Expert Systems With Applications 203 (2022) 117029

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Multi-objective stochastic project scheduling with alternative execution methods: An improved quantum-behaved particle swarm optimization approach

Tao Zhou [a], Qiang Long [b], Kris M.Y. Law [a,c,*], Changzhi Wu [d]

[a] School of Engineering, Faculty of Science Engineering and Built Environment, Deakin University, Geelong, VIC, 3217, Australia
[b] School of Science, Southwest University of Science and Technology, Mianyang, 621010, China
[c] Department of Industrial Engineering Management, University of Oulu, Finland
[d] School of Management, Guangzhou University, Guangzhou, 510006, China

A R T I C L E I N F O

Keywords:
Quantum-behaved particle swarm optimization
Two-stage learning strategy
Chaotic operators
Multi-objective
Stochastic durations
Resource-constrained project scheduling

A B S T R A C T

This paper addresses the multi-objective resource-constrained project scheduling problems with stochastic activity durations and alternative execution methods. Three objectives are considered: minimizing expected makespan, expected cost and robustness. Chance constrained programming is applied for formulating this stochastic problem. A hybrid approach that integrates sample average approximation (SAA) and an improved multi-objective chaotic quantum-behaved particle swarm optimization (MOCQPSO) algorithm is proposed. To improve the diversity of solutions and enhance the global search ability, a two-stage learning strategy that balances the exploration and the exploitation is proposed for MOCQPSO. In addition, chaotic operators including chaotic initialization, crossover and mutation are also introduced. Six benchmark functions and an instance generator based on the RCPSP dataset of PSPLIB are designed to validate the performance of the proposed algorithm. The experimental results demonstrate that our proposed method outperforms the original algorithms in solution diversity and quality.

1. Introduction

Project delay and budget overrun have been the two main prevailing issues that continuously plague the project developers and contractors in the industries, such as the construction industry. Over 70% of the construction projects suffered from delays, and among these projects, 75% of them spent 50% more than their initial budget (Ma et al., 2015). With the ever-growing competition, shorter duration and less expenditure enable the project developers to increase their return on capital and the contractors to gain more profit and avoid inflation (Zhou et al., 2013). Therefore, project stakeholders, including project developers and engineering contractors, strive to reduce the duration of the project and reduce the expenditure of operations. In this case, a lot of attention has been aroused for the research on the well-known NP-hard resource-constrained project scheduling problem (RCPSP) (Lamas & Demeulemeester, 2016). The classical RCPSP aims to generate an optimal schedule to minimize the whole project's makespan satisfying the precedence relationship between activities and the resource constraints simultaneously (Afshar-Nadjafi et al., 2017). For example, Van Peteghem and Vanhoucke (2010), Coelho and Vanhoucke (2011) and Cheng et al. (2015) proposed to minimize the total makespan of the project for an RCPSP with consideration of the constraints for both renewable and non-renewable resources. These studies aim to deal with deterministic project structures, which means that the activities and the corresponding durations included in a project are fixed. However, in reality, the activity durations and the project structures could vary due to various factors.

On the one hand, the projects in some industries, such as construction projects, are less predictable than the manufacturing industry in terms of project makespans due to their complexity and highly heterogeneous activities (Ribeiro et al., 2013). Each project is unique, and its performance is affected by many factors, such as human behaviours and weather conditions. Therefore, the duration of activity should not be deemed as fixed. Conversely, it is an uncertain parameter. One of the most accepted approaches for dealing with indefinite activity duration is assuming that the parameter of activity duration is a stochastic variable with a predefined probability distribution (Lamas & Demeulemeester, 2016). On the other hand, for a mega-project, there are always

* Corresponding author at: School of Engineering, Faculty of Science Engineering and Built Environment, Deakin University, Geelong, VIC, 3217, Australia.
E-mail addresses: tao.zhou@deakin.edu.au (T. Zhou), longqiang@swust.edu.cn (Q. Long), kris.law@deakin.edu.au (K.M.Y. Law), changzhiwu@gzhu.edu.cn (C. Wu).

https://doi.org/10.1016/j.eswa.2022.117029
Received 14 February 2021; Received in revised form 17 November 2021; Accepted 27 March 2022
Available online 6 May 2022

several potential execution methods or solutions to choose from for accomplishing a particular task. For example, multi-mode RCPSP which defines that each activity could be executed in various modes has been widely studied (Afshar-Nadjafi, 2014; Qi et al., 2014). For a multi-mode RCPSP, each activity is assumed to have fixed precedence relations, which indicates that the mode selection could not alter the project structure. However, in many scenarios, selecting alternative execution methods for a specific task could trigger new activities. Kellenbrink and Helber (2015) studied the RCPSP with a flexible project structure and implemented their proposed model on the aircraft turnaround process. In their study, the selections of alternative arrival options would activate the corresponding new activities.

To deal with both uncertain activity durations and flexible project structures in RCPSP, in this paper, we take stochastic durations with a pre-defined probability distribution and alternative execution methods that could trigger additional activities into consideration. Additionally, decision-makers have to measure the uncertainty of conditions on many occasions and simultaneously determine the best scheduling plan based on multiple performance criteria. Therefore, this paper extends the classical RCPSP to a Multi-objective Stochastic Resource-Constrained Project Scheduling Problem with Alternative execution methods, denoted as MSRCPSP-A. A chance-constrained multi-objective model is proposed to minimize the total makespan and total resource costs. On most occasions, the demands of non-renewable materials such as concrete and pipes in a construction project are generally well planned in the design phase with a fixed cost. In contrast, the expenses spent on equipment and other reusable resources, such as cranes and scaffolding materials, would contribute to a higher total cost. These resources are customarily rented instead of purchased or owned (Ballestín, 2008). Therefore, in the proposed model, the resource costs mainly focus on time-dependent resources. Besides, to achieve more robust schedules, a robustness measurement is introduced as an objective function in the proposed model.

This paper proposes a hybrid approach to solve the MSRCPSP-A. The sample average approximation (SAA) method is applied as the outer layer algorithm to generate deterministic sample problems to deal with the stochastic activity duration. SAA is an approach for solving large scale stochastic optimization problems based on Monte Carlo simulation. It approximates the expected value of the objective function by a sample average generated from a group of random samples (Verweij et al., 2003). The effectiveness of SAA has been demonstrated in various stochastic problems, such as resource-constrained project scheduling problems (Lamas & Demeulemeester, 2016) and supply chain design (Schütz et al., 2009). Quantum-behaved particle swarm optimization (QPSO) has shown its advanced global search ability (Singh & Mahapatra, 2016) and chaotic operators are proven effective in enhancing exploration capability (Feng et al., 2017). An improved multi-objective chaotic QPSO (MOCQPSO) algorithm is utilized as the inner layer algorithm to solve each sampled problem. Motivated by Liu et al. (2020) and Xin-gang et al. (2020), we introduce a two-stage learning strategy and chaotic operators to improve the diversity of the non-dominated solutions. The two-stage learning strategy utilizes a dynamic learning coefficient to balance exploration and exploitation. A wider solution space is searched during the exploration stage to enhance the diversity of solution. During the exploitation stage, global best solutions are used to guide the convergence. A chaotic initialization based on the logistic mapping is introduced to enhance the randomness of the initial population. In addition, to overcome the contingency of trapping into local optima, a chaotic crossover operator and a chaotic mutation operator are applied. In general, the contributions of this paper is fivefold:

- A multi-objective resource-constrained project scheduling problem with stochastic activity durations and alternative execution methods (MSRCPSP-A) is presented.
- A chance-constrained multi-objective model with objective functions of minimizing the expected makespan, expected cost and robustness, is proposed.
- A hybrid approach that integrates the SAA and an improved multi-objective chaotic QPSO (MOCQPSO) algorithm is introduced to solve the MSRCPSP-A.
- A two-stage learning strategy and chaotic operators are designed to enhance the performance of the QPSO algorithm.
- An instance generator is designed to compare the performance of the MOCQPSO with its competitors.

The rest of this paper is organized as follows. Section 2 introduces the literature review on both RCPSPs and meta-heuristic algorithms. Section 3 describes the mathematical formulation of the proposed MSRCPSP-A problem and the solution encoding and decoding processes. In Section 4, the framework of a hybrid algorithm that integrates the SAA and MOCQPSO is presented. A two-stage learning strategy and chaotic operators including chaotic initialization, chaotic crossover and mutation are introduced to enhance the performance of QPSO for dealing with multi-objective optimization. In Section 5, experimental project instances are generated and used to evaluate and validate the proposed model and the performance of the proposed algorithm. Finally, the conclusion of this paper is presented in Section 6.

2. Literature review

Over the last few decades, an increasing amount of studies has been conducted on different variants of RCPSP (Habibi et al., 2018) and the meta-heuristic algorithms for resolving these RCPSPs (Pellerin et al., 2020). In this section, a brief review of these studies is provided below.

2.1. Resource-constrained project scheduling problem

For the classical RCPSP, decision-makers aim to determine the start time of activities to minimize the total makespan of the projects (Habibi et al., 2018). In addition to minimizing the makespan, minimizing delay is another popular time-based objective for RCPSP. For example, García-Nieves et al. (2018) proposed a model of RCPSP to minimize the weighted tardiness where tardiness was defined as the delay of executing operations. Meanwhile, various cost-based objectives have also been proposed and studied. Chen et al. (2010) constructed their mathematical model to minimize the final net present value (NPV) of the project using discounted cash flow analysis. The delay penalty is another cost-based criterion that would be considered in real-world project management. The objective of minimizing the total cost of tardiness penalty is commonly proposed for RCPSP (Rajeev et al., 2015). Generally, a construction project's total cost consists of equipment and resource renting, materials acquisition and labour cost (Zhou et al., 2013). Afshar-Nadjafi (2014) and Qi et al. (2014) worked with the multi-mode RCPSP to minimize the resource availability cost with a given due date. Afshar-Nadjafi et al. (2017) presented a novel model for the resource renting in project scheduling problem which encompasses the availability cost of rental resources and the tardiness penalty. In contrast to the fixed rental cost, the authors assumed that the renting cost is associated with the resource's availability length. Moreover, in many cases, more than one performance criteria or objective would be considered and these objectives are often in conflict with each other. Tirkolaee et al. (2019) addressed the multi-objective multi-mode RCPSP intending to maximize the NPV and minimize the completion time concurrently. Nemati-Lafmejani et al. (2019) proposed an bi-objective optimization model to deal with the time-cost trade-off problem for RCPSP. In their model, the selection of contractors was also considered.

In reality, RCPSPs may face various uncertainties. One type of uncertainties that has been widely studied is the uncertain activity duration which is often modelled as a stochastic variable (Lamas &

Demeulemeester, 2016). The three most frequently adopted models for the stochastic RCPSP (SRCPSP) are the expected cost model (ECM) for minimizing the expected cost of the project (Sobel et al., 2009), the chance-constrained model (CCM) with a predetermined confidence level for constraints (Wang et al., 2015) and the probability maximization model (PMM) for maximizing the probability of total cost limited within the budget (Ke & Liu, 2005). Besides stochastic programming, robust optimization has also been proposed for dealing with RCPSP with uncertain activity durations (Balouka & Cohen, 2019; Liang et al., 2020). Apart from the uncertain activity duration, uncertainty also comes from the flexible project structures. Tao and Dong (2017) described an RCPSP with alternative activity chains (RCPSP-AC) with the aim to select one activity chain and schedule the selected activities to minimize the makespan. Tao et al. (2018) extended their model by incorporating stochastic activity durations. Servranckx and Vanhoucke (2019) further considered nested and linked alternative subgraphs for RCPSP and introduced a framework to classify projects with different types of alternative subgraphs.

In recent years, some researchers studied the RCPSPs with uncertainties in a multi-objective setting. Wood (2017) reviewed the time–cost–quality trade-off problem for a gas and oil project. In their multi-objective model, work-item durations and costs were expressed as probability distributions and randomly sampled to generate deterministic cases. A multi-objective model for the scheduling problem in the energy industry proposed by Fu et al. (2019) considered stochastic tardiness and aimed to minimize the makespan and the total energy consumption. A chance-constrained programming approach was applied for formulating stochastic tardiness. Yeganeh and Zegordi (2020) introduced resiliency measures based on the floating times of activities with uncertain durations and the risks of disruption from one activity to its successors. There are three objectives: makespan minimization, and maximizations of the weighted sum of floating times and the total activity values. Hauder et al. (2020) proposed a new RCPSP with structure flexibility and time flexibility to minimize the makespan and maximize the time balance.

2.2. Meta-heuristic algorithms

Many meta-heuristic algorithms have been proven effective to solve variants of RCPCP. Proposed by Eberhart and Kennedy (1995), the particle swarm optimization (PSO) algorithm is one of the most widely studied meta-heuristic algorithms in scheduling problems due to the advantages in computational feasibility and consistency in performance (Joy et al., 2016; Nouiri et al., 2018). Since then, various improvements have been proposed for PSO, which have effectively solved global optimization problems. Zhang et al. (2019) proposed a decision variable clustering method with multi-objective PSO (MOPSO) algorithm to search for multiple Pareto optimal solutions and maintain the diversity. Tang et al. (2019) proposed a hybrid discrete PSO integrated with simulated annealing (SA) algorithm for a multi-objective job-shop scheduling problem. In the proposed algorithm, PSO was applied for the global search and it was further improved by introducing a new initialization method and a novel global best selection approach. Sun et al. (2011) proposed the quantum-behaved PSO (QPSO) algorithm based on quantum mechanics. Unlike the classical PSO whose particles are described by their corresponding positions and velocities in Newtonian mechanics, QPSO assumes that the particle swarm system satisfies the hypothesis of quantum mechanics where positions and velocities could not be determined simultaneously according to the uncertainty principle (Xin-gang et al., 2020). Even though QPSO has shown its advanced global search ability compared to PSO (Singh & Mahapatra, 2016), it also suffers from the fast decline in diversity and the insufficient ability to escape local optima. To deal with these disadvantages, studies have been conducted to improve the performance of QPSO by introducing mutation operator from genetic algorithm (GA) (Singh & Mahapatra, 2016), cultural evolution mechanism (Liu et al., 2016), differential evolution crossover operator (Xin-gang et al., 2020) and so on. However, two main challenges remain to extend QPSO to a multi-objective context: the fast decline of solution diversity and the insufficient ability to escape local optima (Xin-gang et al., 2020). Many other meta-heuristic algorithms have also been proposed to resolve multi-objective RCPSP such as NSGA-II (Yeganeh & Zegordi, 2020), multi-objective differential evolution (MODE) (Nguyen et al., 2019) and artificial algae algorithm (AAA) (Tao et al., 2018).

3. Problem statement and mathematical modelling

This paper aims to deal with the multi-objective stochastic project scheduling with alternative execution methods (MSRCPSP-A). In our proposed scenario, selecting execution methods for a specific task would trigger a series of exclusive optional activities. These optional activities and the mandatory activities of the project will then be scheduled according to their precedence relations as well as resource constraints. As an extension of RCPSP, the proposed MSRCPSP-A problem is described and formulated in the following sections.

3.1. Problem description

Considering a project represented by an activity-on-node (AON) network $G = (\mathscr{I}, \mathscr{V})$, where $\mathscr{I} = \{1, 2, \dots, I\}$ is the set of nodes for all possible activities and $\mathscr{V}$ represents the set of arcs connecting the nodes. Activity 0 and $I + 1$ are dummy nodes that stand for the starting and completing nodes of the project. If activity i and j satisfy the precedence relationship $(i, j) \in \mathscr{V}$, then activity j must not start before completing activity i. In our proposed problem, alternative execution methods are considered for a specific task. Assuming the set of tasks with alternative methods is denoted by $\mathscr{M} = \{1, 2, \dots, M\}$, the possible alternative methods for task m is represented by $\mathscr{O}_m$ and the optional activities triggered by method l is $\mathscr{H}_l$. In addition, there are several types of time-dependent resources $\mathscr{N} = \{1, 2, \dots, N\}$ required for conducting operations for activities. The demand for these resources for each activity would vary, but the total capacity of each resource is limited. These resources will be charged based on the time of utilization. According to the descriptions mentioned above, the proposed MSRCPSP-A problem aims to find a series of solutions for the selection of alternative execution methods and the schedule of selected activities to (1) minimize the total makespan of the project; (2) minimize the total cost of time-dependent resources; (3) minimize the deviation of makespan and cost and their expected values (i.e. robustness).

To better explain this problem, an artificial example is illustrated by the AON network shown in Fig. 1. Two alternative methods can be selected for executing the same task triggered by activity 1 and different optional activities are involved for each method. Activity 1 and 12 are two mandatory activities that will always be selected for scheduling. The arrows indicate the precedence relations among activities; for example, activity 3 can only be scheduled after the completion of activity 2. The alternative execution methods are exclusive which means only one method can be selected for the task. Table 1 shows the information about the activities including their precedence relations, average duration and the demand for resources (only one resource is considered in the example). This example only presents us with a single task, while many tasks could be involved in a complex project. Before proceeding to the mathematical modelling of the proposed problem, several assumptions and premises need to be clarified:

(a) For each task, only one alternative method can be selected.
(b) Each decision-making moment for method selection is independent of others. Therefore, the previous decision on selection would not impact the subsequent ones.
(c) The duration of each activity is assumed as stochastic with a known distribution from empirical experience.

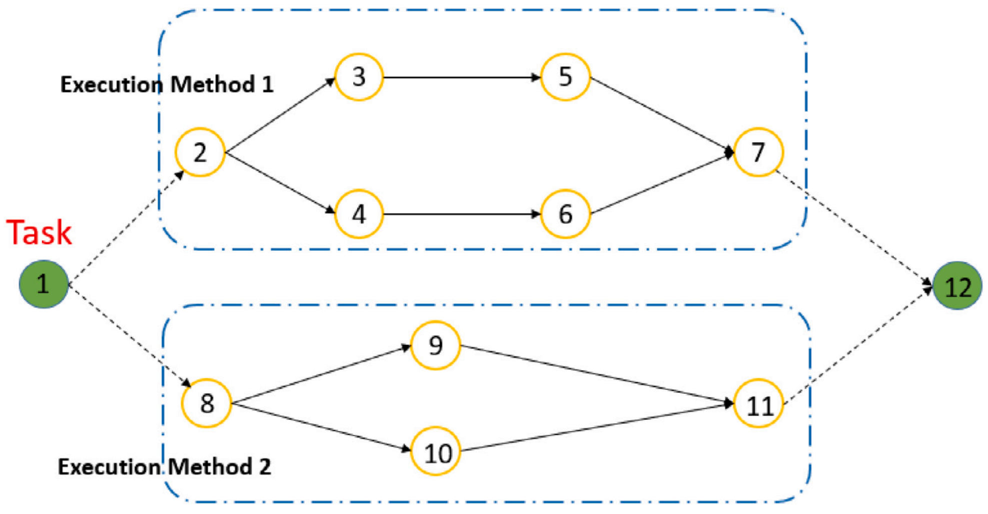

Fig. 1. Example of MSRCPSP-A.

Table 1
Example of alternative execution methods.

Execution method	Optional activity	Predecessor	Resource demand	Duration
Method 1	2	1	2	2
	3	2	4	1
	4	2	2	3
	5	3	1	2
	6	4	3	3
	7	5, 6	3	1
Method 2	8	1	3	4
	9	8	2	3
	10	8	1	1
	11	9, 10	3	2
	Mandatory activity			
Both Methods	1	–	1	1
	12	7, 11	2	2

3.2. Mathematical formulation

3.2.1. General mathematical model

Following the problem description above, the proposed MSRCPSP-A problem is formulated as a multi-objective mathematical model and the notations are shown in Table 2. Apart from two common objectives for RCPSP, which minimize the total project duration and the total cost of resources, minimizing the robustness measurement is also considered in this model because of the stochastic feature of activity durations. The mathematical model is presented as follows:

$$minimize \quad f_1 = S_{I+1} \tag{1}$$

$$minimize \quad f_2 = \sum_{n=1}^{N}\sum_{i=1}^{I} r_{in} * x_i * l_n * \epsilon_i \tag{2}$$

$$minimize \quad f_3 = robustness \tag{3}$$

Subject to

$$x_i = 1, \forall i \in \mathscr{E} \tag{4}$$

$$\sum_{l \in \mathscr{O}_m} y_l = 1, m \in \mathscr{M}, \mathscr{O}_m \in \mathscr{N} \tag{5}$$

$$x_i = y_l, \forall i \in \mathscr{H}_l \tag{6}$$

$$(S_i + \epsilon_i) * x_i * x_j \leq S_j, \forall (i,j) \in \mathscr{V} \tag{7}$$

$$\sum_{i \in P_t} r_{in} \leq R_n, \forall n \in \mathscr{N}, t \in \mathscr{T} \tag{8}$$

$$S_i \in \mathscr{T}, \forall i \in \mathscr{I} \tag{9}$$

$$x_i \in \{0,1\}, \forall i \in \mathscr{I}, m \in \mathscr{M} \tag{10}$$

$$y_l \in \{0,1\}, \forall l \in \mathscr{L} \tag{11}$$

Objective function (1) represents the minimization of the start time of dummy node $I+1$ and Objective function (2) minimizes the total cost for time-dependent resources where ϵ_i is the stochastic duration of activity i. The measurement of robustness in (3) is introduced in the next section. Constraint (4) guarantees that all the mandatory activities are selected and constraint (5) ensures that only one execution method is chosen for each task. Constraint (6) means that when method l is selected, its triggered optional activities will also be chosen. Constraint (7) represents that the precedence relationship for activities should be satisfied and constraint (8) represents the usage of all resources at any time should not exceed the capacity. Eq. (9) defines the start time of each activity i which is an integer number. Eqs. (10) and (11) define two decision variables with binary values: the selection of activity i denoted by x_i and the selection of alternative method l denoted by y_l.

3.2.2. Robustness measurement

The robustness in RCPSP is generally defined as the difference between the planned makespan and the realized makespan after disturbance (Xiong et al., 2012). Except for the project makespan, the cost is another consideration. The deviation of the makespan would alter the cost of renting equipment as well as workforce salary. In this paper, we adopt an efficiency-focused robustness measure proposed by Shen et al. (2017). The definition of robustness measurement is presented as follows:

$$robustness = \sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\max((\frac{f_1^q - \bar{f}_1}{\bar{f}_1}, 0))^2} + \lambda\sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\max((\frac{f_2^q - \bar{f}_2}{\bar{f}_2}, 0))^2} \tag{12}$$

where Q is the number of realizations/uncertain scenarios, f_1^q and f_2^q are the values of the first and second objective functions (i.e. makespan and total cost) under the qth uncertain scenario and $\bar{f}_1$ and $\bar{f}_2$ represent the expected makespan and expected total cost. Excessive makespan and cost are penalized for minimizing the impacts induced by uncertainty and λ is a coefficient that balances the robustness for makespan and cost.

Table 2
Notations.

Sets and indices:	
$\mathscr{I} = \{1, \dots, I\}$	set of all possible project activities indexed by i, j.
$\mathscr{V} \subseteq \mathscr{I}^2$	Immediate precedence relations among project activities, where $(i, j) \in \mathscr{V}$ indicates activity j must start after activity i's completion.
$\mathscr{M} = \{1, \dots, M\}$	set of tasks with alternative methods indexed by m.
$\mathscr{L} = \{1, \dots, L\}$	set of all possible alternative methods indexed by l.
$\mathscr{N} = \{1, \dots, N\}$	set of all types of time-dependent resource indexed by n.
$\mathscr{T} = \{1, \dots, T\}$	set of time slots indexed by t which represents time interval $[t-1, t)$.
Parameters:	
$\mathscr{O}_m \subset \mathscr{N}$	The set of candidate alternative methods for task m.
$\mathscr{E} \subset \mathscr{N}$	Mandatory activities.
$\mathscr{H}_l \subset \mathscr{I}$	The optional activities for method l.
r_{in}	demand for time-dependent resource n for activity i.
R_n	Capacity of resource n.
ϵ_i	duration of activity i.
l_n	cost per unit time for resource n.
Variables:	
S_i	start time of activity i.
P_t	Set of activities in execution at time t.
$x_i \in \{0, 1\}$	$x_i = 1$, if activity i is selected; otherwise, $x_i = 0$.
$y_l \in \{0, 1\}$	$y_l = 1$, if the alternative resource l is selected; otherwise, $y_l = 0$.

3.2.3. Chance-constrained formulation

The chance-constrained programming (CCP) was initially proposed by Charnes and Cooper (1959) for dealing with optimization problems with uncertainty. Over the past few decades, CCP has been constantly applied for coping with stochastic project scheduling problems. For example, Bruni et al. (2011) and Ma et al. (2016) proposed the chance-constrained models for resource-constrained project scheduling problems with stochastic activity durations. The main feature of the CCP method is that the constraints with uncertainty will hold with a confidence level denoted as $1-\alpha$ (Lamas & Demeulemeester, 2016). The constraints (7) and (8) can then be replaced by the following constraints:

$$P_r\left\{(S_i+\epsilon_i) * x_i * x_j \le S_j\right\} \ge 1-\alpha, \forall j \in \mathscr{I}, \forall (i,j) \in \mathscr{V} \tag{13}$$

$$P_r\left\{\sum_{i\in P_t} r_{in} \le R_n\right\} \ge 1-\alpha, \forall i \in \mathscr{I}, \forall n \in \mathscr{N}, t \in \mathscr{T} \tag{14}$$

This formulation ensures the constraints (7) and (8) are not violated with a chance of $(1-\alpha)$. This chance-constrained RCPSP problem has been proven to be NP-hard according to Wang et al. (2015) and Davari and Demeulemeester (2019).

Based on the realization-based reformulation of the above model, the vector of duration $\boldsymbol{\Gamma} = \left(\tilde{\epsilon}_0, \tilde{\epsilon}_1, \dots, \tilde{\epsilon}_{I+1}\right)$ can be represented by a finite supporting set $\boldsymbol{\Xi} = \left\{\epsilon^1, \dots, \epsilon^{|\Xi|}\right\}$ of realizations, where each ϵ^{ξ} represents a vector of durations $\epsilon^{\xi} = (\epsilon_0^{\xi}, \epsilon_1^{\xi}, \dots, \epsilon_{I+1}^{\xi}) \in \boldsymbol{\Xi}$. Each realization ϵ^{ξ} occurs with a certain probability $Pr(\boldsymbol{\Gamma} = \epsilon^{\xi})$ and the summation of the probabilities of all realizations equals one. Constraints (13) and (14) are hard to tackle. Alternatively, we decide to ensure the feasibility of each solution by finding a sufficient subset of realizations for which the solution is feasible. Assuming a subset W is a sufficient subset of realization, hence, $\sum_{\epsilon^{\xi}\in W} Pr(\boldsymbol{\Gamma} = \epsilon^{\xi}) \ge 1-\alpha$. Let $\mathscr{W}_{\Xi}$ be the set of sufficient subsets of $\boldsymbol{\Xi}$. The formulation can be transformed as below:

minimize (1), (2) subject to constraints (4)–(6), (9)–(11) and

$$x_i * x_j * (S_i + \epsilon_i^{\xi}) \le S_j, \forall j \in \mathscr{I}, \forall (i,j) \in \mathscr{V}, \epsilon^{\xi} \in W \tag{15}$$

$$(\sum_{i\in P_t^{\xi}} r_{in}) \le R_n, \forall i \in \mathscr{I}, \forall t \in \mathscr{T}, \forall \epsilon^{\xi} \in W \tag{16}$$

$$\sum_{\epsilon^{\xi}\in W} Pr(\boldsymbol{\Gamma} = \epsilon^{\xi}) \ge 1-\alpha \tag{17}$$

$$W \in \mathscr{W}_{\Xi} \tag{18}$$

3.3. Solution representation and decoding

With reference to the transformation scheme presented in Zhang et al. (2005), the solution representation of the proposed problem consists of two lists, namely the list of activity selections (ASL), $ASL = \{\theta_1, \theta_2, \dots, \theta_I\}$, and the list of priority values (PVL), $PVL = \{p_1, p_2, \dots, p_I\}$, where θ_i stands for the selection of activity i with binary values (i.e. 0 or 1) and p_i is the priority value for activity i. A smaller p_i means a higher priority of activity i.

The constraints handling is vital to transform these two lists into feasible solutions. First of all, mandatory activities should always be selected. Secondly, the selection of optional activities depends on the selection of alternative execution methods for each task. Therefore, a method selection list (MSL) is introduced to replace ASL in the solution representation, where $MSL = \{\beta_1, \beta_2, \dots, \beta_L\}$ and $\beta_l \in \{0, 1\}$. In addition, only one method for each task can be selected and all the optional activities triggered by the selected method are selected.

A two-step schedule generation scheme (SGS) is proposed to decode the MSL and PVL into a feasible list of starting times of activities (STL). The values in the MSL obtained through meta-heuristic algorithms are not necessarily integers. Therefore, a Sigmoid function, $Sigmoid(x) = \frac{e^x}{e^x+1}$, is applied to convert them to values within the range of $[0, 1]$, and the execution method with the highest value among other alternative methods for a specific task is selected. Accordingly, the corresponding optional activities are also included in the schedule. In the second step, activities will be scheduled one by one based on their priority value in PVL and the precedence relationship. For each iteration, a set of candidate activities is formed by the immediate successors of activities that have been selected and stored in the list called *Sequence*. The candidate activity with the smallest values indexed by *ind* in PVL is selected to put into *Sequence*. t stands for the earliest start time of activity *ind* which equals to the maximum completion time of its prior activities, denoted by $max\{s_i + \epsilon_i | (i, ind) \in \mathscr{V}\}$. To satisfy the resource constraints, t will be updated to the minimum completion time of the activities in the process at time t, hence $t = min(STL[i] + \epsilon_i)$ and $i \in P_t$. Until all resource constraints are satisfied, activity *ind* will be eliminated from the set of candidate activities. The starting time of *ind* in STL equals to the current value of t.

The procedure of SGS is illustrated, as shown in Fig. 2, using the example in Section 3.1. The solution representation consists of initial PVL and MSL, where the first 12 cells represent the priority values for each activity. The last two cells represent the selection of execution methods for the task. Following Step 1 in Algorithm 1, the PVL is sorted in descending order where rank 1 means the highest priority.

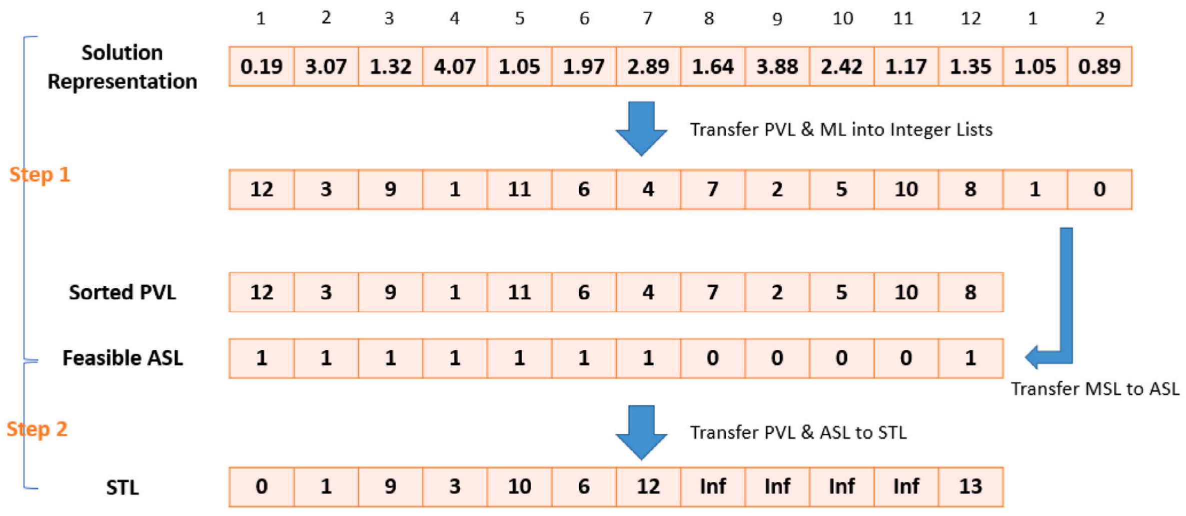

Fig. 2. Example of SGS.

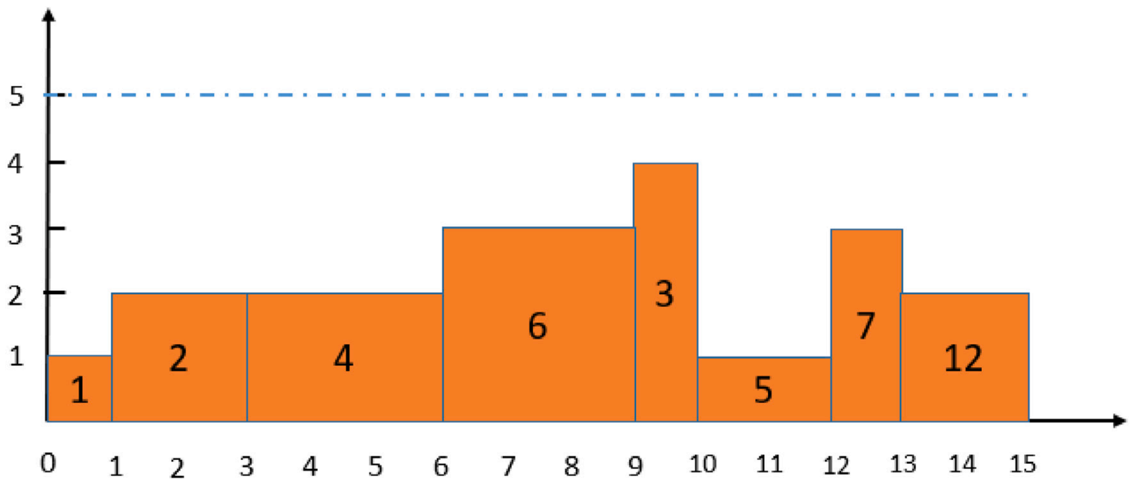

Fig. 3. Example of Schedule.

However, precedence relations should be considered first. For example, activity 1 only has a rank of 12 but it has to be scheduled first. MSL is transferred into a feasible ASL ensuring that only one method is selected for each task. In Step 2 of SGS, PVL and ASL are compiled to feasible STL with resource constraints and precedence relations. Fig. 3 shows the schedule of the example. As we can observe, activities 4 and 6 are placed before 3 as they have a higher priority according to PVL.

4. Proposed hybrid algorithm

4.1. Sample average approximation

As the size of the finite supporting set of realization described in the chance-constrained model is often too large, the SAA method is adopted in this paper to generate a much smaller set $\hat{\Xi}$ of q independent realizations with identically distributed (i.i.d) random variables based on Monte Carlo sampling to approximate the original set Ξ. Hence, the corresponding SAA problem can be formulated by replacing constraints (15)–(18) with following ones and the $\hat{\alpha}$ is the significance level for this SAA problem. It is worth noting that $1-\hat{\alpha} > 1-\alpha$.

$$y^{\xi} * x_i * x_j * (S_i + \epsilon_i^{\xi}) \leq S_j, \forall j \in \mathscr{I}, \forall (i,j) \in \mathscr{V}, \forall \xi = 1,\dots,q \tag{19}$$

$$v^{\xi} * (\sum_{i \in P_t^{\xi}} r_{in}) \leq R_n, \forall i \in \mathscr{I}, \forall t \in \mathscr{T}, \forall \xi = 1,\dots,q \tag{20}$$

$$\sum_{\xi=1}^{q} (1-y^{\xi}) Pr(\boldsymbol{\Gamma} = \epsilon^{\xi}) \leq \hat{\alpha}, \forall i \in \mathscr{I}, \forall \xi = 1,\dots,q \tag{21}$$

$$\sum_{\xi=1}^{q} (1-v^{\xi}) Pr(P_t = P_t^{\xi}) \leq \hat{\alpha}, \forall i \in \mathscr{I}, \forall \xi = 1,\dots,q, \forall t \in \mathscr{T} \tag{22}$$

$$y^{\xi} \in \{0,1\}, \xi = 1,\dots,q \tag{23}$$

$$v^{\xi} \in \{0,1\}, \xi = 1,\dots,q \tag{24}$$

Binary variables y^{ξ} and v^{ξ} represent the satisfaction of precedence relations and resource constraint for realization ξ respectively. Constraints (21) and (22) ensure that the chance of violating either precedence constraint or resource constraint is less than $\hat{\alpha}$.

One common assumption for SAA is that the probability of generating each realization ξ is the uniform, therefore, $Pr(\boldsymbol{\Gamma} = \epsilon^{\xi}) = 1/q$ and $Pr(P_t = P_t^{\xi}) = 1/q$. In this case, constraints (21) and (22) can be replaced by constraints (25) and (26).

$$\sum_{\xi} y^{\xi} \geq \lceil (1-\hat{\alpha}) * q \rceil \tag{25}$$

$$\sum_{\xi} v^{\xi} \geq \lceil (1-\hat{\alpha}) * q \rceil \tag{26}$$

Constraints (25) and (26) ensure that the cumulative probability of realizations that are feasible for constraints (19) and (20) should be

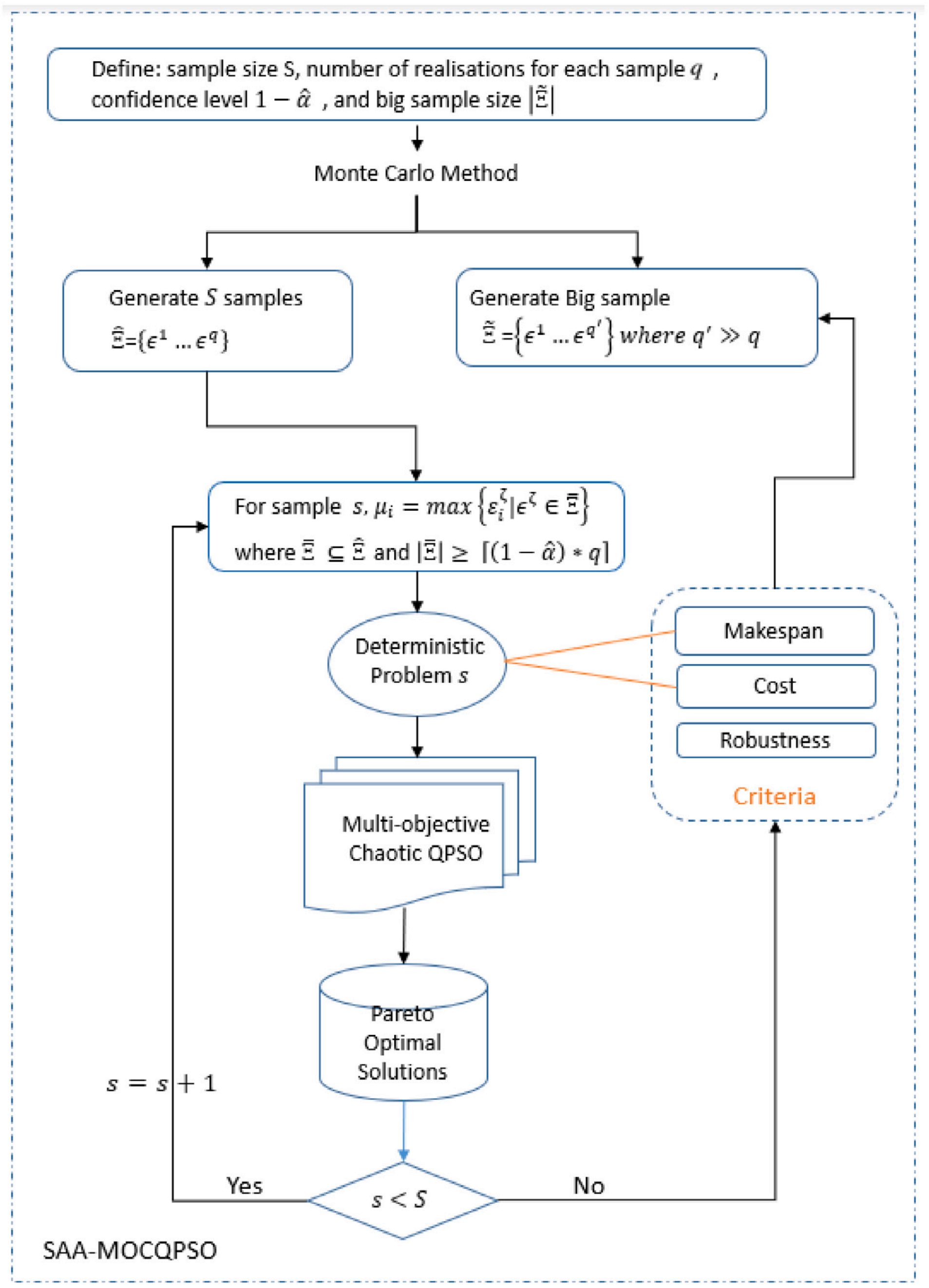
Define: sample size S, number of realisations for each sample q, confidence level $1-\hat{\alpha}$, and big sample size $|\tilde{\Xi}|$
Monte Carlo Method
Generate S samples $\hat{\Xi}=\{\epsilon^1 \dots \epsilon^q\}$
Generate Big sample $\tilde{\Xi}=\{\epsilon^1 \dots \epsilon^{q'}\}$ where $q' \gg q$
For sample s, $\mu_i = max\{\varepsilon_i^\zeta | \epsilon^\zeta \in \bar{\Xi}\}$ where $\bar{\Xi} \subseteq \hat{\Xi}$ and $|\bar{\Xi}| \geq \lceil(1-\hat{\alpha}) * q\rceil$
Deterministic Problem s
Makespan
Cost
Robustness
Criteria
Multi-objective Chaotic QPSO
Pareto Optimal Solutions
$s = s + 1$
Yes
$s < S$
No
SAA-MOCQPSO

Fig. 4. Framework of SAA-MOCQPSO.

greater than $1-\hat{\alpha}$ respectively. For further reformulating this SAA problem, we define a subset $\bar{\Xi} \subseteq \hat{\Xi}$ formed by the randomly sampled realizations. Subset $\bar{\Xi}$ is sufficient if and only if constraints (25) and (26) hold for $\xi \in \bar{\Xi}$ (Lamas & Demeulemeester, 2016). In this case, for sufficient subset $\bar{\Xi}$, constraints (25) and (26) can be reformulated as:

$$|\bar{\Xi}| \geq \lceil(1-\hat{\alpha}) * q\rceil \tag{27}$$

Furthermore, we define μ_i as the maximum duration of set $\bar{\Xi}$ for activity i, the SAA problem is converted to a deterministic problem. Due to the definition of robustness measurements in Section 3.2.2, robustness minimization is not considered in the discrete problems. It will be used as a criterion for evaluating the candidate solutions. The mathematical model of the deterministic problem is presented as follows:

minimize (1), (2) subject to constraints (4)–(6), (9)–(11), (27) and

$$\mu_i = max\left\{\epsilon_i^\xi | \epsilon^\xi \in \bar{\Xi}\right\} \tag{28}$$

$$x_i * x_j * (S_i + \mu_i) \leq S_j, \forall j \in \mathscr{I}, \forall (i,j) \in \mathscr{V} \tag{29}$$

$$(\sum_{i \in P_t^\mu} r_{in}) \leq R_n, \forall i \in \mathscr{I}, \forall t \in \mathscr{T} \tag{30}$$

$$\bar{\Xi} \subset \hat{\Xi} \tag{31}$$

The framework of the proposed hybrid approach is shown in Fig. 4. SAA performs as the outer layer algorithm, which contains three main steps:

1. Generate S samples of scenarios. Each sample $\hat{\Xi}$ contains q realizations, that is $\hat{\Xi} = \{\epsilon^1, \dots, \epsilon^q\}$ where ϵ^i represents a

Algorithm 1: Pseudocode of SGS

```
Input: MSL, PVL, $\mathscr{M}$, $\mathscr{L}$, $\mathscr{O}_m$, $\mathscr{E}$, $\mathscr{H}_l$, $\mathscr{V}$, $\mathscr{N}$
Output: STL, ASL

Step 1: Transforming MSL to feasible ASL
Initialization: MSL = Sigmoid(MSL) & ASL = [0]*len(PVL)
$ASL[l] = 1, l \in \mathscr{E}$
for $m \in \mathscr{M}$ do
    $method_{id} = argmax([MSL[l]\ for\ l \in \mathscr{O}_m])$
    $MSL[method_{id}] = 1$
    $MSL[l] = 0, l \neq method_{id}$ & $l \in \mathscr{O}_m$
    $ASL[l] = 1, l \in \mathscr{H}_{method_{id}}$
end

Step 2: Transforming ASL & PVL to STL
Initialization: STL = [-inf]*len(PVL) & Sequence = [0] &
 Select-num = $count$(ASL=1)
while len(Sequence) < Select-num do
    Candidate-Set = $\{j|(i,j) \in \mathscr{V}, i \in Sequence\}$
    while Candidate-Set do
        ind = $argmin([PVL[j]\ for\ j \in Candidate-Set])$
        t = $max\{s_i + \epsilon_i|(i, ind) \in \mathscr{V}\}$
        resource-constraint = 0
        while resource-constraint < N do
            for $n$ in $\mathscr{N}$ do
                if $Sum(r_{in}) < R_n$ for $i \in P_t$ then
                    resource-constraint += 1
                else
                    t= $min(STL[i] + \epsilon_i)$ for $i \in P_t$
                end
            end
        end
        Sequence = Sequence ∪ {ind}
        Candidate-Set = Candidate-Set \ {ind}
        STL[ind] = t
    end
end
```

realization of a vector of activity durations. Then, generate a big sample $\tilde{\Xi}$ with q' realizations and q' is much larger than q.

2. For each sample s from Step 1, we define the duration of activity i as the maximum duration of $\tilde{\Xi}$ which is randomly sampled from $\hat{\Xi}$ with a confidence level of $1-\hat{\alpha}$, indicated by Eqs. (27) and (28) respectively. The proposed multi-objective chaotic QPSO (MOCQPSO) algorithm is applied to solve the formulated deterministic problem to minimize makespan and total cost. The obtained Pareto optimal solutions for each deterministic problem are stored.
3. If all S sample problems are solved, the Pareto optimal solutions will be evaluated in the big sample $\tilde{\Xi}$. In this step, the expected makespan, expected cost and robustness are used as the criteria for selecting the non-dominated solutions.

4.2. Multi-objective chaotic QPSO algorithm

In classical QPSO, the state of the particle is depicted by a wavefunction $\Psi(v,t)$ where v represents the position of a particle and $|\Psi(v,t)|^2$ equals to the probability density function of particle position at time t (Sun et al., 2004). In quantum mechanics, the wavefunction develops based on the Schrodinger Eq. (32), (33) and Delta potential well Eq. (34):

$$i\hbar\frac{\partial}{\partial t}\Psi(v,t) = \hat{H}\Psi(v,t) \tag{32}$$

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(v) \tag{33}$$

$$V(v) = -\gamma\delta(v-p) = -\gamma\delta(y) \tag{34}$$

where $\hbar$ is the Plank's constant, $\hat{H}$ denotes the Hamiltonian operator and $V(v)$ represents the energy distribution function. Eq. (34) assumes that the particle moves in the potential well with a centre p, a local optimum point.

By solving the Schrodinger equations shown previously (Xu et al., 2017), the probability density function with respect to y is obtained, shown in Eq. (35) with $L = \frac{\hbar^2}{m\gamma}$ represents the length of Delta potential well. With the probability density function $F(y)$, the position v can be obtained through the Monte Carlo simulation as shown in (36) and μ is a random number uniformly distributed within $[0,1]$. By further reformulating this expression (dos Santos Coelho, 2010; Xu et al., 2017), the update procedure for the ith particle's position is expressed in Eq. (37). In Eq. (37), L in (36) is replaced by $2\beta * |Mbest_i(t) - v_i(t)|$ where β is a coefficient that adjusts the convergence speed and r is a random number following a uniform distribution within $[0,1]$. $Mbest(t)$ denotes the average optimal position of all particles in the swarm and $p_i(t)$ is the local attractor that equals to the weighted average of the local optimal position of particle i at time t and the global optimal position of the swarm at time t. The expressions of $Mbest(t)$ and $p_i(t)$ are indicated in Eq. (38) and Eq. (39) respectively where $pbest_i(t)$ denotes the local best position for the ith particle at time t, $gbest(t)$ is the global best position of the swarm at time t, M is the number of particles in the swarm and r_1 is a random number distributed uniformly in $[0,1]$.

$$F(y) = |\Psi(y)|^2 = \frac{1}{L}\exp(-2|y|/L) \tag{35}$$

$$v = p \pm \frac{L}{2}ln(1/\mu) \tag{36}$$

$$\begin{cases} v_i(t+1) = p_i(t) + \beta * |Mbest(t) - v_i(t)| * ln(1/\mu), if \quad r \geq 0.5 \\ v_i(t+1) = p_i(t) - \beta * |Mbest(t) - v_i(t)| * ln(1/\mu), if \quad r < 0.5 \end{cases} \tag{37}$$

$$Mbest(t) = \frac{1}{M}\sum_{i=1}^{M} pbest_i(t) \tag{38}$$

$$p_i(t) = r_1 * pbest_i(t) + (1-r_1) * gbest(t) \tag{39}$$

Although QPSO has been proven to have a better convergence than PSO (Sun et al., 2012, 2011), there are still a few problems when extending to multi-objective QPSO (MOQPSO). Firstly, the learning process of particles mainly relies on their historical optimal positions leading to the decline of diversity in the swarm (Xin-gang et al., 2020). While dealing with multi-objective optimization problems, this could result in fewer non-dominated solutions. Secondly, guided by the historical global optimal particles, there is a risk that the algorithm would be stuck in the local optima and produce sub-optimal solutions (Sun et al., 2012). To overcome these disadvantages, in this section, we introduce the following improvements:

- A two-stage learning strategy for updating particle positions is first proposed
- The chaotic operators including chaotic initialization, chaotic crossover and mutation are introduced.
- The strategy of updating the external archive for storing non-dominated solutions and selecting the global elites are also presented.

The main process of the multi-objective chaotic QPSO (MOCQPSO) is presented in Fig. 5.

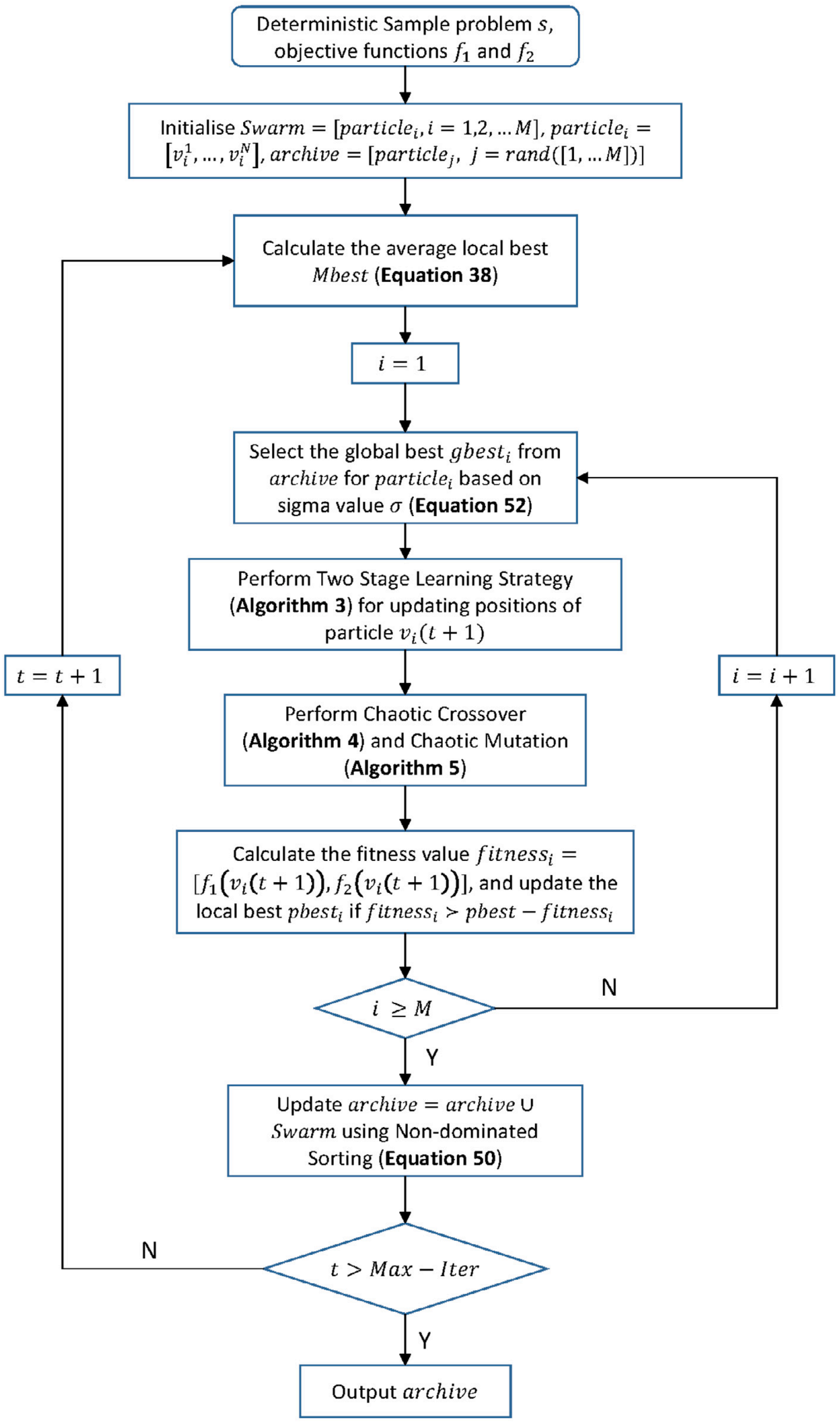
Deterministic Sample problem s, objective functions f_1 and f_2
Initialise $Swarm = [particle_i, i = 1,2, \dots M]$, $particle_i = [v_i^1, \dots, v_i^N]$, $archive = [particle_j,\ j = rand([1, \dots M])]$
Calculate the average local best $Mbest$ (Equation 38)
$i = 1$
Select the global best $gbest_i$ from $archive$ for $particle_i$ based on sigma value σ (Equation 52)
Perform Two Stage Learning Strategy (Algorithm 3) for updating positions of particle $v_i(t+1)$
Perform Chaotic Crossover (Algorithm 4) and Chaotic Mutation (Algorithm 5)
Calculate the fitness value $fitness_i = [f_1(v_i(t+1)), f_2(v_i(t+1))]$, and update the local best $pbest_i$ if $fitness_i \succ pbest - fitness_i$
$i \geq M$
N
Y
$i = i + 1$
Update $archive = archive \cup Swarm$ using Non-dominated Sorting (Equation 50)
$t > Max - Iter$
N
Y
$t = t + 1$
Output $archive$

Fig. 5. The Process of MOCQPSO.

4.2.1. Two-stage learning strategy

The learning strategy of particles in the classical QPSO is solely guided by the average of local bests, i.e. $Mbest$ in Eq. (37). In this section, a two-stage learning strategy that balances exploration and exploitation is proposed. Firstly, a dynamic coefficient β is adopted. During the early stage of learning (i.e. the exploration phase), a larger value of β helps particles explore a wider space. With the increment of iterations, the particles enter the later stage of learning (i.e. the exploitation phase) and exploit the best solutions around their local attractors. Eq. (40) shows the expression of the dynamic β which decrease linearly from β_{max} to β_{min}. Here, t and $Max - Iter$ represent the current iteration and the maximum number of iterations.

$$\beta = \beta_{max} - \frac{\beta_{max} - \beta_{min}}{Max - Iter} * t \tag{40}$$

A threshold value γ of β is predefined to determine whether the particles should explore or exploit. When $\beta \geq \gamma$, the particles are expected to explore rather than exploit. At this stage, to help particles jump out of their current positions, we randomly select a particle from the current population and replace $Mbest(t)$ in Eq. (37) by its current position $v_{rand}(t)$ with a probability of p_l. The learning strategy at the exploration stage can be presented as following:

$$\begin{cases} v_i(t+1) = p_i(t) \pm \beta * |Mbest(t) - v_i(t)| * ln(1/\mu), if \quad rand(\cdot) \geq p_l \\ v_i(t+1) = p_i(t) \pm \beta * |v_{rand}(t) - v_i(t)| * ln(1/\mu), if \quad rand(\cdot) < p_l \end{cases} \tag{41}$$

where $rand(\cdot)$ is a random number within $[0,1]$ and probability p_l can be used to adjust the particles' ability to explore. During the exploiting phase, i.e. $\beta < \gamma$, the particles are expected to search around the best positions to converge towards optima. At this stage, we replace $Mbest$ with a global best position $gbest(t)$ selected from the external archive. The approach of the global best selection for each particle i will be introduced in Section 4.2.5. Eq. (42) shows the expression of the learning strategy at the exploiting phase.

$$v_i(t+1) = p_i(t) \pm \beta * |gbest_i(t) - v_i(t)| * ln(1/\mu) \tag{42}$$

The complete two stage learning strategy is summarized in Algorithm 2.

Algorithm 2: Pseudocode of Two-Stage Learning Strategy

```
Input: Current iteration $t$, Particle current position $v_i(t)$, Local best
       $pbest_i(t)$, Global best $gbest_i(t)$, Average local best $Mbest(t)$,
       Random particle $v_{rand}(t)$, Exploration probability $p_l$, $\beta$ value
       range $[\beta_{min}, \beta_{max}]$, threshold $\gamma$
Output: Particle position at iteration $t+1$, $v_i(t+1)$

Update the current $\beta$ based on Eq. (40)
Generate a random factor $\mu \in [0,1]$
if $\beta \geq \gamma$ then
    if $rand(\cdot) \geq p_l$ then
        $v_i(t+1) = p_i(t) \pm \beta * |Mbest(t) - v_i(t)| * ln(1/\mu)$
    else
        $v_i(t+1) = p_i(t) \pm \beta * |v_{rand}(t) - v_i(t)| * ln(1/\mu)$
    end
else
    Update $v_i(t+1)$ based on Eq. (42)
end
```

4.2.2. Chaotic initialization

The initialization of population is vital in evolutionary algorithms that might influence the convergence rate and the solution quality (Lashkari & Moattar, 2017). Traditionally, the initialization of swarm or population in PSO and its variants is randomly generated following a uniform distribution. Lu et al. (2014) demonstrated the good effects of chaotic initialization on the performance of multi-objective evolutionary algorithms which helps avoid solutions falling into local optimum. Therefore, in this paper, chaotic numbers are used for initialization to improve the diversity of search space. One of the most studied chaotic mappings is the logistic map and it is presented by Eq. (43) where r is a system parameter and when $r = 4$, the generated sequence is chaotic (Kanso & Smaoui, 2009). In (43), $\omega(n) \in (0,1)$ and $\omega(0) \notin \{0.25, 0.5, 0.75\}$, and n is the number of iterations. In this case, the $(n+1)^{th}$ dimension of the particle position is initialized as shown in Eq. (44) where v_{min} and v_{max} are the lower bound and upper bound of particle position values respectively.

$$\omega(n+1) = r * \omega(n)(1 - \omega(n)) \tag{43}$$

$$v_i^{n+1} = v_{min} + (v_{max} - v_{min})\omega(n+1) \tag{44}$$

4.2.3. Chaotic crossover

The updates of the swarm in the original QPSO algorithm greatly rely on the guidance of historical global best particles selected from the external archive, which may likely result in the loss of diversity in the later stage of evolution (Feng et al., 2017). Based on the chaotic methods proven to be effective in evolutionary algorithms presented by Lu et al. (2011) and Feng et al. (2017), a chaotic crossover operator is introduced. Two parent particles are the non-dominated individuals $gbest_1$ and $gbest_2$ randomly selected from the external archive while two children particles $child_1$ and $child_2$ are chosen randomly from the current swarm. For any $particle_i$ in the current swarm, a new candidate particle is generated by Eq. (45) where $\hat{v_i^n}$ is the value of nth dimension for the new candidate particle. η_c is the logistic chaotic mapping of ρ_c based on Eq. (43). Therefore, we have $\eta_c = r * \rho_c * (1 - \rho_c)$.

$$\hat{v_i^n} = v_i^n + \rho_c * (gbest_1^n - child_1^n) + \eta_c * (gbest_2^n - child_2^n) \tag{45}$$

If this new candidate particle dominates the original $particle_i$, $particle_i$ is replaced by this newly generated particle.

Algorithm 3: Pseudocode of Chaotic Crossover

```
Input: Chaotic crossover probability $p_c$, Particle current position
       $v_i(t)$, Global best particles $gbest_1$ and $gbest_2$, Children
       particles $child_1$ and $child_2$, Chaotic system parameter $r$
Output: Particle position at iteration $t+1$, $v_i(t+1)$

for $n = 1$ to $N$ do
    Initialize $\rho_c = rand(\cdot)$ and $\rho_c \notin \{0.25, 0.5, 0.75\}$
    $\eta_c = r * \rho_c * (1 - \rho_c)$
    Conduct crossover operation and obtain new position $\hat{v_i^n}$ based
      on Eq. (45)
end
if $f(\hat{v}_i) \succ f(v_i(t))$ then
    Update the position $v_i(t+1) = \hat{v}_i$
else
    $v_i(t+1) = v_i(t)$, pass
end
```

4.2.4. Chaotic mutation

Inspired by Li et al. (2015), a chaotic mutation operation is applied for diversifying the swarm subject to a particular mutation probability. The chaotic mutation operation contains two parts. Firstly, if the value of the nth dimension of a particle's position, v_i^n, exceeds the limit of $[v_{min}, v_{max}]$, Eq. (46) and (47) are applied to regulate and mutate v_i^n:

$$\rho_m = \frac{v_i^n - v_{min}}{v_{max} - v_{min}} \tag{46}$$

$$\hat{v_i^n} = \eta_m(v_{max} - v_{min}) + v_{min} \tag{47}$$

where η_m is the logistic mapping of ρ_m. Otherwise, Eq. (48) is utilized for mutation.

$$\hat{v_i^n} = \eta_m(gbest_i^n - v_i^n) + v_{rand}^n \tag{48}$$

where $gbest_i$ is selected from the external archive through the approach introduced in Section 4.2.5 and v_{rand} is selected randomly from the current population to enhance the randomness. The original $particle_i$ will be replaced with the new candidate particle generated through a chaotic mutation operator with a probability p_m.

4.2.5. Archive update and elite selection

After each iteration of swarm updating, the personal best $pbest_i$ will be replaced by the newly generated $particle_i$ if the new one dominates the previous particle or they are non-dominated to each other. For storing the non-dominated solutions obtained in each iteration, an external archive is introduced. At each iteration, personal best particles are added into the archive. A non-dominated sorting procedure proposed

Algorithm 4: Pseudocode of Chaotic Mutation

```
Input: Chaotic mutation probability $p_m$, Particle current position
       $v_i(t)$, Global best $gbest_i$, Random particle $v_{rand}(t)$, Position
       range $[v_{min}, v_{max}]$, Chaotic system parameter $r$
Output: Particle position at iteration $t+1$, $v_i(t+1)$

if $rand(\cdot) \leq p_m$ then
    for $n = 1$ to $N$ do
        Calculate $\rho_m$ based on Eq. (46)
        $\eta_m = r * \rho_m * (1 - \rho_m)$
        if $v_i^n(t) > v_{max}$ or $v_i^n(t) < v_{min}$ then
            Update the position $v_i^n(t+1) = \hat{v_i^n}$ based on Eq. (47)
        else
            Update the position $v_i^n(t+1) = \hat{v_i^n}$ based on Eq. (48)
        end
    end
else
    $v_i(t+1) = v_i(t)$, pass
end
```

by Ghoddousi et al. (2013) is conducted to reserve the non-dominated solutions and exclude any dominated ones. As the length of the archive will increase continuously, a maximum length of archive is applied to reduce the computational burden. Crowding distance (CD) is used as a criterion for excluding excessive non-dominated solutions. CD is a measure of the density of the surrounding solutions (Chu & Yu, 2018), which can be calculated as the Euclidean distance between neighbour individuals as indicated in Eq. (49) where f_{j+1}^k and f_{j-1}^k represents the kth objective value of two closet neighbour solutions of individual j and f_{max}^k and f_{min}^k are the maximum and minimum values of the kth objective function respectively.

$$CD_j = \sum_{k=1}^{K} \frac{f_{j+1}^k - f_{j-1}^k}{f_{max}^k - f_{min}^k} \tag{49}$$

Global best solution $gbest$ selected from the external archive performs as a guide for swarm updating. Therefore, the elite selection strategy has a significant impact on convergence. Instead of using CD as the selection strategy, the Sigma method is implemented due to its good performance on convergence (Moslehi & Mahnam, 2011; Yang et al., 2009). Each non-dominated solution in the external archive is assigned with a sigma value σ_i and the general expression of σ_i is shown in Eq. (50). In Eq. (50), $vector(f_{i,j}^2 - f_{i,k}^2)$ represents the vector with $\binom{N}{2}$ formed by the value differences between the square of any two objective values (i,k) of solution i. In our case of resolving the SAA problem described in Section 4.1, two objective functions are considered at this stage and the σ_i for two objective values is a single value as indicated in Eq. (51).

$$\sigma_i = \frac{vector(f_{i,j}^2 - f_{i,k}^2)}{\sum_{k=1}^{K} f_{i,k}^2},\ i,k \in \{1,2,\dots,K\} \text{ and } i \neq k \tag{50}$$

$$\sigma_i = \frac{f_{i,1}^2 - f_{i,2}^2}{f_{i,1}^2 + f_{i,2}^2} \tag{51}$$

Instead of only selecting one global best $gbest$, the best local guide for each $particle_i$ is chosen based on the distance of sigma values. The Euclidean distances between σ_i of $particle_i$ in the swarm and the sigma values of all particles in the archive are calculated. If σ_j of$particle_j$ from the archive has the shortest distance to σ_i, then $particle_j$ is selected as the local best $gbest_i$ for $particle_i$ in Eq. (39). It is worth mentioning that the values of each objective function are normalized to the range of [0,1] before calculating σ_i to avoid that σ_i is greatly affected by the objectives with large numerical values.

5. Experiments and numerical analysis

In this section, we study the performance of the proposed hybrid algorithm through three experiments. Firstly, the MOCQPSO algorithm is evaluated on a set of diverse benchmark functions. Secondly, some experimental instances are generated randomly based on a well-known Project Scheduling Problem Library (PSPLIB) (Kolisch & Sprecher, 1997) to validate the effectiveness of our proposed hybrid algorithm. For both experiments, the proposed algorithm is compared with the MOQPSO algorithm (Feng et al., 2017), the MOPSO algorithm (Moslehi & Mahnam, 2011) and the NSGA-II (Laszczyk & Myszkowski, 2019) to demonstrate its superior performance. Lastly, we conduct a sensitivity analysis to assess the effects of the main parameters of the proposed algorithm. All the algorithms involved in this section are compiled in Python 3.7 running on a computer with Intel Xeon $E5-1650$ v4, 3.6 GHz CPU.

5.1. Experimental design

5.1.1. Benchmark functions

To verify the performance of the MOCQPSO algorithm in dealing with unconstrained multi-objective problems, several benchmark functions are selected from standard test suits, namely ZDT (Zitzler et al., 2000) and DTLZ (Deb et al., 2005). Specifically, we select ZDT1, ZDT2 and ZDT3 functions from the ZDT test suit, which present Pareto fronts that are convex, concave and disconnected, respectively. The ZDT test suit problems, except for ZDT5, have two objectives and 30 variables with the range of $[0,1]$. To consider problems with more objectives, we also conduct testing on DTLZ2, DTLZ4 and DTLZ5 problems. DTLZ2 measures the algorithm's ability to scale up its performance in a large number of objectives, DTLZ4 introduces a bias parameter to investigate the algorithm's ability to maintain a good distribution of solutions, and DTLZ5 tests the ability of an algorithm to converge to a curve (Deb et al., 2005). Meneghini et al. (2020) indicated that the number of decision variables for DTLZ problems is set as $d = m + k - 1$, where m denotes the number of objectives and k is the fixed parameter. In the experiment, we set $m = 3, 5$, $k = 10$ for DTLZ problems and the bias parameter as 100 for DTLZ4, specifically.

5.1.2. MSRCPSP-A instance generation

In this section, an instance generator for the MSRCPSP-A problem is developed and introduced. This generator creates experimental instances based on the RCPSP dataset of PSPLIB determining the precedence relations, deterministic durations, resource demands for activities respectively. For the stochastic durations, an uncertainty ratio θ is introduced and the duration of activity i varies in the range $[\ \lfloor (1-\theta) * \epsilon_i \rfloor, \lceil (1+\theta) * \epsilon_i \rceil\ ]$. In our experiments, four instance groups from the RCPSP dataset, J30 (i.e. instances with 30 activities), J60, J90 and J120, are adopted. The activities from instance groups define the characteristics of mandatory activities in the projects while the alternative execution methods with the corresponding optional activities are generated separately.

The flowchart of the proposed instance generator is shown in Fig. 6. For example, in the J30 instance group, the procedure of generating a project instance is described as follows. First of all, the number of tasks (n_t), the range of number of methods for each task ($range_m$), the range of number of activities for each method ($range_a$), the number of resources (n_r) and the capacity of resource r ($capacity_r$). Then, we generate a big group of candidate J30 RCPSP instances and a big group of candidate alternative methods with the number of activities within $range_a$ utilizing the ProGen data generator from PSPLIB. The next step is to select an instance and insert the optional activities of alternative methods. For example, in a selected J30 instance, an arc from the AON network is selected and a randomly selected alternative method is inserted between the two nodes of the arc. As we can observe from Fig. 6, arc $2-6$ is selected and two alternative methods

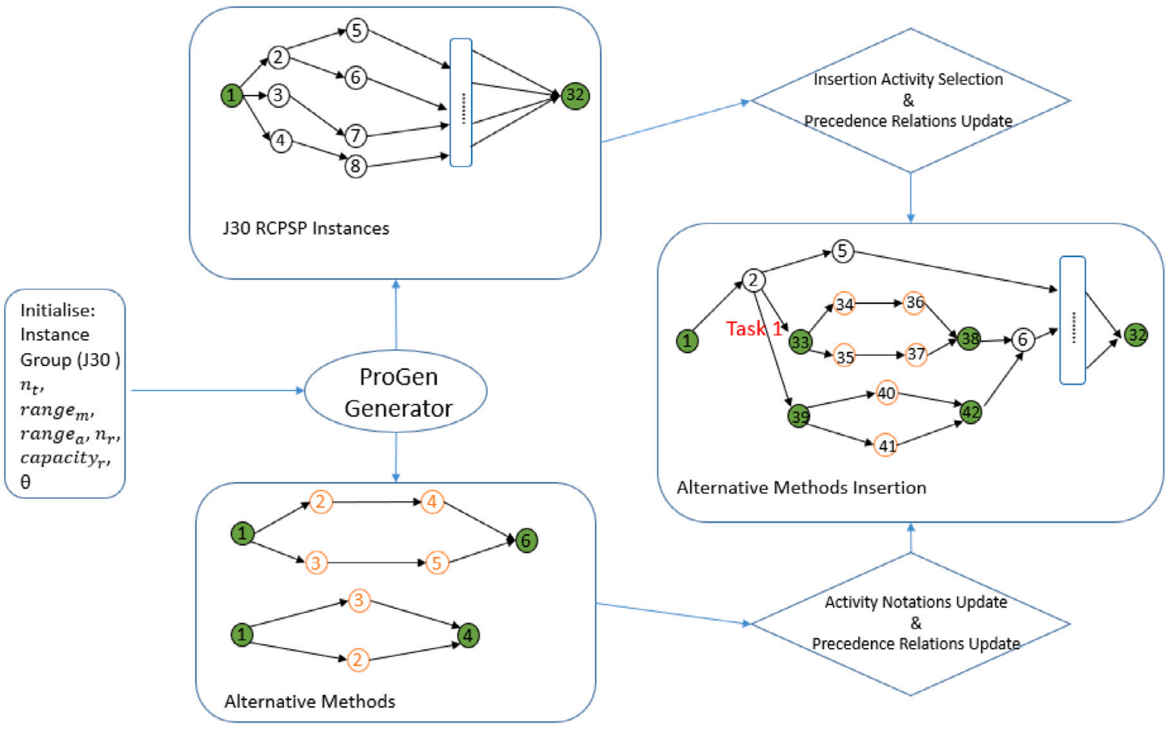
J30 RCPSP Instances
Insertion Activity Selection & Precedence Relations Update
Alternative Methods Insertion
Task 1
Initialise: Instance Group (J30) n_t, $range_m$, $range_a$, n_r, $capacity_r$, θ
ProGen Generator
Alternative Methods
Activity Notations Update & Precedence Relations Update

Fig. 6. Flowchart of Instance Generation.

Table 3
Experimental project instances.

Instance group	Instance no.	Activities	Tasks	Methods	Resource cost	Resource capacity
J30	1	93	4	9	[5, 8, 14, 15]	[10, 10, 14, 14]
	2	87	4	9	[5, 11, 10, 8]	[25, 26, 18, 23]
	3	106	4	11	[8, 9, 8, 12]	[17, 15, 17, 17]
J60	1	181	6	17	[10, 13, 15, 9]	[14, 15, 14, 16]
	2	166	6	16	[8, 8, 12, 13]	[16, 16, 17, 26]
	3	184	6	18	[9, 7, 14, 6]	[18, 17, 28, 17]
J90	1	224	8	19	[9, 14, 5, 8]	[17, 20, 20, 23]
	2	234	8	22	[7, 8, 9 , 14]	[21, 20, 20, 23]
	3	234	8	18	[7, 11, 9, 7]	[20, 23, 15, 32]
J120	1	302	10	27	[5, 14, 14, 14]	[17, 14, 14, 14]
	2	269	10	22	[12, 12, 6, 12]	[27, 24, 26, 23]
	3	267	10	22	[6, 8, 6, 11]	[58, 62, 60, 60]

Table 4
Parameters setting of algorithms.

SAA	Sample Size S: 30 Big Sample Size q': 1000	No. of Realizations q: 50 Robustness Coefficient λ: 1	Confidence Level $1-\alpha$: 0.95
MOCQPSO	β Range: $[0.4, 1]$	Chaotic System Parameter r: 4	Threshold γ: 0.6
MOQPSO (Feng et al., 2017)	β Range: $[0.4, 1]$		
MOPSO (Moslehi & Mahnam, 2011)	Inertial Weight w: 0.5	Cognitive Coefficient c_1: 2	Social Coefficient c_2: 2
NSGA-II (Laszczyk & Myszkowski, 2019)	Crossover Probability p_{cross}: 0.9	Mutation Probability p_{mutate}: 0.2	

for Task 1 are added in between. The green nodes represent dummy nodes that have zero durations. In this case, the precedence relations and notations of activities for execution methods are updated. Until alternative execution methods for all tasks are inserted into the original network, a project instance for the proposed MSRCPSP-A problem is generated.

In this section, 12 project instances are generated based on J30, J60, J90 and J120. The n_t varies within the range of $[4, 10]$ with $range_m = [2, 4]$ and $range_a = [3, 6]$. Four types of resources are considered in our experiments and the original RCPSP instances define the capacity of each resource. Table 3 shows the configuration of each generated instance for our experiments.

5.1.3. *Performance evaluation metrics*

For evaluating the performance of a stochastic multi-objective optimization algorithm, the evaluation metrics should provide indications from several aspects (Goh et al., 2010): (1) the stability of obtained solutions under stochastic settings which the robustness of solutions can represent; (2) the quality of the Pareto optimal solutions compared to other benchmark algorithms; (3) the convergence that the distance can represent between the Pareto set produced by the algorithm and the reference Pareto set; (4) the diversity represented by the space that is covered by the solutions. Apart from average objective values for non-dominated solutions, three evaluation metrics are adopted in this

Table 5
Performance comparison on benchmark functions.

Functions	MOCQPSO			MOQPSO			MOPSO			NSGA-II		
	GD	SM	No. Solu	GD	SM	No. Solu	GD	SM	No. Solu	GD	SM	No. Solu
ZDT1	**0.0616**	**0.0486**	13.20	0.6364	0.0945	6.60	0	–	1.00	0.1783	0.0679	**26.4**
ZDT2	0.1228	**0.0853**	3.60	**0.0956**	0.0943	4.20	0	–	1.00	0.2657	0.0968	**5.20**
ZDT3	**0.0226**	**0.0121**	8.20	0.0611	0.0422	5.50	0	–	1.00	0.1168	0.0674	**25.10**
DTLZ2–3	1.1457	0.2173	23.20	1.5122	0.2246	3.80	1.3628	–	1.00	**0.1127**	**0.0509**	**122.60**
DTLZ4–3	0.9115	**0.0452**	5.20	1.4070	0.0973	1.70	0.8301	–	1.00	**0.1121**	0.0498	**109.20**
DTLZ5–3	**1.1377**	**0.7483**	22.80	1.2037	0.9179	10.20	1.4565	0.7553	1.60	1.2352	0.7864	**121.80**
DTLZ2–5	1.3256	**0.2768**	44.20	1.8667	0.4312	13.80	1.4872	–	1.00	**0.7968**	0.3288	**157.80**
DTLZ4–5	1.0245	**0.2239**	13.40	1.5781	0.2613	7.10	1.3704	–	1.00	**0.4636**	0.3286	**132.20**
DTLZ5–5	**0.7389**	**0.3248**	41.20	0.8552	0.5105	30.80	0.7507	0.3885	1.90	1.0941	0.5492	**128.30**

paper (Gomes et al., 2014; Khalili-Damghani et al., 2013; Lin et al., 2015; Xiong et al., 2012).

Generational distance (GD). The GD measures the distance between the solutions obtained by an algorithm and the reference set. Assuming P_i is the Pareto set of non-dominated solutions obtained by algorithm $i = 1, \dots, I$, and $Ref = P_1 \cup P_2 \dots \cup P_I$ denotes the reference set. The formula of GD can be represented as follows.

$$GD(P_i, Ref) = \frac{\sum_{j=1}^{|P_i|} d_j}{|P_i|} \tag{52}$$

$$d_j = min(\sqrt{\sum_{m=1}^{M} (\frac{f_j^m - \hat{f}_k^m}{\hat{f}_{max}^m - \hat{f}_{min}^m})^2} \mid k \in Ref) \tag{53}$$

where $|P_i|$ means the number of solutions in set P_i and d_j is the normalized minimum Euclidean distance from the jth solution in P_i to the solutions in Ref. In (53), f_j^m and $\hat{f}_k^m$ represents the mth objective value for the jth solution in P_i and kth solution from Ref. $\hat{f}_{max}^m$ and $\hat{f}_{min}^m$ are the maximum and minimum values of the mth objective for Ref. The lower value of GD indicates a smaller deviation towards the reference set which is desirable.

Set cover (SC). The metric SC(A,B) proposed by Xiong et al. (2012) measures the dominance relationship between set A and set B. In this section, SC(A,B) defines the ratio of non-dominated solutions from Pareto set B dominated by the solutions from A. The higher value of SC(A,B) indicates a better performance of Pareto set A. Based on the definition of SC, we use the contribution to the reference set Ref for each algorithm to represent its dominant relationship with the other two algorithms. The contribution to the reference set for each algorithm is denoted by $\%Ref$.

Spacing metric (SM). SM measures the uniformly spread of solutions in a Pareto set. The formula of SM is shown as follows. In (54), d_j is the normalized minimum distance of jth solution to other solutions in P_i and $\bar{d}$ represents the average distance of all d_j. According to Khalili-Damghani et al. (2013), the smaller value of SM indicates more uniformly distributed of the algorithm and is more preferable.

$$SM(P_i) = \sqrt{\frac{\sum_{j=1}^{|P_i|} (\bar{d} - d_j)^2}{|P_i| - 1}} \tag{54}$$

Infeasibility rate (inf). When evaluating the non-dominated solutions in a large sample $\tilde{\Xi}$ in Step 3 of SAA, candidate solutions might be infeasible for some realization instances. The infeasibility implies that the schedules decoded from the solutions violate either precedence constraints or resource constraints. To evaluate the feasibility of a solution, the infeasibility rate (Inf) is introduced. The mathematical expression of Inf is shown in Eq. (55), where $\tau^{\xi} = 1$ representing that the solution is not feasible for realization $\xi \in \tilde{\Xi}$, otherwise $\tau^{\xi} = 0$. τ^{ξ} is a combination of y^{ξ} and v^{ξ} in Eq. (25) and (26).

$$Inf = \frac{\sum_{\xi \in \tilde{\Xi}} \tau^{\xi}}{q'} \tag{55}$$

5.1.4. Experiments setting

In the first experiment, we aim to verify the performance of the proposed MOCQPSO in dealing with deterministic multi-objective problems. We compare the results of GD, SM and the average number of Pareto solutions for the MOCQPSO, the MOQPSO, the MOPSO and the NSGA-II on benchmark functions. Pareto-optimal fronts of the benchmark functions provided by Zitzler et al. (2000) and Deb et al. (2005) are used as the reference set Ref in GD. A comprehensive analysis is conducted for the second experiment by implementing hybrid algorithms to solve the randomly generated MSRCPSP-A instances. To keep the fairness of comparison, we apply the proposed SAA algorithm with all multi-objective algorithms, i.e. SAA-MOCQPSO, SAA-MOQPSO, SAA-MOPSO and SAA-NSGA-II. The comparisons consist of three groups according to different criteria and purposes: (1) comparison of average objective values; (2) comparison of Pareto fronts; (3) comparison under different uncertainty levels. The first two comparisons are conducted under the same uncertainty ratio θ, which is set as 0.3 in our experiment. For the third group of comparison, one instance is selected from each instance group for evaluating the performance of algorithms under different uncertainty ratio θ. The parameters of these algorithms are set according to the previous studies. For all algorithms involved in the first two experiments, we set the size of population $M = 100$ and maximum iterations $Max - Iter = 100$. For PSO-based algorithms, i.e. MOCQPSO, MOQPSO and MOPSO, the maximum length of archive $Max - Arch = 1000$. The unique parameters for each algorithm and the SAA parameters shared by all algorithms are presented in Table 4. For each benchmark function and MSRCPSP-A instance, every algorithm is performed 10 times independently and the results are averaged.

5.2. Experimental results

5.2.1. Performance comparison on benchmark functions

The numerical results of the algorithms' comparison on benchmark functions from ZDT and DTLZ test suits are reported in Table 5. In general, the MOCQPSO has achieved a competitive performance compared to other algorithms. According to the results, the GD values of the MOCQPSO are the best on 4 out of 9 benchmark problems and the corresponding SM values achieve the best on 8 problems.

The MOCQPSO algorithm has a dominated performance compared to its original variants, i.e. the MOQPSO and the MOPSO. For the ZDT functions, the MOPSO finds only one optimal non-dominated solution for every time of running. Therefore, the GD values of the MOPSO are zeros and the SM values are not available. The MOCQPSO has the best performance on GD and SM values compared to the MOQPSO, expect for the ZDT2 function. As we can observe, the proposed MOCQPSO outperforms the MOQPSO and the MOPSO on all metrics for both 3-objective and 5-objective DTLZ functions. A significant improvement in the average number of the non-dominated solutions for the MOCQPSO can be observed, which indicates the effectiveness of the two-stage learning strategy in encouraging exploration. Additionally, the lower

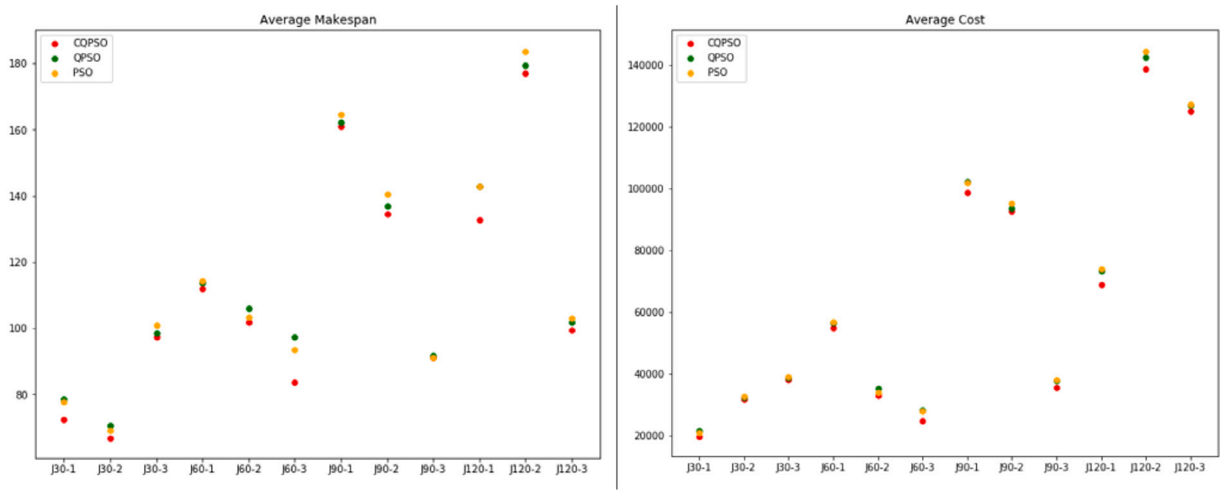

Fig. 7. Scatter Plot of Average Makespan and Average Cost.

Table 6
Performance comparison with PSO-based algorithms: average objectives.

Instances	MOCQPSO			MOQPSO			MOPSO		
	Makespan	Cost	Robust	Makespan	Cost	Robust	Makespan	Cost	Robust
J30–1	**72.25**	**19637.25**	**0.1013**	77.59	21563.07	0.1123	77.31	21007.42	0.1022
J30–2	**67.03**	**31885.04**	0.1127	71.72	32507.31	**0.0937**	69.65	32809.53	0.1133
J30–3	**92.49**	**38005.07**	**0.0737**	98.64	38788.53	0.0839	100.76	38957.67	0.0807
J60–1	**107.63**	**53902.41**	**0.0805**	112.69	56631.07	0.0893	115.77	56931.51	0.0863
J60–2	**101.25**	**32906.07**	**0.0717**	106.73	34983.36	0.0796	104.91	34225.03	0.0792
J60–3	**84.06**	**24399.24**	0.0713	96.37	28359.17	**0.0704**	94.65	27909.45	0.0750
J90–1	**157.85**	**97637.07**	**0.0518**	163.07	102431.47	0.0581	165.25	101770.32	0.0554
J90–2	**131.13**	**89705.75**	**0.0567**	137.09	93597.06	0.0585	141.1	95424.12	0.0571
J90–3	**89.54**	**35757.43**	**0.0807**	92.63	37687.58	0.0833	91.83	37886.25	0.0836
J120–1	**133.85**	**68748.38**	**0.0411**	142.69	73169.16	0.0455	143.23	73831.38	0.0447
J120–2	**172.97**	**136847.29**	0.0525	179.98	142769.61	0.0539	182.54	143709.69	**0.0492**
J120–3	**98.58**	**125283.25**	**0.0629**	101.37	128035.34	0.0665	102.73	127876.05	0.0633

GD and SM values for the MOCQPSO indicates that it could obtain the solutions with better optimality and distribution.

NSGA-II has shown a significant dominance in its ability of exploration. It could obtain much more non-dominated solutions compared to PSO-based algorithms. However, the MOCQPSO outperforms the NSGA-II on SM values for 8 of 9 benchmark functions. For GD values, we can observe that the performance of the MOCQPSO on 2-objective ZDT functions is significantly better than the NSGA-II. By contrast, NSGA-II achieves lower GD values on DTLZ2 and DTLZ4 functions, which indicates that NSGA-II has a better ability to with large numbers of objectives. It is worth mentioning that the performance of the MOCQPSO is similar to the NSGA-II on DTLZ4-3. The MOCQPSO shows better performance on both DTLZ5-3 and DTLZ5-5 than the NSGA-II, which implies its good ability to converge to a curve.

In a nutshell, we can see that the proposed MOCQPSO algorithm has shown a significant improvement on the convergence and diversity of solutions compared to the other two PSO-based algorithms. Even though the NSGA-II is better performed on the average number of solutions, the MOCQPSO still outperforms the GD and SM values on 5 and 8 benchmark functions, respectively. Therefore, the proposed MOCQPSO algorithm is competitive in dealing with multi-objective problems.

5.2.2. *Performance comparison on MSRCPSP-a*

In this section, the numerical results for the MSRCPSP-A problem are reported. Firstly, the comparisons between MOCQPSO and its original variants (i.e. MOQPSO, MOPSO) in terms of average objective values and evaluation metrics are presented in Tables 6 and 7. Secondly, instances J30-1, J60-2 and J90-1 are selected to validate the performance of algorithms for dealing with different levels of uncertainty, and the results are indicated in Table 8. Thirdly, we evaluate the performance of MOCQPSO by comparing with NSGA-II, and the results are shown in Table 9.

From Table 6, it can be seen clearly that the proposed MOCQPSO algorithm could obtain better results than the original MOQPSO and MOPSO on objective values, especially for average makespan and average cost. As we can observe from Fig. 7, both the average makespan and average cost for MOCQPSO are lower than the other two algorithms for all instances. For example, the average makespan and cost obtained by MOCQPSO for instance J30-2 are 67.03 and 31885.04 respectively, which are lower than 71.72 and 32507.31 for MOQPSO and 69.65 and 32809.53 for MOPSO. For instances with a large structure, such as J120-1 with 302 activities and 27 alternative methods, MOCQPSO also outperforms the other two algorithms with 9.38 units shorter in average makespan and 6862.4 lower in average cost compared to MOPSO. MOCQPSO is better performed for most instances for robustness measurement, except for instance J30-2, J60-3 and J120-2. However, the gaps in robustness for these three algorithms are tiny.

From Table 7, we can conclude that the solutions of MOCQPSO dominates most of the Pareto optimal solutions obtained by MOQPSO and MOPSO. MOCQPSO has the highest percentage of contribution to the reference sets ($\%Ref$) for all instances indicating that the quality of non-dominated solutions for MOCQPSO is much better than the others. MOCQPSO contributes to 88.57% and 88.09% of the reference sets for J30-2 and J120-2 while the lowest $\%Ref$ it ever achieves is

Table 7
Performance comparison with PSO-based algorithms: evaluation metrics.

Instances	MOCQPSO				MOQPSO				MOPSO			
	GD	SM	No. Solu	% Ref	GD	SM	No. Solu	% Ref	GD	SM	No. Solu	% Ref
J30–1	0.0013	0.1213	17.30	75.62	0.3693	0.1431	12.10	6.21	0.2224	0.1884	7.30	18.17
J30–2	0.0015	0.1063	28.40	88.57	0.0895	0.1108	12.70	9.33	0.1615	0.0883	21.60	2.10
J30–3	0.1092	0.1457	18.10	87.08	0.7326	0.1803	8.40	8.63	1.0035	0.1974	10.70	4.29
J60–1	0.0065	0.0835	29.50	68.37	0.1036	0.1169	19.10	12.97	0.0787	0.1326	20.2	18.66
J60–2	0.0096	0.0995	18.50	64.34	0.1553	0.1032	18.10	12.03	0.0762	0.1059	17.70	23.67
J60–3	0	0.1079	9.50	83.35	1.0136	0.1132	8.70	2.37	0.9044	0.1455	10.60	14.28
J90–1	0.0012	0.0752	26.30	83.84	0.2954	0.1034	19.80	7.89	0.2039	0.1165	21.40	8.72
J90–2	0.0384	0.0924	14.10	55.46	0.0693	0.1197	13.60	42.18	0.2092	0.1772	14.60	2.36
J90–3	0	0.0871	13.70	61.91	0.1831	0.0854	16.90	17.05	0.1895	0.0977	19.10	21.04
J120–1	0.0008	0.0597	15.60	68.54	0.1637	0.1354	14.40	13.33	0.1369	0.1839	9.90	18.13
J120–2	0.0015	0.0749	28.30	88.09	0.1748	0.1245	17.80	8.76	0.2174	0.1032	22.90	3.15
J120–3	0.0137	0.1044	18.40	72.55	0.1404	0.0948	17.90	20.71	0.1336	0.0847	16.90	6.74

Table 8
Performance comparison with different uncertainty ratio.

Instances	Uncertainty	MOCQPSO				MOQPSO			
		Dev.Makespan	Dev.Cost	Avg.Robust	Avg.Inf	Dev.Makespan	Dev.Cost	Avg.Robust	Avg.Inf
	0.3	0.0769	0.0086	0.1013	–	0.0731	0.0293	0.1123	–
J30-1	0.5	0.0840	0.0185	0.1396	0.0149	0.0904	0.0533	0.1562	0.0173
	0.8	0.0897	0.0107	0.2114	0.0195	0.0945	0.0697	0.2335	0.0233
	0.3	0.0464	0.0339	0.0718	0.0074	0.0533	0.0494	0.0796	0.0088
J60-2	0.5	0.0493	0.0367	0.1037	0.0147	0.0455	0.0514	0.1169	0.0225
	0.8	0.0484	0.0313	0.1522	0.0471	0.0502	0.0447	0.1767	0.0531
	0.3	0.0283	0.0126	0.0518	–	0.0317	0.0215	0.0581	–
J90-1	0.5	0.0468	0.0195	0.0674	0.0178	0.0501	0.0218	0.0716	0.0225
	0.8	0.0363	0.0279	0.0843	0.0281	0.0287	0.0205	0.1145	0.0441

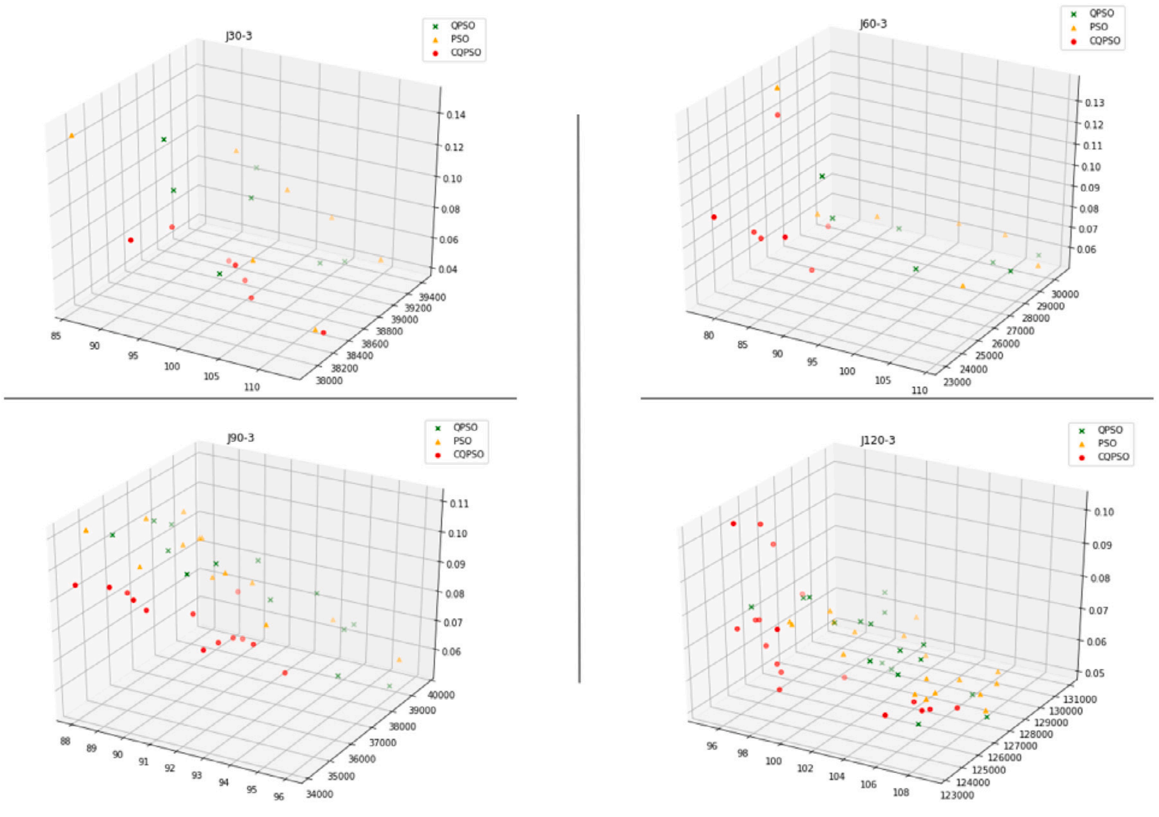

Fig. 8. Non-dominated Solutions.

55.46% for J90-2. On the contrary, the highest $\%Ref$ for MOQPSO and MOPSO are only 42.18% and 23.67% respectively. Fig. 8 presents the distribution of non-dominated solutions of three algorithms for J30-3, J60-3, J90-3 and J120-3 instances. The solutions that locate around the lower-left corner of the coordinate system are better performed. It is evident that solutions of MOCQPSO (i.e. red dots) outperform

Table 9
Performance comparison with NSGA-II.

Instances	NSGA-II				GAP (w. MOCQPSO)				SC (%)
	Makespan	Cost	Robust	SM	Makespan	Cost	Robust	SM	
J30–1	73.31	19572.01	0.1173	0.1306	−1.06	65.24	−0.0135	−0.0093	63.64
J30–2	67.47	31879.09	0.1044	0.0790	−0.44	5.95	0.0028	0.0273	60.45
J30–3	98.75	38264.31	0.0786	0.1519	−6.26	−259.24	−0.0025	−0.0062	62.51
J60–1	112.21	54646.32	0.0856	0.1716	−4.58	−743.91	−0.0014	−0.0881	70.59
J60–2	102.27	32721.13	0.0927	0.2536	−1.02	184.94	−0.0195	−0.1541	52.86
J60–3	85.17	24849.4	0.0889	0.1065	−1.11	−450.16	−0.0095	0.0014	57.14
J90–1	161.63	98722.63	0.0528	0.1058	−3.78	−1085.56	−0.0006	−0.0306	65.38
J90–2	136.02	92300.82	0.0599	0.1529	−4.89	−2595.07	−0.0012	−0.0605	52.33
J90–3	91.1	36621.37	0.0821	0.1258	−1.56	−863.94	−0.0002	−0.0387	43.24
J120–1	133.82	69972.58	0.0554	0.097	0.03	−1224.2	−0.0149	−0.0373	76.19
J120–2	179.18	138631.09	0.0568	0.0755	−6.21	−1783.8	−0.0039	−0.0006	73.57
J120–3	101.71	126865.43	0.0717	0.0786	−3.13	−1582.18	−0.0009	0.0258	50.46

the others. In terms of GD, MOCQPSO has zero values in 6 instances which means the solutions obtained by MOCQPSO are all included in the reference sets. For most instances, MOCQPSO obtains more non-dominated solutions than MOQPSO and MOPSO. For example, MOCQPSO could find 18.40 solutions averagely for instance J120-3 compared to 17.90 for MOQPSO and 16.90 for MOPSO. MOCQPSO also has a better performance on SM for over 80% of instances. For example, its SM for instance J120-1 is only 0.0597, much lower than 0.1354 and 0.1839 for MOQPSO and MOPSO. The above results demonstrate that the proposed algorithm outperforms its competitors in the aspects of convergence and diversity while dealing with multi-objective project scheduling.

Table 8 presents the comparison of MOCQPSO and MOQPSO under different uncertainty, where uncertainty ratio $\theta = [0.3, 0.5, 0.8]$ can be interpreted as low uncertainty, mild uncertainty and high uncertainty. As we can observe, both the average robustness and average infeasibility rate for the two algorithms increase with the increment of uncertainty ratio. MOCQPSO has lower average robustness for its solutions than MOQPSO in all instances. Both algorithms achieve low average feasibility rate while this figure for MOCQPSO is comparatively lower. For example, the infeasibility rate of MOCQPSO for J90-1 with $\theta = 0.8$ is 2.81% and this value for MOQPSO is 4.41%. In addition, the relative deviations of both makespan and cost of MOCQPSO are lower than that of MOQPSO.

The performance of the NSGA-II algorithm is indicated in Table 9. To present the comparison clearly, column " Gap " in Table 9 shows the differences of average values between the proposed MOCQPSO and NSGA-II. For makespan, it is obvious that MOCQPSO achieves a shorter average completion time for all instances. For average total cost, MOCQPSO outperforms NSGA-II in most instances, except for instances J30-1, J30-2 and J60-2. The non-dominated solutions obtained by MOCQPSO are more robust than NSGA-II, except for the J30-2. In a few instances, NSGA-II could achieve better performance of SM than MOCQPSO. For example, the SM of NSGA-II is 0.0273 lower than MOCQPSO for the J30-2 instance. However, MOCQPSO has shown its superiority in most instances indicating its solutions' better uniformity. In Table 9, the SC values represent the ratio of non-dominated solutions of NSGA-II dominated by MOCQPSO. J90-3 is the only instance with SC value lower than 50%. It can be observed that the quality of non-dominated solutions found by MOCQPSO is better than those of NSGA-II.

To test the significance of the results presented above, a non-parametric Wilcoxon Signed-rank Test (Rahman et al., 2020) is conducted to analyse the statistical difference between MOCQPSO and other algorithms. Table 10 presents the p-value obtained for different evaluation criteria. Using a 5% significance level, we can observe that all the obtained p-value are smaller than 0.05 demonstrating the significant difference of MOCQPSO compared with other algorithms.

5.2.3. *Sensitive analysis*

In this section, we evaluate the effects of three main parameters of the proposed SAA-MOCQPSO algorithm, namely the threshold γ, the learning coefficient β and the confidence level $1-\alpha$. We use instance $J30-1$ as an example for the sensitive analysis and keep other parameters fixed as the baseline settings indicated in Table 4 while analysing one parameter. Four evaluation metrics including the average makespan, the average cost, the average robustness and the average set cover (SC) are adopted for the evaluation. To explicitly show the change in the evaluation metrics, we present the "GAP" values by calculating the average differences of objective values between the new parameter and the baseline settings. In this section, the SC value denotes the ratio of non-dominated solutions from the algorithm with the baseline setting dominated by the solutions from the algorithm with new parameter values. The results are reported in Table 11.

To assess the impact of the threshold γ, we compare the algorithm's performance with γ varied within $[0.4, 0.9]$ to the baseline value $\gamma = 0.6$. As we can observe, $\gamma = 0.5$ achieves similar performance with the baseline. When the value of γ is less than 0.5 or greater than 0.6, the performance of both the objective values and SC degrades. It shows that an appropriate selection of γ balances the exploration and exploitation of the algorithm. In our baseline setting, we adopt a dynamic coefficient β which decreases linearly from 0.9 to 0.4 with the increment of iterations. The results clearly indicate that fixing the value of β could lower the performance of the proposed algorithm. As we can observe, the SC value of $\beta = 0.4$ is the lowest among all cases with only 18.73%. It indicates that the convergence speed at $\beta = 0.4$ is slow and the algorithm needs more iterations to search for better solutions. However, if the value of β is large, the algorithm might trap in the sub-optimal solutions. For example, the average objective values on $\beta = 0.9$ are higher than the dynamic setting. For the confidence level $1-\alpha$, the baseline setting is 0.95. The results indicate that the average robustness increases and the SC value decreases with the declining confidence level.

6. Conclusions and future research

This paper focuses on a multi-objective stochastic project scheduling problem considering alternative project structures (MSRCPSP-A) determined by selecting execution methods. The problem is formulated as a stochastic chance-constrained programming model by ensuring the satisfaction of precedence relations and resource constraints higher than a confidence level. Apart from expected makespan and cost, a robustness measurement is introduced as the third objective. A hybrid approach that integrates sample average approximation (SAA) and multi-objective chaotic quantum-behaved particle swarm optimization (MOCQPSO) algorithm is proposed. The MOCQPSO improves the original QPSO from three main aspects: (1) a two-stage learning strategy is proposed to divide the learning procedure into the exploration and

Table 10
Wilcoxon signed-rank test results (p-value).

w. MOCQPSO	Makespan	Cost	Robust	GD	SM	% Ref/SC
MOQPSO	0.0022	0.0022	0.0414	0.0022	0.0096	0.0022
MOPSO	0.0022	0.0022	0.0096	0.0022	0.0121	0.0022
NSGA-II	0.0029	0.0096	0.0121	–	0.0414	0.0108

Table 11
Results of sensitive analysis on SAA-MOCQPSO (GAP).

γ	Makespan	Cost	Robust	SC(%)	β	Makespan	Cost	Robust	SC(%)	$1-\alpha$	Makespan	Cost	Robust	SC(%)
0.4	1.25	212.08	−0.0061	38.25	0.4	3.25	637.75	0.0071	18.73	1	1.38	−18.50	−0.0109	64.59
0.5	−0.13	122.75	−0.0006	56.33	0.5	3.75	1302.75	−0.0149	23.56	0.9	0.29	8.25	−0.0007	48.37
0.6	–	–	–	–	0.6	4.75	146.75	−0.0337	45.25	0.8	2.87	74.44	0.0004	27.76
0.7	−0.52	76.33	0.0085	47.88	0.7	3.43	474.08	0.0078	27.63	0.7	2.27	469.75	0.0068	32.05
0.8	1.53	589.95	0.0009	35.24	0.8	−0.43	376.51	0.0206	37.33	0.6	2.55	170.95	0.0036	22.36
0.9	2.75	170.89	−0.0003	38.58	0.9	0.55	223.15	0.0003	30.94	0.5	1.83	403.37	0.0103	17.31

exploitation stages with a dynamic learning coefficient and a threshold. During the exploration stage, the algorithm explores in a wider solution space. During the exploitation stage, the algorithm focuses more on the local search around global best solutions; (2) chaotic operators including chaotic initialization based on logistic mapping, chaotic crossover and mutation are proposed to improve the diversity of solutions; (3) a sigma method-based elite selection and archive updating strategy is developed. To validate the proposed algorithm's effectiveness, we first evaluate its performance on six benchmark multi-objective functions including ZDT1-3 and DTLZ2-5. The comparison has shown that the proposed MOCQPSO is superior to its original variants. In the second experiment, an instance generator for the MSRCPSP-A problem is developed based on the ProGen generator from PSPLIB. 12 randomly generated instances are used for experimental analysis. The results indicate that the proposed MOCQPSO outperforms both the PSO-based variants and the NSGA-II in solution diversity and convergence. Our study provides a framework for solving multi-objective project scheduling problems with uncertainty attributes which could be beneficial to the decision-making procedures during the management of projects in reality. In the future, case studies on real-life projects can be conducted and other effective approaches are expected to be developed.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Afshar-Nadjafi, B. (2014). Multi-mode resource availability cost problem with recruitment and release dates for resources. *Applied Mathematical Modelling*, *38*(21–22), 5347–5355.

Afshar-Nadjafi, B., Basati, M., & Maghsoudlou, H. (2017). Project scheduling for minimizing temporary availability cost of rental resources and tardiness penalty of activities. *Applied Soft Computing*, *61*, 536–548.

Ballestín, F. (2008). Different codifications and metaheuristic algorithms for the resource renting problem with minimum and maximum time lags. In *Recent advances in evolutionary computation for combinatorial optimization* (pp. 187–202). Springer.

Balouka, N., & Cohen, I. (2019). A robust optimization approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*.

Bruni, M. E., Beraldi, P., Guerriero, F., & Pinto, E. (2011). A heuristic approach for resource constrained project scheduling with uncertain activity durations. *Computers & Operations Research*, *38*(9), 1305–1318.

Charnes, A., & Cooper, W. W. (1959). Chance-constrained programming. *Management Science*, *6*(1), 73–79.

Chen, W.-N., Zhang, J., Chung, H. S.-H., Huang, R.-Z., & Liu, O. (2010). Optimizing discounted cash flows in project scheduling an ant colony optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(1), 64–77.

Cheng, J., Fowler, J., Kempf, K., & Mason, S. (2015). Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. *Computers & Operations Research*, *53*, 275–287.

Chu, X., & Yu, X. (2018). Improved crowding distance for NSGA-II. arXiv preprint arXiv:1811.12667.

Coelho, J., & Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, *213*(1), 73–82.

Davari, M., & Demeulemeester, E. (2019). A novel branch-and-bound algorithm for the chance-constrained resource-constrained project scheduling problem. *International Journal of Productions Research*, *57*(4), 1265–1282.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization* (pp. 105–145). Springer.

dos Santos Coelho, L. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications*, *37*(2), 1676–1683.

Eberhart, R., & Kennedy, J. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks. Vol. 4* (pp. 1942–1948). Citeseer.

Feng, Z.-k., Niu, W.-j., & Cheng, C.-t. (2017). Multi-objective quantum-behaved particle swarm optimization for economic environmental hydrothermal energy system scheduling. *Energy*, *131*, 165–178.

Fu, Y., Tian, G., Fathollahi-Fard, A. M., Ahmadi, A., & Zhang, C. (2019). Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint. *Journal of Cleaner Production*, *226*, 515–525.

García-Nieves, J., Ponz-Tienda, J. L., Salcedo-Bernal, A., & Pellicer, E. (2018). The multimode resource-constrained project scheduling problem for repetitive activities in construction projects. *Computer-Aided Civil and Infrastructure Engineering*, *33*(8), 655–671.

Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., & Javanmardi, A. (2013). Multi-mode resource-constrained discrete time–cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. *Automation in Construction*, *30*, 216–227.

Goh, C. K., Tan, K. C., Liu, D., & Chiam, S. C. (2010). A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, *202*(1), 42–54.

Gomes, H. C., das Neves, F. d. A., & Souza, M. J. F. (2014). Multi-objective metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations. *Computers & Operations Research*, *44*, 92–104.

Habibi, F., Barzinpour, F., & Sadjadi, S. (2018). Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, *3*(2), 55–88.

Hauder, V. A., Beham, A., Raggl, S., Parragh, S. N., & Affenzeller, M. (2020). Resource-constrained multi-project scheduling with activity and time flexibility. *Computers & Industrial Engineering*, *150*, Article 106857.

Joy, J., Rajeev, S., & Narayanan, V. (2016). Particle swarm optimization for resource constrained-project scheduling problem with varying resource levels. *Procedia Technology*, *25*, 948–954.

Kanso, A., & Smaoui, N. (2009). Logistic chaotic maps for binary numbers generations. *Chaos, Solitons & Fractals*, *40*(5), 2557–2568.

Ke, H., & Liu, B. (2005). Project scheduling problem with stochastic activity duration times. *Applied Mathematics and Computation*, *168*(1), 342–353.

Kellenbrink, C., & Helber, S. (2015). Scheduling resource-constrained projects with a flexible project structure. *European Journal of Operational Research*, *246*(2), 379–391.

Khalili-Damghani, K., Abtahi, A.-R., & Tavana, M. (2013). A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems. *Reliability Engineering & System Safety*, *111*, 58–75.

Kolisch, R., & Sprecher, A. (1997). PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European Journal of Operational Research*, *96*(1), 205–216.

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.

Lashkari, M., & Moattar, M. (2017). Improved COA with chaotic initialization and intelligent migration for data clustering. *Journal of AI and Data Mining*, *5*(2), 293–305.

Laszczyk, M., & Myszkowski, P. B. (2019). Improved selection in evolutionary multi–objective optimization of multi–skill resource–constrained project scheduling problem. *Information Sciences*, *481*, 412–431.

Li, C., Zhou, J., Lu, P., & Wang, C. (2015). Short-term economic environmental hydrothermal scheduling using improved multi-objective gravitational search algorithm. *Energy Conversion and Management*, *89*, 127–136.

Liang, Y., Cui, N., Hu, X., & Demeulemeester, E. (2020). The integration of resource allocation and time buffering for bi-objective robust project scheduling. *International Journal of Productions Research*, *58*(13), 3839–3854.

Lin, Q., Li, J., Du, Z., Chen, J., & Ming, Z. (2015). A novel multi-objective particle swarm optimization with multiple search strategies. *European Journal of Operational Research*, *247*(3), 732–744.

Liu, T., Jiao, L., Ma, W., Ma, J., & Shang, R. (2016). Cultural quantum-behaved particle swarm optimization for environmental/economic dispatch. *Applied Soft Computing*, *48*, 597–611.

Liu, H., Zhang, X.-W., & Tu, L.-P. (2020). A modified particle swarm optimization using adaptive strategy. *Expert Systems with Applications*, *152*, Article 113353.

Lu, H., Wang, X., Fei, Z., & Qiu, M. (2014). The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms. *Mathematical Problems in Engineering*, *2014*.

Lu, Y., Zhou, J., Qin, H., Wang, Y., & Zhang, Y. (2011). Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects. *Engineering Applications of Artificial Intelligence*, *24*(2), 378–387.

Ma, W., Che, Y., Huang, H., & Ke, H. (2016). Resource-constrained project scheduling problem with uncertain durations and renewable resources. *International Journal of Machine Learning and Cybernetics*, *7*(4), 613–621.

Ma, G., Gu, L., & Li, N. (2015). Scenario-based proactive robust optimization for critical-chain project scheduling. *Journal of Construction Engineering and Management*, *141*(10), Article 04015030.

Meneghini, I. R., Alves, M. A., Gaspar-Cunha, A., & Guimaraes, F. G. (2020). Scalable and customizable benchmark problems for many-objective optimization. *Applied Soft Computing*, *90*, Article 106139.

Moslehi, G., & Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, *129*(1), 14–22.

Nemati-Lafmejani, R., Davari-Ardakani, H., & Najafzad, H. (2019). Multi-mode resource constrained project scheduling and contractor selection: mathematical formulation and metaheuristic algorithms. *Applied Soft Computing*, *81*, Article 105533.

Nguyen, S., Thiruvady, D., Ernst, A. T., & Alahakoon, D. (2019). A hybrid differential evolution algorithm with column generation for resource constrained job scheduling. *Computers & Operations Research*, *109*, 273–287.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 603–615.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *280*(2), 395–416.

Qi, J., Guo, B., Lei, H., & Zhang, T. (2014). Solving resource availability cost problem in project scheduling by pseudo particle swarm optimization. *Journal of Systems Engineering and Electronics*, *25*(1), 69–76.

Rahman, H. F., Chakrabortty, R. K., & Ryan, M. J. (2020). Memetic algorithm for solving resource constrained project scheduling problems. *Automation in Construction*, *111*, Article 103052.

Rajeev, S., Kurian, S., & Paul, B. (2015). A modified serial scheduling scheme for resource constrained project scheduling weighted earliness tardiness problem. *International Journal of Information and Decision Sciences*, *7*(3), 241–254.

Ribeiro, P., Paiva, A., Varajão, J., & Dominguez, C. (2013). Success evaluation factors in construction project management some evidence from medium and large portuguese companies. *KSCE Journal of Civil Engineering*, *17*(4), 603–609.

Schütz, P., Tomasgard, A., & Ahmed, S. (2009). Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research*, *199*(2), 409–419.

Servranckx, T., & Vanhoucke, M. (2019). A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs. *European Journal of Operational Research*, *273*(3), 841–860.

Shen, X.-N., Han, Y., & Fu, J.-Z. (2017). Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. *Soft Computing*, *21*(21), 6531–6554.

Singh, M. R., & Mahapatra, S. S. (2016). A quantum behaved particle swarm optimization for flexible job shop scheduling. *Computers & Industrial Engineering*, *93*, 36–44.

Sobel, M. J., Szmerekovsky, J. G., & Tilson, V. (2009). Scheduling projects with stochastic activity duration to maximize expected net present value. *European Journal of Operational Research*, *198*(3), 697–705.

Sun, J., Chen, W., Fang, W., Wun, X., & Xu, W. (2012). Gene expression data analysis with the clustering method based on an improved quantum-behaved particle swarm optimization. *Engineering Applications of Artificial Intelligence*, *25*(2), 376–391.

Sun, J., Fang, W., Wu, X., Xie, Z., & Xu, W. (2011). QoS multicast routing using a quantum-behaved particle swarm optimization algorithm. *Engineering Applications of Artificial Intelligence*, *24*(1), 123–131.

Sun, J., Feng, B., & Xu, W. (2004). Particle swarm optimization with particles having quantum behavior. In *Proceedings of the 2004 congress on evolutionary computation. Vol. 1* (pp. 325–331). IEEE.

Tang, H., Chen, R., Li, Y., Peng, Z., Guo, S., & Du, Y. (2019). Flexible job-shop scheduling with tolerated time interval and limited starting time interval based on hybrid discrete PSO-SA: An application from a casting workshop. *Applied Soft Computing*, *78*, 176–194.

Tao, S., & Dong, Z. S. (2017). Scheduling resource-constrained project problem with alternative activity chains. *Computers & Industrial Engineering*, *114*, 288–296.

Tao, S., Wu, C., Sheng, Z., & Wang, X. (2018). Stochastic project scheduling with hierarchical alternatives. *Applied Mathematical Modelling*, *58*, 181–202.

Tirkolaee, E. B., Goli, A., Hematian, M., Sangaiah, A. K., & Han, T. (2019). Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms. *Computing*, *101*(6), 547–570.

Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, *201*(2), 409–418.

Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., & Shapiro, A. (2003). The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, *24*(2–3), 289–333.

Wang, L., Huang, H., & Ke, H. (2015). Chance-constrained model for RCPSP with uncertain durations. *Journal of Uncertainty Analysis and Applications*, *3*(1), 12.

Wood, D. A. (2017). Gas and oil project time-cost-quality tradeoff: Integrated stochastic and fuzzy multi-objective optimization applying a memetic, nondominated, sorting algorithm. *Journal of Natural Gas Science and Engineering*, *45*, 143–164.

Xin-gang, Z., Ji, L., Jin, M., & Ying, Z. (2020). An improved quantum particle swarm optimization algorithm for environmental economic dispatch. *Expert Systems with Applications*, Article 113370.

Xiong, J., Chen, Y.-w., Yang, K.-w., Zhao, Q.-s., & Xing, L.-n. (2012). A hybrid multiobjective genetic algorithm for robust resource-constrained project scheduling with stochastic durations. *Mathematical Problems in Engineering*, *2012*.

Xu, M., Zhang, L., Du, B., Zhang, L., Fan, Y., & Song, D. (2017). A mutation operator accelerated quantum-behaved particle swarm optimization algorithm for hyperspectral endmember extraction. *Remote Sensing*, *9*(3), 197.

Yang, J., Zhou, J., Liu, L., & Li, Y. (2009). A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization (MOPSO). *Computers & Mathematics with Applications*, *57*(11–12), 1995–2000.

Yeganeh, F. T., & Zegordi, S. H. (2020). A multi-objective optimization approach to project scheduling with resiliency criteria under uncertain activity duration. *Annals of Operations Research*, *285*(1), 161–196.

Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, *14*(3), 393–404.

Zhang, W., Li, G., Zhang, W., Liang, J., & Yen, G. G. (2019). A cluster based PSO with leader updating mechanism and ring-topology for multimodal multi-objective optimization. *Swarm and Evolutionary Computation*, *50*, Article 100569.

Zhou, J., Love, P. E., Wang, X., Teo, K. L., & Irani, Z. (2013). A review of methods and algorithms for optimizing construction scheduling. *Journal of the Operational Research Society*, *64*(8), 1091–1105.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, *8*(2), 173–195.