immediate successors of activity i.
- TSD_i: The duration sum of all immediate and non-immediate successors of activity i.
- RR_i: The total number of resource types required for activity i.
- $AvgRR_i$: The average resource requirement of activity i across its duration.
- $MaxRR_i$: The maximum resource requirement of activity i across its duration.
- $MinRR_i$: The minimum resource requirement of activity i across its duration.

3.1.3 The function set

On the basis of Chand et al. (2019), the function set extends three kinds of function operators, whose calculation method is shown in Table 2.

Table 2 The function set

Symbol	Function	Formula	Symbol	Function	Formula
+	Add(a,b)	$a+b$	-	Sub(a,b)	$a-b$
*	Mul(a,b)	$a \times b$	Neg	Neg(a)	$-1 \times a$
Exp	Exp(a)	e^a	Abs	Abs(a)	$\begin{cases} a & if\ a \geq 0 \\ -1 \times a & otherwise \end{cases}$
÷	Div(a,b)	$\begin{cases} a/b & if\ b \neq 0 \\ 0 & otherwise \end{cases}$	Max	Max(a,b)	$\begin{cases} a & if\ a \geq b \\ b & otherwise \end{cases}$

Min	Min(a,b)	$\begin{cases} a & if\ a \le b \\ b & otherwise \end{cases}$	If	If(c,a,b)	$\begin{cases} a & if\ c = 0 \\ b & otherwise \end{cases}$

3.2 RB-policies integrated with critical path method

Within the six policy categories (see the Section 2), RB-policies are chosen in the proposed HH-EGP for the following considering. According to the analysis (Stork, 2001), taking PSPLIB data as an example, the average number of the minimal forbidden set in different instances is 326 with 30 activities, but the average value reaches 243871 when the activity number increases to 120. Thus, if we adopt the policy class with minimal forbidden set, HH-EGP will increase a lot of additional calculation. And for the policy class with extra constraint, from the research of Ashtiani et al. (2011) and Rostami et al. (2018), there are no any effective direct methods, but turn to meta-heuristics to add extra FS or SS constraints between activities. However, the search process for adding extra constraints is contrary to the original intention of PRs scheduling to avoid iteration. Therefore, the direct policy class is the best choice, in which RB-policies are better than AB-policies in both efficiency and ability to obtain excellent objective value (Chen et al., 2018), so the RB-policy class is our final choice.

In addition, we integrate the RB-polices with the critical path method as shown in Algorithm I. It is known that RB-policies will generate an activity priority sequence, but this sequence is known before the decision (sorted by one PR) and cannot change in the decision. However, when the RB-policy is combined with the critical path method, this integrated approach can recalculate the start time and finish time of each activity through the critical path method to generate a temporary schedule at decision time, so that the generated sequence and activity priority will change dynamically. In this way, the function of RB-policies is similar to that of parallel SGS in RCPSP. Referring to Villafáñez et al. (2019), if parallel SGS generates a temporary schedule according to the critical path method at every decision-making time, it will achieve better scheduling because it can dynamically change the priority of activities and the sequence of activities. Similarly, in SRCPSP, the dynamic change of activity priority is more important, because the initial activity sequence only depends on the estimated value of one attribute. Therefore, this paper designs an improved RB-policy integrated with the critical path method.

Algorithm I The RB-policy with the critical path method

Initialise decision time t = 0 and get all activities V in the project;
while $|V|>0$
    // Step 1: Temporary schedule construction
    Generate a temporary schedule according to the critical path method;
    // Step 2: Eligible set filtering
    Initialise the eligible set AE_t;
    for i **in** V
        Get the predecessor set PS_i of activity i;
        if all activities in PS_i are completed
            Put the activity i in AE_t;
        end if

end for

// Step 3: Priority assignment and sorting of activities

Calculate the priority of each activity in AE_t and maximise or minimise sorting to generate AE_{t1};

// Step 4: Activity scheduling

for i **in** AE_{t1}

if the remaining resources can meet the requirement of activity i

schedule activity i and remove i from V;

update remaining resources;

end if

end for

set t =the minimum completion time of the started activities;

end while

Note that because RB-policies are based on greedy use of resources, it will bring Graham anomalies. The reason for Graham anomaly is that the project itself has changed, such as reducing the duration of one activity. In this paper, the RB-policies combined with the critical path only makes the priority and activity sequencing change dynamically without changing the project characteristics. Therefore, this integrated approach only dynamically calculates the priority without changing the project structure. Therefore, it does not affect the characteristics of RB-policies, that is, it can neither eliminate Graham anomalies nor bring about additional problems.

4 The evolution of HH-EGP

Unlike the single hyper-heuristic, HH-EGP is based on a decision ensemble rather than a single PR. A decision ensemble contains multiple PRs, each of which is evolved by a subpopulation. Each PR in the decision ensemble is equal, that is, HH-EGP does not assign more weight to a particular PR decision or prefer any particular PR. Therefore, the ensemble scheduling is to measure the activity priority sequence determined by each PR to generate a comprehensive sequence at each decision time, i.e., HH-EGP generates scheduling results through multiple single hyper-heuristics cooperative decisions.

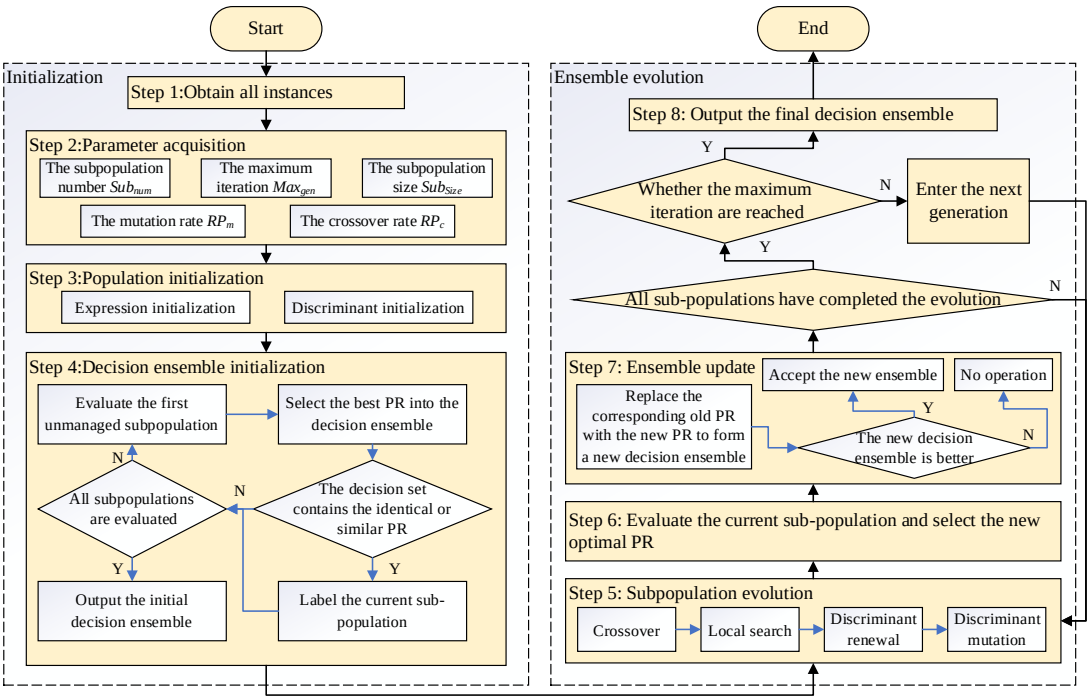


Fig.4 The evolutionary process of HH-EGP

The goal of HH-EGP is to generate a decision ensemble through evolution (initialisation and iteration), and the evolutionary process of HH-EGP is shown in Fig.4, in which the highlighted part represents its main execution steps. As can be seen from Fig.4, the evolutionary process of HH-EGP mainly includes two parts, initialisation and ensemble evolution. The main work of initialisation is to prepare all the inputs needed in the iteration, including obtaining instances to evaluate PRs and the decision ensemble, setting parameters, initialising population based on gene structure, and initialising the decision ensemble by selecting representative PR of each subpopulation. There are four steps in the ensemble evolution. Firstly, each subpopulation evolves itself to form a new PRs through genetic operators. Secondly, the optimal new PR is selected to update the decision ensemble. Thirdly, the scheduling ability of the new ensemble is evaluated to decide whether to accept the new ensemble. And finally, when the termination condition is reached, the final decision ensemble will be output. It is worth mentioning that the data under all conditions are difficult to be obtained in practice. Therefore, the evolution of HH-EGP can only be carried out under the available data, i.e., the training set, while in the execution of HH-EGP (Fig.4), the involved instances are the training set. From Fig.4, the two key problems of HH-EGP are how to iterate the subpopulation and how to evaluate the decision ensemble, that is, how to make collaborative decision.

4.1 Subpopulation iteration

In HH-EGP, the PRs contained in the decision ensemble are derived from multiple subpopulations for recommendation. Therefore, the subpopulation iteration is the basis of forming a better decision ensemble, including initialising subpopulation, generating new PRs, evaluating new PRs and updating subpopulation.

4.1.1 Population initialisation and PR evaluation

Before the iteration in Fig.4, each subpopulation needs to be initialised, i.e., PRs with different structure as shown in Fig.3 are generated. Due to the structure characteristic, the initialisation of PRs is divided into two parts. The priority expression in the lower layer is generated by the classical ramped half-and-half method, and its maximum depth is controlled between 2 and 6 (Luke & Panait, 2001), while the discriminant is controlled by a random number, that is, if the generated random number is less than 0.5, it is "fall", otherwise it is "rise". In addition, PR evaluation depends on Eq.(1), where the expected makespan $E(\boldsymbol{d})$ of a single instance is the scheduling result according to Algorithm I when all activities take the expected duration d^*. In this way, during the training process, PRs only need to perform scheduling once in a single instance, thus greatly reducing the computation amount and time.

4.1.2 Population evolution

Population evolution is a key part of subpopulation iteration in HH-EGP because it can generate different new PR structure by splitting and combining, and its execution process is shown in Algorithm II.

Algorithm II The evolution of subpopulation

// Step 1: Obtain subpopulation and parameters
Obtain the subpopulation, the decision ensemble, the mutation rate RP_m and the crossover rate RP_c;
Copy old subpopulation to generate a temporary subpopulation
// Step 2: Crossover operation
Shuffle randomly to produce a non-repeating sequence;
for *num*:Sub_{size}/2
  if random< RP_c
    Obtain two individuals from temporary subpopulation according to sequence order and perform crossover;
    Insert the new individual into the corresponding position;
  end if
end for
// Step 3: Local search operation
for *num*: Sub_{size}
  Get the *num*th individual in the temporary subpopulation;
  Generate a random integer $rand_{int}$ from 0 to 2;
  switch($rand_{int}$**)**
    case 0: subtree replacement local search;
    break;
    case 1: node replacement local search;
    break;
    case 2: subtree deletion local search;
    break;

end switch

end for

// Step 4: Discriminant mutation operation

for *num*: Sub_{size}

if random< RP_m

Get the *num*th individual in the temporary subpopulation;

Execute discriminant mutation;

end if

end for

// Step 5: Discriminant subpopulation renewal

if the old subpopulation is labelled

Delete all identical or similar individuals in the old subpopulation with the decision ensemble;

end if

Merge the temporary subpopulation into old subpopulation;

Select Sub_{size} optimal individuals from old subpopulation to form next subpopulation;

1 Crossover

In genetic programming, the common way of crossover is subtree crossing (Zhou, Yang. & Zheng, 2019), which is that two parent individuals exchange their part subtree to form new individuals, as shown in (a) of Fig.5.

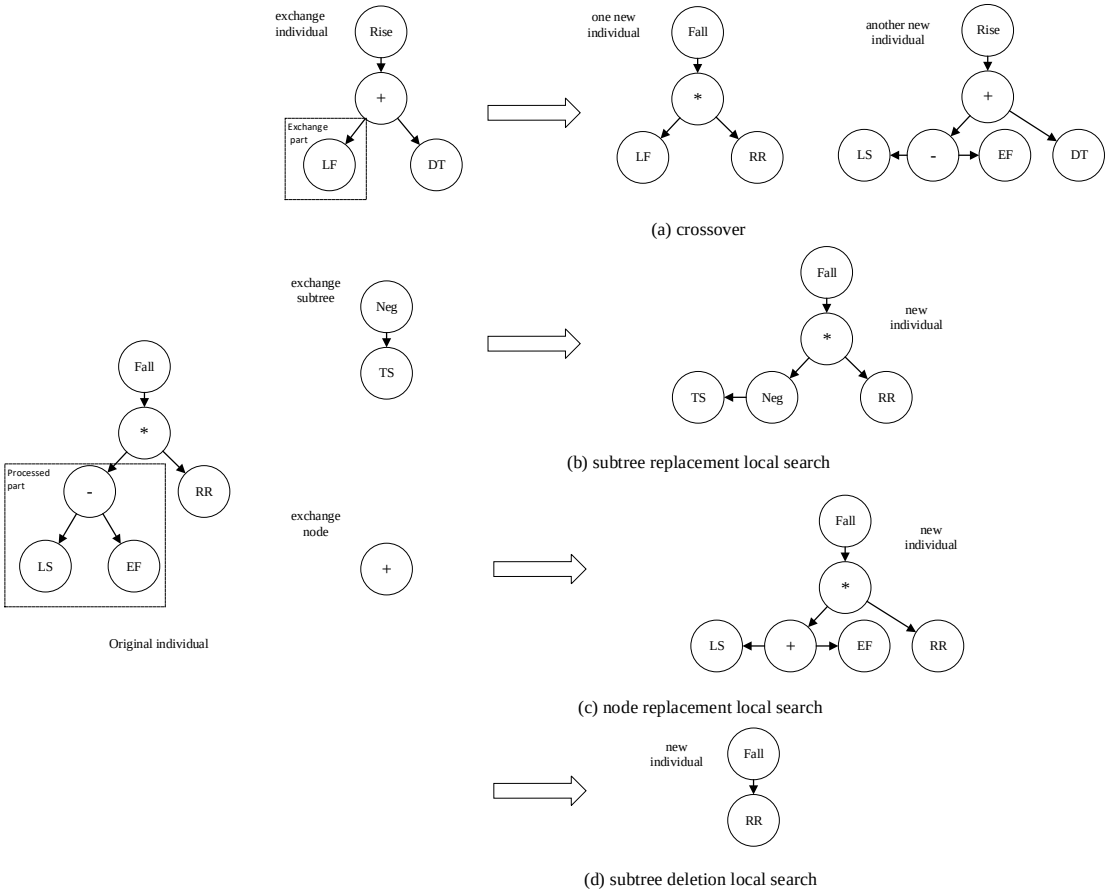


Fig.5 Crossover and local searches

2 Local searches

Different from the existing research (Chand et al., 2018; Chand, Singh, & Ray, 2019; Lin, Zhu, & Gao, 2020), this paper designs three local searches to replace the mutation operation in genetic programming, so as to obtain more combination ways to provide more choices for the decision ensemble composition. The examples of three local searches are shown in (b), (c) and (d) of Fig.5.

Subtree replacement local search: this local search comes from the mutation of genetic programming, that is, replacing a part of the original tree with a new randomly generated subtree, which may increase the new characteristics that do not exist in the original population.

Node replacement local search: different from others, this local search will not change the tree structure, but only replace one of the nodes with the same property. For example, if the selected node is the function symbol “+”, it can only be replaced by other function symbols with two child nodes, such as "-", when performing the local search. In addition, if the selected node is "if", its first child node will be replaced, for example, converting from "0" to "1" to change the selection of "if".

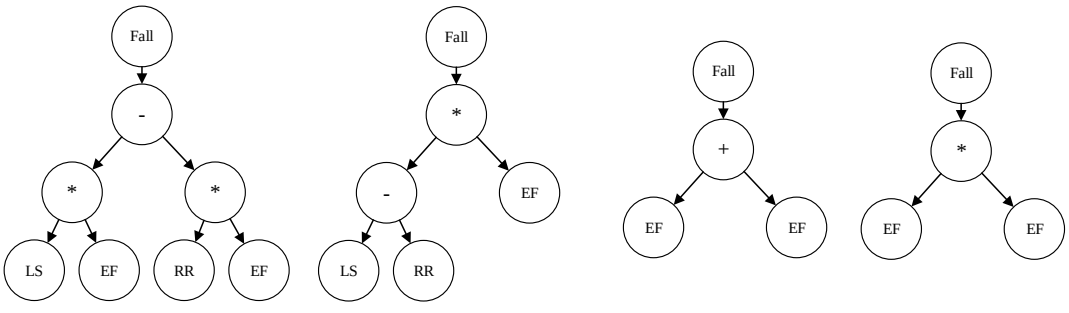
Subtree deletion local search: it is not necessary that all subtrees of an overly complex structure tree are effective, so this paper designs the local search to randomly delete some subtrees. There are two special cases in performing this local search: 1) if the parent node of the selected node for deletion is "if", then its judgment node "0" or "1" should also be deleted; 2) If its parent node has only one child node, we will randomly select an attribute from the terminal set to replace the subtree with its root node.

3 Discriminant mutation

Although three local searches are used to replace the original genetic programming mutation to realise further changes of structure tree, due to the coding particularity, a discriminant mutation is designed to transform the top discriminant. As shown in Algorithm II, when the generated random number is less than the mutation rate, it will become "rise" if the top-level discriminant of the original individual is "fall".

4 Discriminant population renewal

This discriminant population renewal is to eliminate the identical or similar individuals in the decision ensemble, which refers to the same or similar priority expression generated by the inconsistent structure tree. For example, in Fig.6 (a), the priority expressions generated by two different structure trees are the same $(LS\text{-}RR)\times EF$, while in (b), although they are $2EF$ and EF^2, respectively, the earliest start time of each activity cannot be negative, so their effects are the same. All the labelled subpopulations in Fig.4 will perform this operation to destroy the existing optimal individuals to re-evolve, resulting in avoiding multiple identical or similar individuals in the decision ensemble to affect collaborative decision-making.



(a) identical individuals (b) similar individuals

Fig.6 The same or similar individuals

4.2 Ensemble evaluation

Similar to the individual evaluation, the ensemble evaluation is also to calculate the objective function Eq.(1) after scheduling in multiple instances, so the key is how to realise the collaborative decision between PRs at every decision time when scheduling. In job shop scheduling, an ensemble decision is to continuously select the job with the highest priority by voting or other ways (Hart & Sim, 2016; Park et al., 2018; Part et al., 2015). However, the principle that only one job can be selected at each decision time is contrary to RB-policies, whose principle is shown in Algorithm I that multiple activities can be scheduled at decision time after ranking the eligible set as long as the relevant constraints are not violated. Therefore, a sequential voting mechanism is designed in this paper shown in Algorithm III, which can generate the final sequence at each decision time by voting according to the multiple sequences of PRs, so as to realise collaborative decision under SRCPSP.

Algorithm III The sequence voting mechanism

// Step 1: Obtain input and initialisation
Obtain the current eligible set AE_t, the decision ensemble DE;
Initialise the final sequence S_{final};
// Step 2: PR independent prioritisation
for PR **in** DE
    Calculate activity priority based on PR expression and sort AE_t;
end for
// Step 3: Collaborative decision sequencing
while the number of activities in S_{final} is less than the size of AE_t;
    Initialise a temporary set TS;
    for PR in DE
        Select the first activity of PR corresponding sequence to add to TS;
    end for
    Count the occurrence number of each activity in TS and select the activity with the maximum number;
    if the maximum number of multiple activities is equal
        Select the activity with the lowest code;
    end if
    Add the selected activities to S_{final};
    for PR in DE
        Delete the selected activity from PR corresponding sequence;
    end for
end while

In order to further understand Algorithm III, take the collaborative scheduling of 4 activities with 3 PRs as an example, and the process of generating the final sequence is shown in Fig.7. First of all, each PR in the decision ensemble will calculate the priority of each activity in the eligible

set according to its own representation and generate a sequence by sorting the eligible set shown in (a) of Fig.7. Then, each PR will recommend the activity with the highest priority to the decision ensemble, and the decision ensemble will then select the one with the most recommended number, as the A1 in Fig.7 (a). If there is a tie in this process, for example, supposed that the activity with the highest priority of PR1 in Fig.7 (a) is A4 rather than A1, then the recommended number of A1, A3 and A4 are equal. The decision ensemble will select the smallest code activity (A1) rather than random selection to avoid instability of decision ensemble in each scheduling. Next, when the optimal activity A1 is obtained, it will be added to the final sequence and removed from each PR corresponding sequence because the activity has been decided, and the result is shown in Fig.7 (b). In Fig.7 (b), the A3 recommended by PR2 in the first round is not added to the final sequence, so PR2 will continue to recommend activity A3. PR1 and PR3 will recommend the highest priority activity in the sequence after removing A1, i.e., A4 and A3, respectively. By analogy, the decision set will be selected again in a new set of recommended activities in such a cycle until all activities are selected to generate the final sequence as shown in (d) of Fig.7.

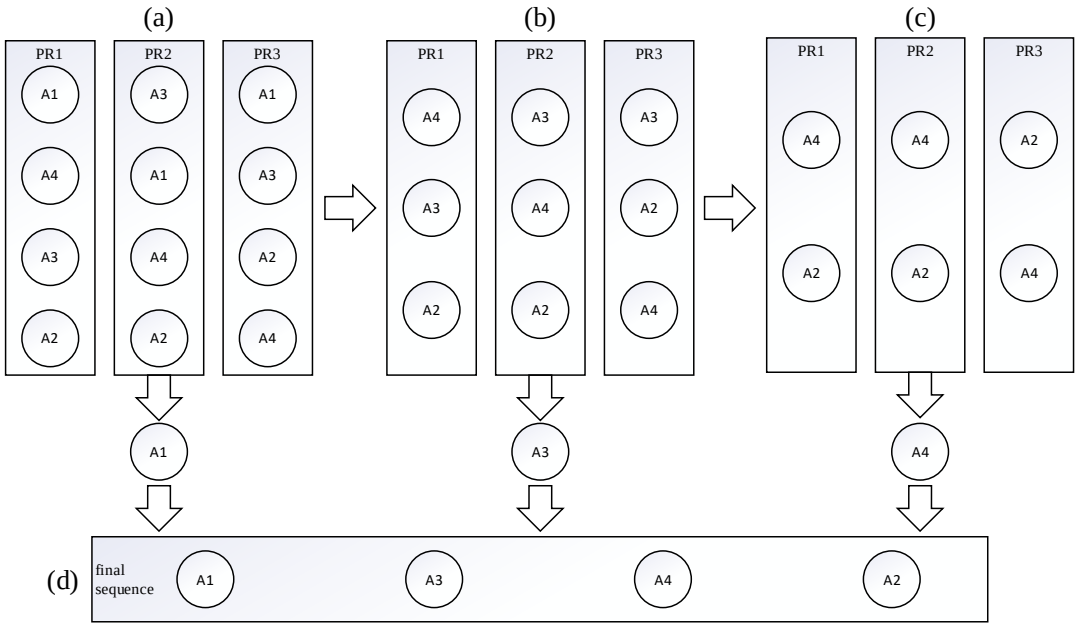


Fig.7 The sequence voting mechanism

5 Computational experiments

5.1 Experimental setup

All experiments are performed on an Intel Core i5-4200 quadcore processor computer with 2.50 gigahertz clock speed and 8 gigabyte RAM, and all HH-EGP related programs are coded in Java using MyEclipse 2017 CI. The parameters of HH-EGP mainly refer to Chand et al. (2019), and the related values are shown in Table 3, where *value* is the parameter value of HH-EGP, and *valueR* is the parameter value in Chand et al. (2019). Since it is not an ensemble scheduling problem in Chand et al. (2019), there are no values of Sub_{num} and Sub_{Size}.

Table 3 The parameters of HH-EGP

Name	Symbol	*value*	*valueR*
subpopulation number	Sub_{num}	6	N/A
subpopulation size	Sub_{Size}	40	
mutation rate	RP_m	0.2	0.1
crossover rate	RP_c	0.9	0.9
maximum number of iterations	Max_{gen}	25	25

By referring to Ballestín (2007), Ballestin and Leus (2009), Ashtiani et al. (2011), Fang et al. (2015), Chen et al. (2018) and Rostami et al. (2018), the main verified data in this paper are the J120 set in PSPLIB, which contains the largest activity number in PSPLIB and includes 600 different instances (obtain at http://www.om-db.wi.tum.de/psplib/ library.html). In addition, the J30, the J60 and the J90 of PSPLIB, all of which include 480 instances, are also used to verify the generalisation performance of HH-EGP compared with traditional PRs. The five different distributions are adopted as shown in Table 4, including two low variance distributions, two medium variance distributions and one high variance distribution. Under each distribution, the expected activity duration is equal to the deterministic activity duration in PSPLIB, and the generation of random number with different distributions depends on math3.jar in Java.

Table 4 five distributions of activity duration

Distribution type	Code	Range	Variance
Uniform distribution	U1	$U(d'_{pa}-\sqrt{d'_{pa}}, d'_{pa}+\sqrt{d'_{pa}})$	$d'_{pa}/3$
	U2	$U(0, 2d'_{pa})$	$(d'_{pa})^2/3$
Beta distribution	B1	$B(d'_{pa}/2, 2d'_{pa}, d'_{pa}/2-1/3, d'_{pa}-2/3)$	$d'_{pa}/3$
	B2	$B(d'_{pa}/2, 2d'_{pa}, 1/6, 1/3)$	$(d'_{pa})^2/3$
Exponential distribution	E	$E(d'_{pa})$	$(d'_{pa})^2$

As described in Section 4, a training set should be selected for participating in the evolution of HH-EGP, while PSPLIB divides every 10 instances into a group according to similar design parameters (Sprecher & Kolisch, 1996), so we choose the first 50% of each group in the J120 to form a training set including 300 instances to meet more different conditions. At the same time, the test set used to obtain the experimental results in Section 5.2 to Section 5.4 consists of four parts, which are composed of all instances in the J30, the J60, the J90 and the J120, respectively. In each part, the performance evaluation of different methods depends on Eq.(1), in which $E(\boldsymbol{d})$ is replaced by means of 1000 simulations to achieve sufficient accuracy (Chen et al., 2018).

5.2 Comparison with heuristics and meta-heuristics

5.2.1 Objective value comparison under the J120

Because there are no hyper-heuristic methods to solve SRCPSP, in this part, we chose the existing heuristic and meta-heuristic methods for comparison. In heuristics, 17 PRs are derived from Chen et al. (2018), including 12 direct PRs and 5 simulation-based PRs, and their results tested under the J120 are shown in Table 5. In addition, HH-EGP is executed 10 times to train 10 decision ensembles, and the scheduling objective values tested under the J120 are shown in Table 6, where *Num* represents the number of experiments. The objective values of the optimal direct PR (LFT) and simulation-based PR (SLFT) and the average objective values of HH-EGP under different variance distributions are shown in Fig.8.

Table 5 The objective values of 17 PRs under the J120

PR	U1	U2	B1	B2	E
LFT	48.05	55.59	48.05	55.56	70.95
LST	48.38	55.93	48.38	55.93	71.40
OGRPW	48.93	56.25	48.95	56.16	71.30
MTS	49.83	56.93	49.85	56.85	71.74
WCS	50.94	58.75	50.94	58.71	73.96
ACS	51.04	58.89	51.02	58.82	74.03
IRSM	53.79	61.56	53.78	61.45	76.32
MIS	54.90	61.57	54.88	61.45	75.65
RSM	57.78	64.70	57.75	64.56	78.59
GRPW	59.68	66.07	59.66	65.91	79.58
SPT	61.63	68.10	61.63	67.94	81.51
GRD	62.11	68.61	62.09	68.53	82.35
SLFT	48.04	55.48	48.04	55.45	70.76
SLST	48.41	55.79	48.40	55.73	71.01
CI	54.29	62.82	54.33	62.82	78.66
SMSLK	55.96	62.65	55.91	62.54	76.61
MAV	63.84	70.23	63.85	70.12	83.61

Table 6 The objective values of HH-EGP under the J120

Num	U1	U2	B1	B2	E
1	44.27	50.86	44.82	52.87	64.56
2	44.22	50.80	44.76	52.8	64.52
3	46.03	52.38	46.59	54.59	66.04
4	45.03	51.66	45.59	53.83	65.45
5	44.21	50.78	44.75	52.78	64.38
6	44.54	51.12	45.08	53.19	64.81

7	44.09	50.75	44.63	52.73	64.52
8	44.48	51.16	45.04	53.19	65.06
9	45.24	51.93	45.81	54.16	66.12
10	44.19	50.82	44.71	52.75	64.51
avg	44.63	51.226	45.178	53.289	64.997

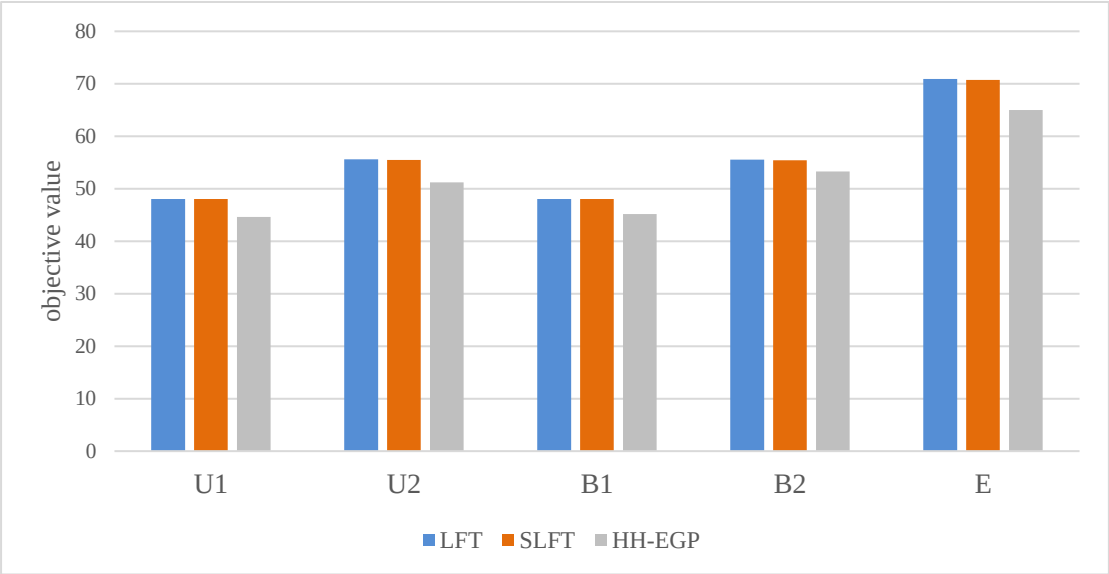


Fig.8 The objective values of PRs and HH-EGP under the J120

By comparing Table 5 and Table 6, regardless of the variance distribution, the objective values of HH-EGP are superior to those of all PR, including direct PRs and simulation-based PRs. Moreover, it can be seen from Fig.8 that the differences of objective values between optimal PRs and HH-EGP do not decrease with the distribution variance increasing, even approaches to 6% under high variance distribution. It shows that PR has no optimisation ability and *the PR evolution and collaborative decision-making are very effective in solving SRCPSP compared with a single PR, thus proving the superiority of this method.*

Meanwhile, five meta-heuristics were selected, which are the genetic algorithm with AB-policies (GA-AB) proposed by Ballestín (2007), the greedy adaptive search procedure based on AB-policies (GS-AB) described by Ballestin and Leus (2009), the two-phase genetic algorithm with PP-policies (GA-PP) implemented by Ashtiani et al. (2011), the estimation of distribution algorithm with RB-policies (ED-RB) developed by Fang et al. (2015) and the two-stage hybrid meta-heuristic based on GP-policies (HM-GP) proposed by Rostami et al. (2018). The relevant objective values under the J120 are shown in Table 7, where *Schedules* represents the scheduling number, while the values in Table 7 are the optimal objective values of 5000 or 25000 schedules performed in the above literature. The objective growth trend of HH-EGP and different meta-heuristics with the increase of distribution variance is shown in Fig.9.

Note that the other two policies introduced in Section 2.2.2 in terms of ES-policies and PS-policies, they are not listed in Table 7 for comparison with two main reasons. First, to the best of our knowledge, we have not found the results of the two policies in PSPLIB. Secondly, the two policies are based on the minimum forbidden set. As described in Section 3.2, the computational complexity of the minimal forbidden set increases nonlinearly with the increasing scale of the

problem. Therefore, we believe that these two policies have little potential in practical application.

Table 7 The objective values of five meta-heuristics under the J120

Method	*Schedules*	U1	U2	B1	B2	E
GA-AB	5000	51.49	78.65	/	/	120.22
	25000	49.63	75.38	/	/	116.83
GS-AB	5000	56.84	72.58	47.17	75.97	114.42
	25000	45.21	70.95	45.60	74.17	112.37
GA-PP	5000	48.86	58.91	49.01	58.82	76.03
	25000	47.21	58.07	47.25	57.95	74.56
ED-RB	5000	47.29	56.54	47.65	58.29	72.50
	25000	46.66	56.07	47.04	57.82	72.05
HM-GP	5000	46.71	55.95	46.87	55.95	71.71
	25000	44.98	55.37	45.12	55.42	71.29

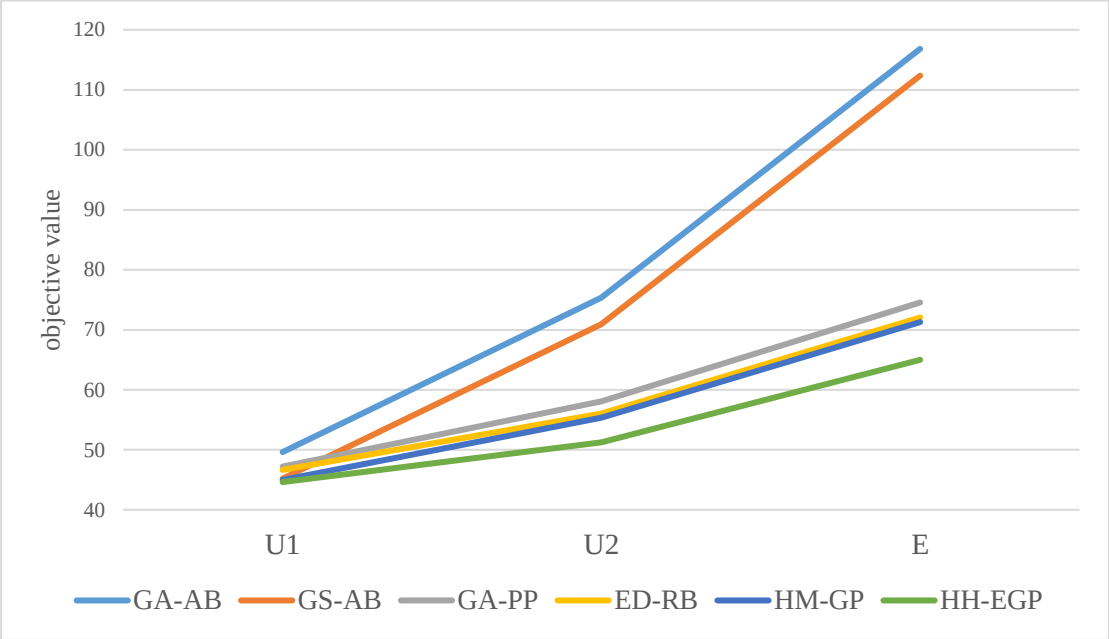


Fig.9 The objective growth trend of HH-EGP and meta-heuristics under different distribution

It can be seen from Table 6 and Table 7 that even under multiple schedules (such as 5000 or even 25000), the objective values of five meta-heuristics are still inferior to that of HH-EGP, which only needs to be scheduled once in one instance. At the same time, it can be seen from Fig.9 that although some meta-heuristics can approach the accuracy of HH-EGP under the low variance distribution, with the distribution variance increasing, the objective gap between them and HH-EGP is gradually widening due to the large number of random operations in meta-heuristics. It is worth mentioning that Table 7 shows the optimal rather than the average value under multiple scheduling, and the mate-heuristics need to search again in the face of new data, resulting in a lot of computation time waste. Thus, the superiority of HH-EGP is proved again, and *the potential of meta-heuristic is weaker than that of hyper-heuristic in dynamic problems*.

5.2.2 Generalisation performance verification

In addition, in order to further verify the generalisation ability of HH-EGP, the J30, J60 and J90 are used to test the training results in J120. The objective values of two optimal traditional PRs (Chen et al., 2018) and HH-EGP are shown in Table 8, where *Ins* represents the instances, *Dis* represents the distribution, and Num_z represents the zth training result of HH-EGP. Because Chen et al. (2018) did not carry out the traditional PR experiment in the J90, only the objective values of HH-EGP in J90 are given. Under the J30 and the J60, the objective values of LFT and SLFT and the average objective values of HH-EGP are shown in Fig.10 and Fig.11, respectively.

Table 8 The objective values of HH-EGP and traditional PRs under other instances

Ins	*Dis*	LFT	SLFT	Num_1	Num_2	Num_3	Num_4	Num_5	Num_6	Num_7	Num_8	Num_9	Num_{10}	Num_{avg}
	U1	21.60	21.60	20.61	20.66	21.16	20.87	20.57	20.67	20.69	20.96	20.99	20.65	20.783
	U2	30.89	30.83	29.53	29.64	29.98	20.83	29.47	29.58	29.68	30.03	29.86	29.60	28.82
J30	B1	21.59	21.60	20.97	21.04	21.52	21.22	20.94	21.06	21.05	21.35	21.34	21.00	21.149
	B2	30.87	30.76	31.07	31.11	31.58	31.25	30.98	31.05	31.19	31.58	31.46	31.06	31.233
	E	46.47	46.32	44.61	44.67	44.95	44.76	44.43	44.58	44.73	45.10	44.89	44.56	44.728
	U1	19.94	19.89	18.53	18.50	19.14	18.83	18.53	18.63	18.47	18.66	18.94	18.48	18.671
	U2	28.49	28.42	26.85	26.82	27.36	27.06	26.80	26.87	26.75	27.04	27.17	26.81	26.953
J60	B1	19.95	19.90	18.88	18.85	19.51	19.17	18.88	18.96	18.81	19.01	19.27	18.81	19.015
	B2	28.63	28.55	28.42	28.37	29.05	28.70	28.44	28.47	28.44	28.61	28.84	28.41	28.575
	E	44.97	44.94	42.66	42.66	43.24	42.90	42.56	42.73	42.74	42.92	43.12	42.68	42.821
	U1	/	/	17.04	17.00	17.69	17.27	17.05	17.11	16.98	17.17	17.35	16.98	17.164
	U2	/	/	24.81	24.75	25.38	24.97	24.78	24.86	24.73	24.95	25.13	24.73	24.909
J90	B1	/	/	17.34	17.31	18.02	17.57	17.34	17.42	17.29	17.49	17.66	17.30	17.474
	B2	/	/	26.36	26.33	26.99	26.61	26.38	26.42	26.31	26.50	26.80	26.31	26.501
	E	/	/	40.47	40.42	41.02	40.65	40.37	40.54	40.36	40.67	40.92	40.38	40.58

As can be seen from Table 8, Fig.10 and Fig.11, when the decision ensemble trained under the J120 is used to make decisions on the J30 and the J60, the objective values of other conditions are better than the optimal traditional PRs except that the effect under B2 distribution is similar to that of LFT and SLFT, which *proves that HH-EGP has good generalisation performance*. Despite the lack of comparison, we also believe that the objective values of HH-EGP under the J90 is not weaker than that of LFT and SLFT, because all subsets of PSPLIB depend on "ProGen" (Sprecher & Kolisch, 1996) with different parameters.

Moreover, by comparing Fig.8, Fig.10 and Fig.11, it can be found that with the increase of the activity number in the instances, the objective value gap between HH-EGP and traditional PRs gradually increases (which is better than the traditional PRs under B2 distribution in the J120). Thus, it can be proved that *HH-EGP is more effective in dealing with large-scale SRCPSP*.

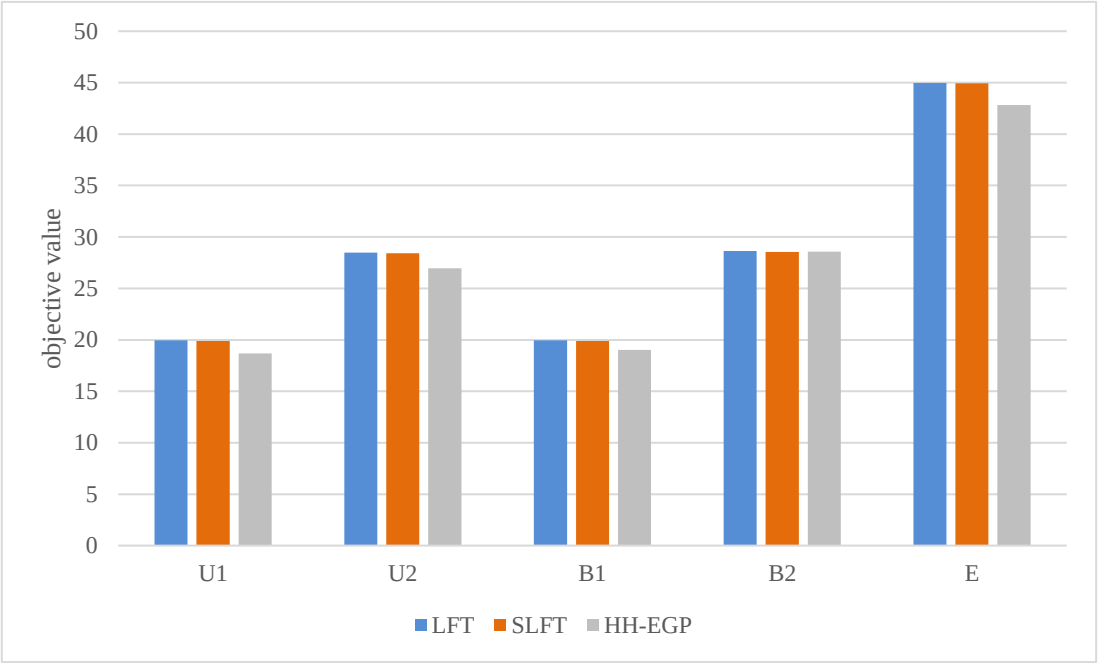


Fig.10 The objective values of PRs and HH-EGP under the J30

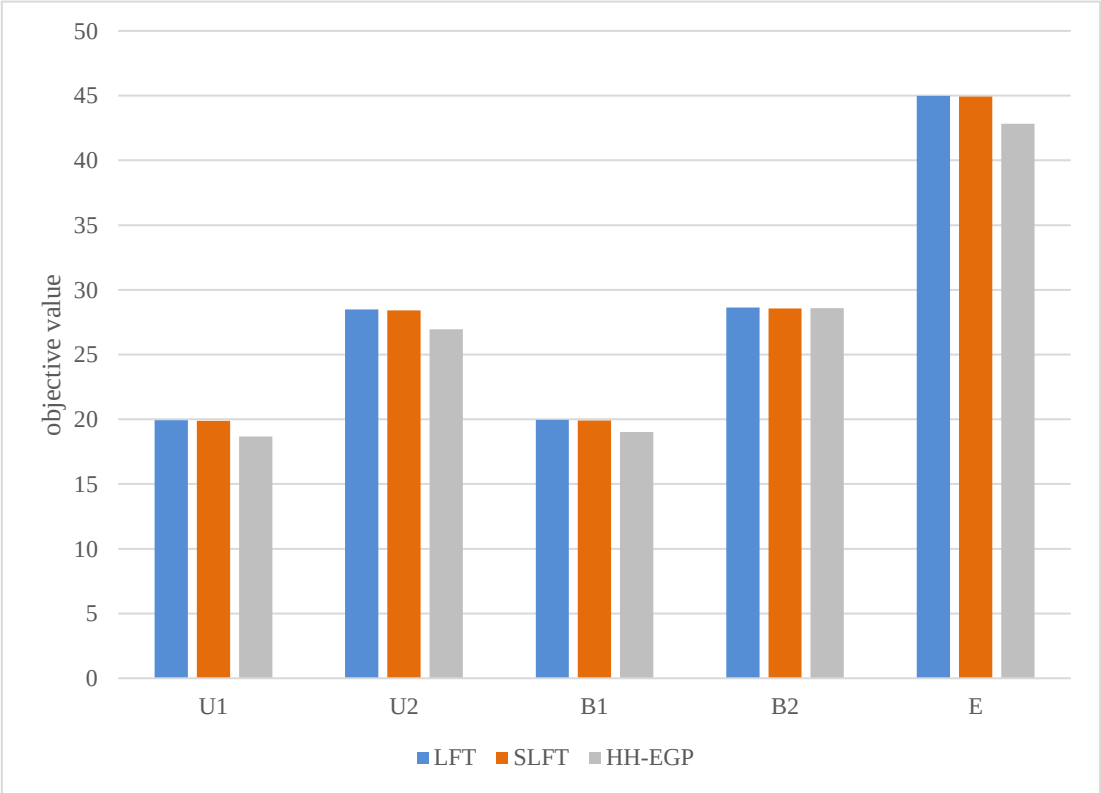


Fig.11 The objective values of PRs and HH-EGP under the J60

5.3 Comparison with Single hyper-heuristic

In order to further explore the effectiveness of collaborative decision-making, we use a single hyper-heuristic to compare with HH-EGP. Since there is no research on the use of hyper-heuristic to solve SRCPSP, we adopt Fig.4 and algorithm III without collaborative decision to implement

the single hyper-heuristic for evolving rules. From Section 5.1, it can be seen that HH-EGP is evolved from six subpopulations under the condition that the Sub_{Size} is equal to 40. In order to ensure the same search amount in each generation, the single hyper-heuristic Sub_{Size} is 240 and Table 9 shows the results of 10 evolutionary PRs (EPRs) obtained through 10 trainings. At the same time, the fluctuating lines of HH-EGP and single hyper-heuristic (single-HH) under all distributions in 10 training times are shown in Fig.12.

Table 9 The objective values of single hyper-heuristic under the J120

Num	U1	U2	B1	B2	E
1	44.51	51.17	45.03	53.26	65.32
2	52.73	58.67	53.47	61.22	71.68
3	44.32	50.92	44.85	52.88	64.58
4	51.39	57.16	52.11	59.54	69.76
5	44.27	50.95	44.81	52.88	64.71
6	44.26	50.88	44.83	52.90	64.67
7	45.02	51.60	45.58	53.73	65.31
8	46.94	53.19	47.54	55.30	66.43
9	55.10	62.12	55.85	64.52	75.52
10	44.61	51.31	45.17	53.27	65.00
avg	47.315	53.797	47.924	55.95	67.298

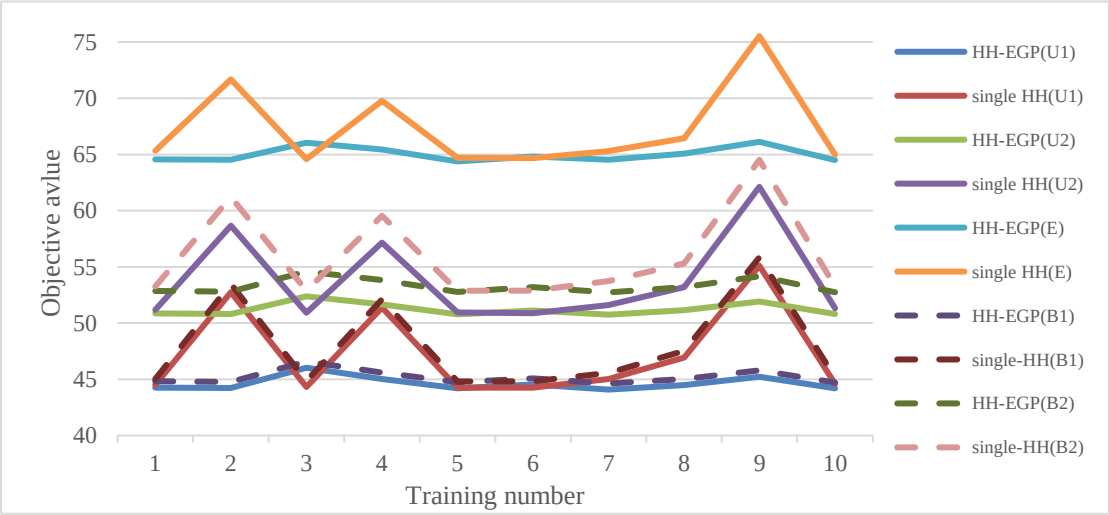


Fig.12 The fluctuation lines of HH-EGP and single hyper-heuristic under all distributions

It can be seen from Table 9 that the average objective value of the single hyper-heuristic under the same searches can be closer to HH-EGP than that of heuristics or meta-heuristics. However, the fluctuation curve in Fig.12 shows that the single hyper-heuristic in the way of evolving PR is unstable, and some EPRs are even worse than traditional PRs or meta-heuristics, which can prove that the training of single hyper-heuristic has a certain degree of risk. On the contrary, *HH-EGP is very stable*, which will reduce this risk, and further prove *the effectiveness of collaborative decision-making with multiple-heuristics* and the superiority of HH-EGP.

In addition, each subpopulation in HH-EGP can be regarded as an independent single

hyper-heuristic. Then, HH-EGP can be described as a group of EPRs participating in collaborative decision-making after Sub_{num} training times under the condition that Sub_{Size} is equal to 40. In order to explore whether this collaborative decision-making can improve the scheduling ability of single EPRs, we took the first training result of HH-EGP as an example and scheduled the 6 EPRs that made up HH-EGP separately. The results are shown in Table 10, where EPR1 to EPR6 represent the EPRs in decision ensemble of HH-EGP.

Table 10 The objective values of EPRs under the J120

	U1	U2	B1	B2	E
EPR1	45.10	51.82	45.64	53.99	66.42
EPR2	44.84	51.54	45.38	53.65	65.7
EPR3	46.25	53.76	46.84	56.41	70.12
EPR4	45.43	52.25	46.00	54.26	66.08
EPR5	45.18	51.89	45.76	53.91	65.74
EPR6	48.99	55.05	49.68	57.36	67.94
HH-EGP	44.27	50.86	44.82	52.87	64.56

By comparing Table 9 and Table 10, we can see that the scheduling ability of evolved EPRs from the single hyper-heuristic becomes weaker when the Sub_{Size} decreases, which leads to insufficient search. However, it can be seen from Table 10 that when 6 EPRs make collaborative decisions, the objective values will be better optimised, especially in the case of EPRs with poor scheduling ability, such as EPR6, which can be made up by the help of other ERPs. *The instability of the single hyper-heuristic* in Table 9 *can be improved, thus verifying the effectiveness of collaborative decision-making and HH-EGP.*

5.4 Effects of Sub_{Size} and Sub_{num} in HH-EGP

The validity and superiority of HH-EGP are verified by the above experiments. The influence of parameters Sub_{num} and Sub_{Size} on the HH-EGP performance is worthy of further exploration, that is, when HH-EGP is used to solve different SRCPSP under the expected search amount, more attention should be paid to let more EPRs participate in ensemble decision by weakening the evolution ability of each subpopulation or allocating more calculation to let each subpopulation evolve better EPR. Therefore, we designed five groups of experiments while keeping the total search amount unchanged: Sub_{num}=4 & Sub_{Size}=60, Sub_{num}=5 & Sub_{Size}=48, Sub_{num}=6 & Sub_{Size}=40, Sub_{num}=8 & Sub_{Size}=30 and Sub_{num}=10 & Sub_{Size}=24, to reduce Sub_{Size} by expanding Sub_{num}, while keeping the total search (Sub_{num}* Sub_{Size}) unchanged. The reason that two or three subpopulations are not involved in the experiment is that they will produce a lot of couple ties in the collaborative decision-making. According to the strategy described in Section 4.2, when the Sub_{num} are two or three, it is more likely to be the scheduling result of the first EPR in the ensemble. After training 10 times under all conditions, the objective values are shown in Table 11 (when Sub_{num} is 6, the results are shown in Table 6), and the mean values of different conditions under low variance (U1) and high variance (E) are shown in Fig.13 and Fig.14.

Table 11 The objective values of HH-EGP with different parameters under the J120

Condition	*Num*	U1	U2	B1	B2	E
Sub_{num}=4 Sub_{Size}=60	1	56.04	63.13	56.74	65.68	76.93
	2	45.28	51.87	45.83	53.97	65.89
	3	51.19	57.63	51.89	60.06	70.86
	4	44.42	51.04	44.95	53.14	65.13
	5	44.87	51.43	45.43	53.45	65.02
	6	45.23	51.93	45.81	54.14	66.14
	7	46.03	52.68	46.62	54.96	66.70
	8	44.65	51.20	45.23	53.25	64.82
	9	44.71	51.34	45.26	53.35	64.99
	10	44.43	51.03	44.97	52.98	64.68
	avg	46.685	53.328	47.273	55.498	67.116
Sub_{num}=5 Sub_{Size}=48	1	44.69	51.32	45.22	53.27	65.05
	2	46.53	53.36	47.11	55.35	67.16
	3	44.25	50.80	44.79	52.80	64.47
	4	45.30	51.78	45.88	53.87	65.28
	5	44.88	51.44	45.45	53.58	65.04
	6	44.43	51.03	45.00	53.09	64.77
	7	44.22	50.77	44.75	52.74	64.55
	8	44.81	51.49	45.36	53.59	65.62
	9	45.73	52.12	46.33	54.23	65.56
	10	44.39	50.99	44.96	53.1	64.98
	avg	44.923	51.51	45.485	53.562	65.248
Sub_{num}=8 Sub_{Size}=30	1	44.71	51.31	45.27	53.39	65.01
	2	44.32	50.99	44.87	53.17	65.2
	3	45.14	51.75	45.71	53.82	65.64
	4	44.55	51.12	45.09	53.14	64.76
	5	45.43	52.26	46.05	54.28	66.07
	6	44.70	51.25	45.25	53.27	64.83
	7	44.45	51.17	45.01	53.25	65.22
	8	44.41	51.10	44.95	53.18	65.06
	9	44.19	50.83	44.74	52.85	64.61
	10	44.19	50.74	44.74	52.77	64.48
	avg	44.609	51.252	45.168	53.312	65.088
Sub_{num}=10 Sub_{Size}=24	1	44.55	51.17	45.1	53.31	65.1
	2	44.23	50.80	44.75	52.76	64.45
	3	44.88	51.50	45.45	53.58	65.25
	4	45.25	51.99	45.83	54.04	65.71
	5	44.95	51.48	45.53	53.51	65.01
	6	44.25	50.86	44.8	52.86	64.65
	7	44.72	51.38	45.26	53.51	65.55

	8	46.54	53.11	47.18	55.17	66.57
	9	44.99	51.65	45.57	53.63	65.27
	10	45.17	51.85	45.75	54.08	65.96
	avg	44.953	51.579	45.522	53.645	65.352

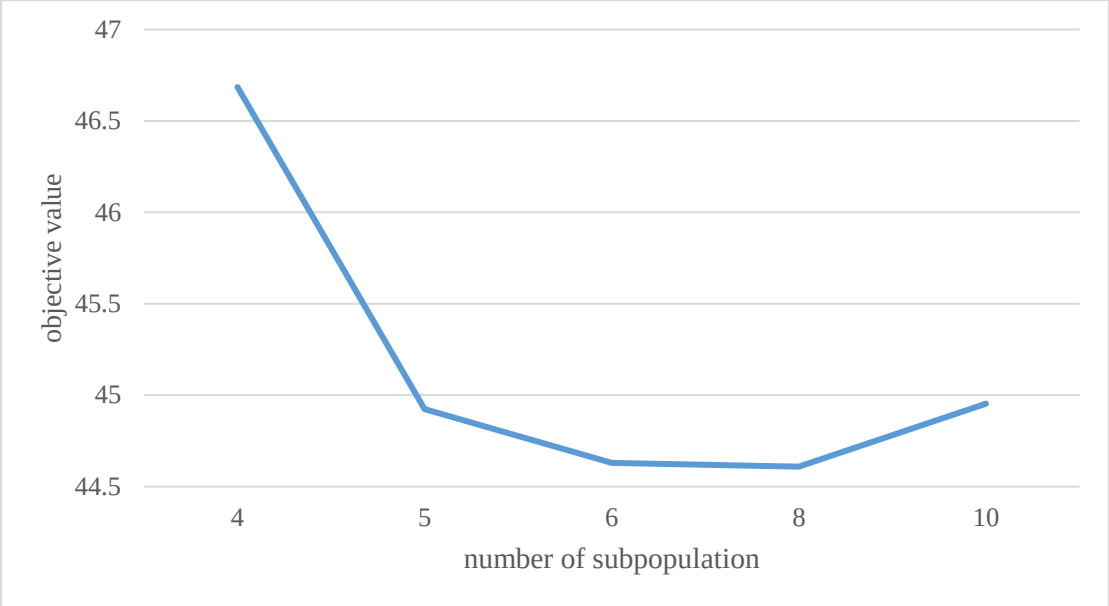


Fig.13 The mean value of HH-EGP with different Sub_{num} under low variance distribution

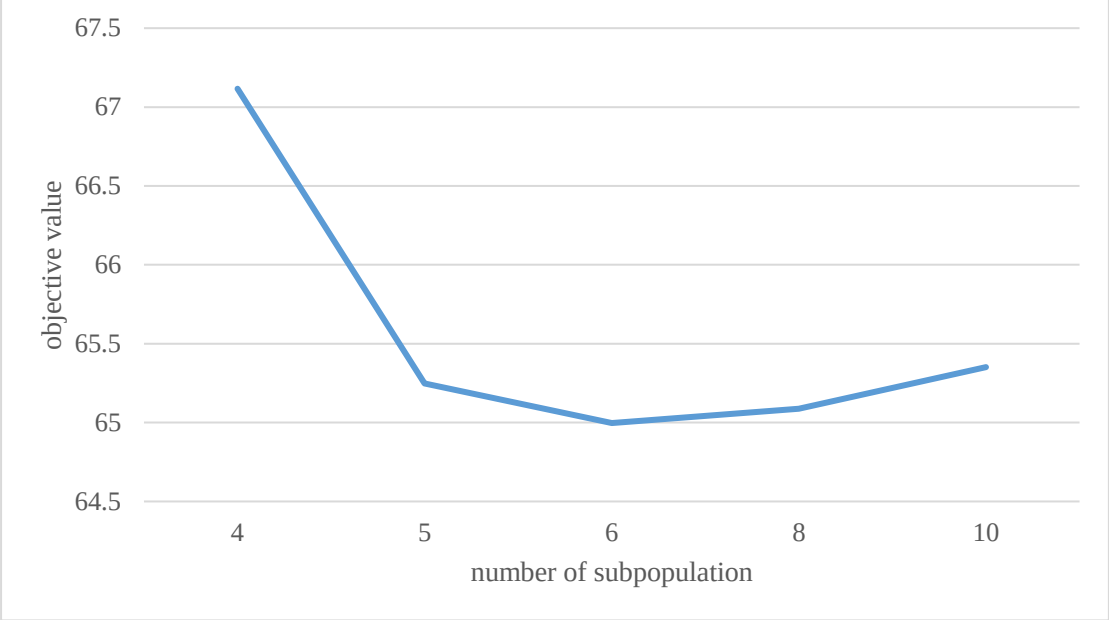


Fig.14 The mean value of HH-EGP with different Sub_{num} under high variance distribution

From the data in Table 6, Table 11 and Fig.13, it can be seen that the objective mean value curve of HH-EGP is similar to parabola, which will degrade the scheduling performance when Sub_{num} is too small or too large. The reason for this phenomenon is that when Sub_{num} is too small, the probability of tie will increase at each decision time, so the actual scheduling is more inclined to the first EPR in the ensemble and the voting proportion of a single EPR will be large. If there is

a poor EPR in the decision ensemble, the possibility that other EPRs can make up will be reduced. On the contrary, if Sub_{num} is too large, then Sub_{Size} of each subpopulation will be very small, so the amount of searching is insufficient, resulting in the inability to provide better EPRs. At the same time, by comparing Fig.13 and Fig.14, when the distribution instability increases, the lowest point of the curve will move to the left, which can prove that in the face of very unstable problem solving, too many EPRs in the decision set will have a bad impact on the decision.

To sum up, our experiment shows two characteristics of HH-EGP: 1) *when Sub_{num} is too small or too large, the performance of HH-EGP will be reduced due to the insufficient number of ERPs participating in ensemble decision or the insufficient search amount of each sub population*. Therefore, in the case of the same search amount, Sub_{num} tends to the median as much as possible; 2) *in the case of high variance distribution, the optimal point of Sub_{num} is more to the left than that of low variance distribution.* Therefore, it can be seen that the number of EPRs involved in ensemble needs to be reduced by using HH-EGP in the case of high variance distribution. Note that whether the two characteristics are consistent with all common distributions (such as triangle distribution) needs to be further verified in the future.

6 Conclusion

In this paper, an HH-EGP method is proposed to solve SRCPSP. To the best of our knowledge, this is the first time to apply the idea of hyper-heuristic scheduling and ensemble decision scheduling to SRCPSP. In order to achieve this goal, we have investigated the novel ways of implementing the proposed framework of HH-EGP, namely adopting RB-policies combined with the critical path method, modifying the function set, attribute set and coding structure applicable to SRCPSP, improving the genetic programming evolution and designing a sequence voting mechanism for collaborative scheduling. The effectiveness and superiority of this method are verified by experiments, and the influence of subpopulation number and subpopulation size on HH-EGP are analysed and explored. This method can not only improve the non-optimisation ability of PRs, but also do not need a lot of meta-heuristic iterations in the decision-making process, so it has a strong potential for future research.

At the same time, it can be seen from the experiments that HH-EGP has better effect than the single hyper-heuristic in the case that each subpopulation search is less than that of the single hyper-heuristic. Therefore, if HH-EGP can combine distributed computing technology such as multi-agent, it has a potential advantage. Since the evolution of each subpopulation in HH-EGP is independent, an agent can be assigned to each subpopulation to control the evolution, then HH-EGP with multi-agents can get better SRCPSP results in the same training time. Accordingly, if only the result of the single hyper-heuristic scheduling is needed, then HH-EGP can be allocated with small population size, so that the computing time is less under multi-agent than the single hyper-heuristic. Therefore, our future research on HH-EGP will focus on following two aspects. First, we will expand the problem by adding more dynamic factors, so as to more widely verify the performance of HH-EGP. Secondly, we will add distributed technology and machine learning to HH-EGP, the latter is to play the role of data analysis and feature extraction, so as to replace the existing known mathematical distribution. These technologies can provide some technical support for the intelligent manufacturing and even digital twin.

Acknowledgements

This research is supported by the Major State Basic Research Program in Sichuan Province of China (Grant number 20YYJC4377).The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions.

References

Anagnostopoulos, K., & Koulinas, G. (2012). Resource-constrained critical path scheduling by a GRASP-based hyperheuristic. *Journal of Computing in Civil Engineering*, *26*(2), 204-213.

Arkhipov, D., Battaïa, O., & Lazarev, A. (2019). An efficient pseudo-polynomial algorithm for finding a lower bound on the makespan for the Resource Constrained Project Scheduling Problem. *European Journal of Operational Research*, *275*(1), 35-44.

Ashtiani, B., Leus, R., & Aryanezhad, M. B. (2011). New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing. *Journal of Scheduling*, *14*(2), 157-171.

Ballestín, F. (2007). When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling*, *10*(3), 153-166.

Ballestin, F., & Leus, R. (2009). Resource-constrained project scheduling for timely project completion with stochastic activity durations. *Production and Operations Management*, *18*(4), 459-474.

Bettemir, Ö. H., & Sonmez, R. (2015). Hybrid genetic algorithm with simulated annealing for resource-constrained project scheduling. *Journal of Management in Engineering*, *31*(5), 04014082.

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, *5*(1), 11-24.

Boctor, F. F. (1996). Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, *34*(8), 2335-2351.

Browning, T. R., & Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, *126*(2), 212-228.

Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *107*(2), 272-288.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12), 1695-1724.

Chakrabortty, R. K., Sarker, R. A., & Essam, D. L. (2017). Resource constrained project scheduling with uncertain activity durations. *Computers & Industrial Engineering, 112*(Oct.), 537-550.

Chakrabortty, R. K., Sarker, R., & Essam, D. (2015). Resource constrained project scheduling: A branch and cut approach. In *Proceedings of the 45th international conference on computers and industrial engineering Metz* (Vol. 132).

Chand, S., Huynh, Q., Singh, H., Ray, T., & Wagner, M. (2018). On the use of genetic

programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences*, *432*, 146-163.
Chand, S., Singh, H., & Ray, T. (2019). Evolving heuristics for the resource constrained project scheduling problem with dynamic resource disruptions. *Swarm and evolutionary computation*, *44*, 897-912.
Chen, Z., Demeulemeester, E., Bai, S., & Guo, Y. (2018). Efficient priority rules for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, *270*(3), 957-967.
Chen, H., Ding, G., Zhang, J., & Qin, S. (2019). Research on priority rules for the stochastic resource constrained multi-project scheduling problem with new project arrival. *Computers & Industrial Engineering*, *137*, 106060.
Chen, W., Shi, Y. J., Teng, H. F., Lan, X. P., & Hu, L. C. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences*, *180*(6), 1031-1039.
Chen, W. N., & Zhang, J. (2013). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering*, 39(1), 1-17.
Creemers, S. (2015). Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *Journal of Scheduling*, *18*(3), 263-273.
Davari, M., & Demeulemeester, E. (2019). The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling*, *22*(2), 211-237.
Fang, C., Kolisch, R., Wang, L., & Mu, C. (2015). An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem. *Flexible Services and Manufacturing Journal*, *27*(4), 585-605.
Golenko-Ginzburg, D., & Gonik, A. (1997). Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics*, *48*(1), 29-37.
Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, *17*(2), 416-429.
Hart, E., & Sim, K. (2016). A hyper-heuristic ensemble method for static job-shop scheduling. *Evolutionary Computation*, *24*(4), 609-635.
Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *207*(1), 1-14.
Igelmund, G., & Radermacher, F. J. (1983). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks, 13*(1), 1-28.
Jia, Q., & Seo, Y. (2013). An improved particle swarm optimization for the resource-constrained project scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *67*(9-12), 2627-2638.
Kadri, R. L., & Boctor, F. F. (2018). An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, *265*(2), 454-462.
Kolisch, R. (1996a). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, *90*(2), 320-333.
Kolisch, R. (1996b). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, *14*(3), 179-192.
Koulinas, G. K., & Anagnostopoulos, K. P. (2012). Construction resource allocation and leveling

using a threshold accepting–based hyperheuristic algorithm. *Journal of Construction Engineering and Management*, *138*(7), 854-863.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences*, 277, 680-693.

Lamas, P., & Demeulemeester, E. (2016). A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, *19*(4), 409-428.

Li, H., & Womer, N. K. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, *246*(1), 20-33.

Lin, J., Zhu, L., & Gao, K. (2020). A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications*, *140*, 112915.

Luke, S., & Panait, L. (2001, July). A survey and comparison of tree generation algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 81-88). Morgan Kaufmann San Francisco, California, USA.

Mendes, J. J., Gonçalves, J. F., & Resende, M. G. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, *36*(1), 92-109.

Möhring, R. H., Radermacher, F. J., & Weiss, G. (1985). Stochastic scheduling problems II-set strategies. *Mathematical Methods of Operations Research, 29*(3),
65–104.

Moukrim, A., Quilliot, A., & Toussaint, H. (2015). An effective branch-and-price algorithm for the Preemptive Resource Constrained Project Scheduling Problem based on minimal Interval Order Enumeration. *European Journal of Operational Research*, *244*(2), 360-368.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2012). A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, *17*(5), 621-639.

Park, J., Nguyen, S., Zhang, M., & Johnston, M. (2015, April). Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. In *European Conference on Genetic Programming* (pp. 92-104). Cham: Springer.

Park, J., Mei, Y., Nguyen, S., Chen, G., & Zhang, M. (2018). An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Applied Soft Computing*, *63*, 72-86.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21-45.

Radermacher, F. J. (1981). Cost-dependent essential systems of ES-strategies for stochastic scheduling problems. *Methods of Operations Research*, *42*, 17-31.

Rostami, S., Creemers, S., & Leus, R. (2018). New strategies for stochastic resource-constrained project scheduling. *Journal of Scheduling*, *21*(3), 349-365.

Shan, S., Hu, Z., Liu, Z., Shi, J., Wang, L., & Bi, Z. (2017). An adaptive genetic algorithm for demand-driven and resource-constrained project scheduling in aircraft assembly. *Information Technology and Management*, 18(1), 41-53.

Sprecher, A., & Kolisch, R. (1996). PSPLIB—a project scheduling problem library. *European Journal of Operational Research*, *96*, 205-216.

Stork, F. (2001). *Stochastic resource-constrained project scheduling* (Ph.D. thesis). Technical University of Berlin

Tahooneh, A., & Ziarati, K. (2011, June). Using artificial bee colony to solve stochastic resource constrained project scheduling problem. In *International Conference in Swarm Intelligence* (pp. 293-302). Berlin, Heidelberg: Springer.

Tsai, Y. W., & Gemmill, D. D. (1998). Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research*, *111*(1), 129-141.

Villafáñez, F., Poza, D., López-Paredes, A., Pajares, J., & del Olmo, R. (2019). A generic heuristic for multi-project scheduling problems with global and local resource constraints (RCMPSP). *Soft Computing*, *23*(10), 3465-3479.

Wang, J., Zhang, J., & Wang, X. (2018). A data driven cycle time prediction with feature selection in a semiconductor wafer fabrication system. *IEEE Transactions on Semiconductor Manufacturing*, 31(1), 173-182.

Wang, Y., He, Z., Kerkhove, L. P., & Vanhoucke, M. (2017). On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. *Computers & Industrial Engineering*, *114*, 223-234.

Zhou, Y., Yang, J. J., & Zheng, L. Y. (2019). Multi-agent based hyper-heuristics for multi-objective flexible job shop scheduling: A case study in an aero-engine blade manufacturing plant. *IEEE Access*, *7*, 21147-21176.