Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

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

# A hybrid genetic algorithm for scheduling jobs sharing multiple resources under uncertainty

Hanyu Gu, Hue Chi Lam [*], Yakov Zinder

*School of Mathematical and Physical Sciences, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Australia*

A R T I C L E   I N F O

*Keywords:*
Genetic algorithm
Local search
Sample average approximation
Stochastic scheduling
Discrete distribution

A B S T R A C T

This study addresses the scheduling problem where every job requires several types of resources. At every point in time, the capacity of resources is limited. When necessary, the capacity can be increased at a cost. Each job has a due date, and the processing times of jobs are random variables with a known probability distribution. The considered problem is to determine a schedule that minimises the total cost, which consists of the cost incurred due to the violation of resource limits and the total tardiness of jobs. A genetic algorithm enhanced by local search is proposed. The sample average approximation method is used to construct a confidence interval for the optimality gap of the obtained solutions. Computational study on the application of the sample average approximation method and genetic algorithm is presented. It is revealed that the proposed method is capable of providing high-quality solutions to large instances in a reasonable time.

[*] Corresponding author.
*E-mail addresses:* hanyu.gu@uts.edu.au (H. Gu), hue.lam@student.uts.edu.au (H.C. Lam), yakov.zinder@uts.edu.au (Y. Zinder).

https://doi.org/10.1016/j.ejco.2022.100050

## 1. Introduction

This paper is concerned with the problem of scheduling the set of jobs $J = \{1, ..., \mathcal{J}\}$ where each job requires several types of resources. The planning horizon is discretised into a number of time periods of equal length indexed $1, ..., H$ and the set of all time periods is denoted by $T = \{1, ..., H\}$. The processing time of each job $j$ is a discrete random variable $p_j$ which assumes integer values and $0 \le p_j^{\min} \le p_j \le p_j^{\max} \le H$, for all $j \in J$, where $p_j^{\min}$ and $p_j^{\max}$ are the minimal and maximal possible processing times of job $j$, respectively. All random variables $p_j$ are independently distributed.

All jobs are available for processing from period $t = 1$ and have to be executed without preemption, i.e. once the processing of a job starts, no interruption is allowed until its completion. A schedule $\mathbf{s}$ specifies for each job $j$ the period $s_j$ when its processing starts. Each job $j$ is given a period $d_j$ and it is desired to complete this job at period $d_j$ or earlier. The tardiness of any job $j$ is $\max\{s_j + p'_j - d_j - 1, 0\}$, where $p'_j$ is a realisation of $p_j$, i.e. an element of the set $\{p_j^{\min}, ..., p_j^{\max}\}$. A realisation of the job's processing times is referred to as a scenario. For any schedule $\mathbf{s} = [s_1, ..., s_{\mathcal{J}}]$, the expected total tardiness is

$$G_1(\mathbf{s}) = G_1(s_1, ..., s_{\mathcal{J}}) = \sum_{j \in J} \sum_{p'_j \in \{p_j^{\min}, ..., p_j^{\max}\}} \Pr(p_j = p'_j) \max\{s_j + p'_j - d_j - 1, 0\} \quad (1)$$

where $\Pr(p_j = p'_j)$ is the probability that the processing time of job $j$ is $p'_j$.

The processing of each job requires $\mathcal{K}$ types of renewable resources. The set of resources is denoted by $K = \{1, ..., \mathcal{K}\}$. In each period $t$, it is desired that the total consumption of resource $k$ should not exceed a certain non-negative integer $R_{tk}$, which will be referred to as the capacity of resource $k$ in period $t$. If the capacity $R_{tk}$ is exceeded, this attracts a certain penalty. During its processing, at each period, a job $j \in J$ consumes $r_{jk}$ units of resource $k$, where $r_{jk}$ is a non-negative integer. Given a schedule $\mathbf{s}$, for any $k \in K$ and any period $t$, denote by $C_{tk}(\mathbf{s})$ the total amount of resource $k$ consumed in period $t$. Since all processing times are random variables, $C_{tk}(\mathbf{s})$ is a random variable. The penalty for the violation of the limit, imposed by the capacity $R_{tk}$, is calculated using the three given parameters $U_{tk}, \alpha_{tk}$ and $\beta_{tk}$, where $\beta_{tk} > \alpha_{tk} > 0$ and $U_{tk}$ is a positive integer. Each extra unit of resource $k$ in the range $[R_{tk}, R_{tk} + U_{tk}]$ increases the penalty by $\alpha_{tk}$, whereas each extra unit of resource $k$ in addition to $R_{tk} + U_{tk}$ increases the penalty by $\beta_{tk}$. In other words, the penalty is calculated as follows:

$$f_{tk}(C_{tk}(\mathbf{s})) = \begin{cases} \alpha_{tk}(C_{tk}(\mathbf{s}) - R_{tk}) & \text{if } R_{tk} < C_{tk}(\mathbf{s}) \le R_{tk} + U_{tk} \\ (\alpha_{tk} - \beta_{tk})U_{tk} + \beta_{tk}(C_{tk}(\mathbf{s}) - R_{tk}) & \text{if } C_{tk}(\mathbf{s}) > R_{tk} + U_{tk} \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

For any schedule $\mathbf{s}$, denote by $G_2(\mathbf{s})$ the expected total penalty for violating the resource capacity in $\mathbf{s}$, i.e.

$$G_2(\mathbf{s}) = G_2(s_1, ..., s_{\mathcal{J}}) = \sum_{t \in T} \sum_{k \in K} \mathbb{E}\left[f_{tk}(C_{tk}(\mathbf{s}))\right] \tag{3}$$

where $\mathbb{E}$ is the expectation operator. The goal is to minimise

$$G(\mathbf{s}) = G_1(\mathbf{s}) + G_2(\mathbf{s}). \tag{4}$$

The considered scheduling problem has been motivated by a project with a rolling stock maintenance centre. This maintenance centre is responsible for the heavy maintenance of passenger trains. The duration of maintenance tasks is uncertain at the time of planning. This is because maintenance duration depends on the condition of the train-set; the availability and composition of the workforce; the availability of spare parts, e.g. bogie, wheel and air conditioning set; and many other factors. The maintenance tasks are carried out by various teams of floor personnel. Each team has different size and skill set. Therefore, the real-world problem is with multiple types of resources. Each passenger train has a desired time window within which the maintenance of this train should commence. This time window is determined by the rolling stock operator, and is based on the validity period of the heavy maintenance that was previously performed. Due to the limited capacity of resources, it may not be possible for all the trains to arrive at the centre within the desired time windows. If the capacity is exceeded, this attracts a certain penalty. Research on the planning of rolling stock maintenance with one type of resource, undertaken by the authors of this paper, has been published in [26], [28] and [27], where Genetic Algorithm was used in [26] and [28]. Below also considers a solution approach based on Genetic Algorithm (GA), but the proposed method is different from the GA used in our previous publications in several ways, which we discuss in detail in Section 2.

The remainder of this paper is organised as follows. Section 2 provides a review of the related work. Section 3 presents a mixed integer linear programming formulation of the considered problem, and an efficient algorithm for computing the value of the objective function. The sample average approximation approach for assessing solution quality is described in Section 4. In section 5, a genetic algorithm enhanced by local search is proposed for solving the stochastic programming problem. This is followed in Section 6 by the results of computational experiments. Conclusions and directions for further research are given in Section 7.

## 2. Related work

The problem considered in this paper falls into the realm of stochastic optimisation, which is well-known to be computationally difficult [8]. Exact methods, developed to solve stochastic programming problems, include Benders decomposition which is commonly referred to as L-shaped algorithm [58], [37], [19]; dual decomposition method [18], [42]; and progressive hedging [49], [63], [4]. The L-shaped algorithm has been successfully applied to solve many different stochastic scheduling problems from production scheduling [33], to project scheduling [15], and operating room scheduling [20,21]. Strategies to enhance the performance of L-shaped algorithm have been proposed in [2] and applied to solve the travelling salesman problem with drone [59], distribution network problem [3], and two-stage 0-1 stochastic program [51].

The difficulty of stochastic programming problems also inspires the development of approximation and bounding algorithms [13] and sampling-based algorithms [52], among which the Sample Average Approximation (SAA) approach has received much attention thus far (see, for example, [61], [33], [41], [10], [70]). The history of SAA dates back to the 1990s when it was first known by the name "stochastic counterpart method" [50], and then "sample-path optimisation method," [48,46]. The convergence of solutions obtained by SAA has been studied in [53], [34], [52], [54], and [16]. SAA method is also used to determine solution quality in stochastic programs via computing the lower and upper bounds [40,61,6,7]. For a comprehensive overview on sampling-based methods for stochastic optimisation, see the survey [30].

Metaheuristics algorithms, such as Particle Swarm Optimisation [24], Greedy Randomised Search Procedures [5], Variable Neighbourhood Search [1], Evolutionary Algorithm [68], and Genetic Algorithm [39] have been successfully used in many different applications of stochastic optimisation. A survey on metaheuristics for stochastic combinatorial optimisation is provided by [12], wherein four metaheuristic approaches, i.e. Ant Colony Optimisation, Evolutionary Computation, Simulated Annealing, and Tabu Search, are considered. For a recent review of metaheuristics, matheuristics, simheuristics, and learnheuristics methods, and their applications to stochastic combinatorial optimisation problems, see [32]. In a recent review paper, [25] report that out of 100 papers about flow-shop scheduling problems under uncertainty published from 2001 to 2016, Genetic Algorithm was the most used metaheuristics method and constituted the largest proportion (42%) of the total study papers (53 papers).

One of the key issues when applying Genetic Algorithm to stochastic optimisation problems is the computation of the objective function, which involves expected values. The majority of work in the literature use simulation methods to approximate the objective function. One strategy is to evaluate all solutions in all generations of GA with a small number of scenarios (see, for example, [39], [29], and [62]). However, it is possible that the final solution returned by GA as the best solution is of low-quality with respect to the true objective value [64]. This issue is addressed in [66] and [31], where a set of good solutions found by GA is re-evaluated with a sample of $10^5$ scenarios to accurately select the best solution. Our paper also proposes a solution approach based on Genetic Algorithm, but in contrast to the above-mentioned studies, we can compute the objective function. So, at each generation, our GA can precisely determine whether a solution is better than another one.

The proposed approach is different from the Genetic Algorithm in our previous studies (see, [28], [26]) in several ways. First, [28] and [26] use only the standard Genetic Algorithm. As the application of local search methods can improve solutions and speed up the convergence process [57], the proposed solution method incorporates local search within the GA procedure. Second, this paper employs SAA for assessing the quality of

GA solutions. Third, [28] and [26] use the random key encoding scheme, which requires a procedure to transform the chromosome into a solution. This paper uses a solution-based representation, wherein a solution is directly represented by a chromosome.

The idea to incorporate local search method within a Genetic Algorithm can be traced back at least to [47] and has been successfully applied to the permutation flowshop scheduling problem [56], parallel machines scheduling problem [55], and job shop scheduling problem [60], among others. Most studies found in the literature design and use the hybrid Genetic Algorithm for deterministic problems. Under the assumption that parameters are known constants, it is easy to incorporate the local search method within the genetic algorithm framework as the evaluation of neighbour solutions does not require much time. However, for optimisation problems where random parameters are present in the objective function, it is expensive to compute the objective function value for a solution. For this reason, studies that use genetic algorithm enhanced by local search for solving stochastic optimisation problems are limited. The local search used in this paper is designed with the aim that the neighbour solutions can be evaluated quickly, making it efficient enough to be used in the GA framework.

The problem under study in this paper is closely related to [33], in which the problem is formulated as a two-stage stochastic model with complete recourse. The first-stage decision is to determine the starting times for jobs based on knowledge about the distribution of the uncertain processing times. The second-stage recourse cost describes a penalty for the amount in which the resource capacity is exceeded. This formulation allows for the application of L-shaped method. To solve large instances, the authors of [33] develop a sequential sampling procedure. In contrast to [33], our paper presents an alternative approach based on Genetic Algorithm metaheuristic. Furthermore, our paper exploits the problem structure and uses an efficient method to compute the value of the objective function. The proposed method is fast enough that it can be included within the optimisation procedure.

## 3. Mixed integer linear programming formulation

In this section, we first present a mixed integer linear programming model for the considered problem. Next, we describe an efficient algorithm for evaluation of the objective function.

### *3.1. Mixed integer linear program*

In order to rewrite the objective function in a more convenient form, let $\Omega$ denote the set of all scenarios, $\pi_\omega$ denote the probability of a scenario $\omega \in \Omega$, and $(p_1^\omega, ..., p_{\mathcal{J}}^\omega)$ denote the realisation of processing times in scenario $\omega$. As discussed in Section 1, for each period $t$ and each resource $k$, there is a regular capacity $R_{tk}$ and an expansion capacity $U_{tk}$. If the total amount of resource $k$ consumed in period $t$ exceeds the regular capacity, then additional resources are needed. Let $y_{tk\omega}$ be the expansion of resource $k$ in period $t$ in the range $[R_{tk}, R_{tk} + U_{tk}]$ under scenario $\omega$. Thus $y_{tk\omega} = \min\{U_{tk}, (\sum_{j\in J}\sum_{s\in S_{jt}^{\omega}} r_{jk}x_{js} - R_{tk})^{+}\}$, where $a^{+} = \max\{0, a\}$. Moreover, if the total amount of resource $k$ consumed in period $t$ exceeds $R_{tk} + U_{tk}$, then additional penalty is incurred. Let $o_{tk\omega}$ be the expansion of resource $k$ in addition to $R_{tk} + U_{tk}$ under scenario $\omega$. Thus $o_{tk\omega} = (\sum_{j\in J}\sum_{s\in S_{jt}^{\omega}} r_{jk}x_{js} - R_{tk} - y_{tk\omega})^{+}$. The objective function can be written as

$$G_1(\mathbf{s})+G_2(\mathbf{s}) = \sum_{\omega\in\Omega}\pi_\omega\left\{\sum_{j\in J}\max\{s_j+p_j^{\omega}-d_j-1,0\}+\sum_{t\in T}\sum_{k\in K}(\alpha_{tk}y_{tk\omega}+\beta_{tk}o_{tk\omega})\right\}. \tag{5}$$

To guarantee job $j$ can complete within the planning horizon under all scenarios, the latest time to start $j$ is restricted to $t_j^{\max} = H - p_j^{\max} + 1$. Therefore, job $j$ can start in any periods in $T_j = \{1, ..., t_j^{\max}\}$. For any job $j$, any period $t$, and any scenario $\omega$, let

$$S_{jt}^{\omega} = \{\tau \mid \tau + p_j^{\omega} - 1 \geq t, \tau \leq t\} \cap\ T_j \tag{6}$$

denote the set of starting times which makes job $j$ to be processed during time period $t$. For each $j \in J$ and each $t \in T_j$, let

$$x_{jt} = \begin{cases} 1 & \text{if job } j \text{ starts in period } t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Then, the starting time for each job $j$ can be determined by

$$s_j = \sum_{t\in T_j} t x_{jt}$$

where variables $x_{jt}$ are obtained by solving the following mixed integer linear program

$$\text{(MILP)}\ z^* = \min\sum_{\omega\in\Omega}\pi_\omega\left\{\sum_{j\in J}\sum_{t\in T_j}\max\{t+p_j^{\omega}-d_j-1,0\}\ x_{jt}\right.$$
$$\left.+\sum_{t\in T}\sum_{k\in K}(\alpha_{tk}y_{tk\omega}+\beta_{tk}o_{tk\omega})\right\} \tag{8}$$

$$\text{s.t.}\ \sum_{t\in T_j} x_{jt} = 1, \qquad j \in J \tag{9}$$

$$\sum_{j\in J}\sum_{s\in S_{jt}^{\omega}} r_{jk}x_{js} - y_{tk\omega} - o_{tk\omega} \leq R_{tk},\ \omega\in\Omega,\ t\in T,\ k\in K \tag{10}$$

$$0 \leq y_{tk\omega} \leq U_{tk}, \qquad \omega\in\Omega,\ t\in T,\ k\in K \tag{11}$$

$$o_{tk\omega} \geq 0, \qquad \omega\in\Omega,\ t\in T,\ k\in K \tag{12}$$

$$x_{jt} \in \{0,1\}, \qquad j\in J,\ t\in T_j \tag{13}$$

The objective function (8) is a weighted sum of two components: the expected total tardiness and the expected total penalty for violating the capacity, which we wish to minimise. Constraint (9) ensures that all jobs are completed by the end of planning horizon. Constraint (10) calculates the additional units of resource $k$ required beyond the capacity $R_{tk}$ in period $t$ under scenario $\omega$. Constraints (11) - (12) describe the domain for $y_{tk\omega}$ and $o_{tk\omega}$, respectively. Constraint (13) states the integrality restriction on the decision variable $x_{jt}$.

As mentioned in the Introduction section, the probability distribution of $p_j, j \in J$ is discrete. The assumption of discrete processing times is reasonable because a job is scheduled in a discrete time interval (e.g. 1 day). Hence, optimal solution of the original two-stage problem can be obtained by solving the MILP (8)-(13). This technique is mentioned in the tutorial [36]. In this paper, CPLEX is used to solve the problem (8)-(13). It is reported in the computational experiments that only small-sized instances can be solved to optimality.

Alternatively, the considered problem can be formulated as the following two-stage stochastic program with complete recourse [14]:

$$\begin{aligned}
\text{(SP)}\min\ & \sum_{\omega\in\Omega}\pi_\omega\left\{\sum_{j\in J}\sum_{t\in T_j}\max\{t+p_j^\omega-d_j-1,0\}\ x_{jt}\right\}+\sum_{\omega\in\Omega}\pi_\omega\mathcal{Q}(\mathbf{x},\omega)\\
\text{s.t. } & \sum_{t\in T_j}x_{jt}=1, \qquad j\in J\\
& x_{jt}\in\{0,1\}, \qquad j\in J,\ t\in T_j\\
\text{where } \mathcal{Q}(\mathbf{x},\omega) &= \min\sum_{t\in T}\sum_{k\in K}(\alpha_{tk}y_{tk\omega}+\beta_{tk}o_{tk\omega})\\
\text{s.t. } & \sum_{j\in J}\sum_{s\in S_{jt}^\omega}r_{jk}x_{js}-y_{tk\omega}-o_{tk\omega}\le R_{tk},\ \omega\in\Omega,\ t\in T,\ k\in K\\
& 0\le y_{tk\omega}\le U_{tk}, \qquad \omega\in\Omega,\ t\in T,\ k\in K\\
& o_{tk\omega}\ge 0, \qquad \omega\in\Omega,\ t\in T,\ k\in K
\end{aligned}$$

The first stage variable is $x_{jt},\ j\in J,\ t\in T_j$ and the second stage variables are $y_{tk\omega}$ and $o_{tk\omega},\ t\in T,\ k\in K,\ \omega\in\Omega$. In the above formulation, $\mathcal{Q}(\mathbf{x},\omega)$ is a function of the first stage variable $\mathbf{x}$ and of the realisation of processing times under a scenario $\omega\in\Omega$. The first stage requires to determine the starting time of each job, whereas the second stage is concerned with the expansion of the capacity of each resource.

### *3.2. Evaluation of the objective function*

In this section, we discuss the evaluation of the objective function (4). We first present a method to find, for a given schedule, the probability distribution of $C_{tk}(\mathbf{s})$ in (2), $k\in K,\ t\in T$. Then, we show how the objective function can be computed.

Let $y_j(u)$ denote the probability that the processing time of job $j$ is $u$. For any time period $t$ and any resource $k$, job $j$ requires $r_{jk}$ if it is being processed in period $t$. Given a schedule $\mathbf{s} = [s_1, ..., s_{\mathcal{J}}]$, the probability that job $j$ is still being processed in period $t$, denoted by $e_{jt}(s_j)$, can be computed as follows.

$$e_{jt}(s_j) = \begin{cases} \sum_{u=t-s_j+1}^{p_j^{\max}} y_j(u), & t \in \{s_j, ..., H\} \\ 0, & t \in \{1, ..., s_j - 1\} \end{cases}. \tag{14}$$

For any job $j \in J$, any period $t \in T$, and any resource $k \in K$, let

$$\mathscr{E}_{jtk}(s_j) = \begin{cases} r_{jk} & \text{with probability } e_{jt}(s_j) \\ 0 & \text{with probability } 1 - e_{jt}(s_j) \end{cases}. \tag{15}$$

The total resource consumption of resource $k$ in period $t$, resulting from the schedule $\mathbf{s}$, is the sum $C_{tk}(\mathbf{s}) = \sum_{j \in J} \mathscr{E}_{jtk}(s_j)$. For a specific period $t$, the random variables $\mathscr{E}_{jtk}(\mathbf{s})$ are independently distributed. Therefore, $C_{tk}(\mathbf{s})$ is a random variable that follows the Generalised Poisson-Binomial (GPB) distribution [69].

In the following, we propose to compute $C_{tk}(\mathbf{s}), k \in K,\ t \in T$ by means of convolutions. For any schedule $\mathbf{s} = [s_1, \cdots, s_{\mathcal{J}}]$, any period $t \in T$, and any resource $k \in K$, let $P(C_{tk}(\mathbf{s}^l) = i)$ be the probability that $i$ resources of type $k$ from the partial schedule $\mathbf{s}^l = [s_1, \ldots, s_l]$ are consumed in period $t$. Then,

$$P(C_{tk}(\mathbf{s}^1) = r_{1k}) = e_{1t}(s_1) \quad \text{and} \quad P(C_{tk}(\mathbf{s}^1) = 0) = 1 - e_{1t}(s_1)$$

and for all $1 \leq l < \mathcal{J}$

$$\begin{cases} P(C_{tk}(\mathbf{s}^{l+1}) = 0) = (1 - e_{(l+1)t}(s_{l+1}))P(C_{tk}(\mathbf{s}^l) = 0) \\ P(C_{tk}(\mathbf{s}^{l+1}) = i) = (1 - e_{(l+1)t}(s_{l+1}))P(C_{tk}(\mathbf{s}^l) = i) \\ \qquad + e_{(l+1)t}(s_{l+1})P(C_{tk}(\mathbf{s}^l) = i - r_{(l+1)k}), \\ \qquad 1 \leq i < \sum_{j=1}^{l+1} r_j \\ P(C_{tk}(\mathbf{s}^{l+1}) = r_{lk} + r_{(l+1)k}) = e_{(l+1)t}(s_{l+1})P(C_{tk}(\mathbf{s}^l) = r_{lk}) \end{cases}.$$

Such a method was presented in [27] to calculate the exact distribution of the number of trains residing at a maintenance centre on a particular day, which is a random variable that follows Poisson Binomial distribution. We extend this method to the case of Generalised Poisson Binomial distributed random variables. The entire procedure is outlined in Algorithm 1. To simplify notation, we suppress dependence on the given schedule $\mathbf{s}$ in Algorithm 1, and simply use $e_{jt}$ instead of $e_{jt}(\mathbf{s})$, $\mathscr{E}_{jtk}$ instead of $\mathscr{E}_{jtk}(\mathbf{s})$ and $C_{tk}$ instead of $C_{tk}(\mathbf{s})$. Furthermore, we apply Algorithm 1 for each period $t$ and each resource $k$. If it

**Algorithm 1** Direct convolution.

```
1: Input: The set of two-point random variable $\mathscr{E}_j$, which takes the value 0 with probability $1 - e_j$, and $r_j$
   with probability $e_j$, $j \in \{1, ..., \mathcal{J}\}$
2: Output: The probability mass function of the sum $C = \sum_{j=1}^{\mathcal{J}} \mathscr{E}_j$
3: procedure
4:     $\Pr(C^1 = 0) = 1 - e_1$, $\Pr(C^1 = r_1) = e_1$
5:     $\Pr(C^1 = c) = 0$, for $c = 1, ..., r_1 - 1$
6:     Set $\ell = 1$
7:     for $j$ from 2 to $\mathcal{J}$ do
8:         $\Pr(C^{\ell+1} = 0) = (1 - e_j) \cdot \Pr(C^\ell = 0)$
9:         $R = \sum_{i=1}^{j} r_i$
10:        for $i$ from 1 to $R - 1$ do
11:            $\Pr(C^{\ell+1} = i) = e_j \cdot \Pr(C^\ell = i - r_j) + (1 - e_j) \cdot \Pr(C^\ell = i)$
12:        end for
13:        $\Pr(C^{\ell+1} = R) = e_j \cdot \Pr(C^\ell = R - r_j)$
14:        Set $\ell = \ell + 1$
15:    end for
16:    $\Pr(C = c) = \Pr(C^\ell = c)$, for $c = 0, ..., \sum_{j=1}^{N} r_j$
17:    return the probability mass function of $C$
18: end procedure
```

is clear which period and resource are considered, the subscript $t$ and $k$ can be dropped and the notation $e_j$, $\mathscr{E}_j$, and $C$ can be used instead of $e_{jt}$, $\mathscr{E}_{jtk}$, and $C_{tk}$, respectively.

As a result, let $C_k^{\max} = \sum_{j \in J} r_{jk}$, the objective function (4) can be computed as in (16). Therefore, we utilise (16) to accurately calculate the objective function value for solutions that we get from both the hybrid genetic algorithm and the sample average approximation method.

$$\begin{aligned} G_1(\mathbf{s}) + G_2(\mathbf{s}) = G_1(\mathbf{s}) & \\ + \sum_{t \in T} \sum_{k \in K} \sum_{c=R_{tk}+1}^{R_{tk}+U_{tk}} \alpha_{tk}(c - R_{tk}) \Pr(C_{tk}(\mathbf{s}) = c) & \\ + \sum_{t \in T} \sum_{k \in K} \sum_{c=R_{tk}+U_{tk}+1}^{C_k^{\max}} \beta_{tk}(c - R_{kt} - U_{tk}) \Pr(C_{tk}(\mathbf{s}) = c) & \end{aligned} \tag{16}$$

It should be mentioned that the complexity of the direct convolution procedure by Algorithm 1 is $\mathcal{O}(r_{\max}\mathcal{J}^2)$, where $r_{\max} = \max_{j=1}^{\mathcal{J}}\{r_j\}$. The overall complexity to compute the total resource consumption $C_{tk}(\mathbf{s})$, for all resources $k$ and for all time periods $t$ is $\mathcal{O}(HK\mathcal{J}^2 r_{\max})$.

## 4. Sample average approximation

The Sample Average Approximation (SAA), as its name suggests, is an approach of replacing the original problem with its sampling approximation. In this section, we first describe the SAA approach as in [40]. Then we explain how to obtain a statistical estimate for a lower bound on the optimal value $z^*$ of the objective function for the considered stochastic optimisation problem (also called the true problem). The optimality gap and statistical confidence intervals on the quality of an obtained SAA solution are also constructed accordingly.

The following mathematical model describes the SAA problem of the (MILP) with a sample size $N$. Given a sample $\omega^1, \omega^2, ..., \omega^N$ of $N$ scenarios, we can estimate the expected total tardiness in (1) and the expected total penalty for capacity violation in (3) by the average total tardiness and average total penalty over all scenarios, respectively. The resulting SAA problem is a large mixed integer program:

$$z_N^* = \min \frac{1}{N} \sum_{i=1}^{N} \left\{ \sum_{j \in J} \sum_{t \in T_j} \max\{t + p_j^{\omega^i} - d_j - 1, 0\}\, x_{jt} \right.$$
$$\left. + \sum_{t \in T} \sum_{k \in K} (\alpha_{tk} y_{tk\omega^i} + \beta_{tk} o_{tk\omega^i}) \right\} \tag{17}$$

subject to (9), (13)

$$\sum_{j \in J} \sum_{s \in S_{jt}^{\omega^i}} r_{jk} x_{js} - y_{tk\omega^i} - o_{tk\omega^i} \le R_{tk},$$
$$1 \le i \le N,\ t \in T,\ k \in K \tag{18}$$
$$0 \le y_{tk\omega^i} \le U_{tk},\quad 1 \le i \le N,\ t \in T,\ k \in K \tag{19}$$
$$o_{tk\omega^i} \ge 0,\quad 1 \le i \le N,\ t \in T,\ k \in K \tag{20}$$

Let $\mathbf{x}$ be a vector of decision variables $x_{jt}$, $\forall j \in J$, $\forall t \in T_j$. The SAA method provides a means to obtain a statistical estimate for the lower bound and optimality gap. It consists of generating $M$ independent random samples, each of size $N$, and solving the resulting SAA problems. The optimal objective value is denoted by $z_N^m$, $m = 1, ..., M$, and the optimal solution is denoted by $\mathbf{x}_N^m$. The average of the optimal objective values of the $M$ SAA problems

$$\bar{z}_N = \frac{1}{M} \sum_{m=1}^{M} z_N^m \tag{21}$$

is a statistical estimate for a lower bound on $z^*$ [40,43]. The sample variance can be estimated by

$$\sigma_{z_N^*}^2 = \frac{1}{(M-1)} \sum_{m=1}^{M} (z_N^m - \bar{z}_N)^2 \tag{22}$$

Given a solution $\hat{\mathbf{x}}$, we can accurately calculate the objective function value $G(\hat{\mathbf{x}})$ as described in Section 3.2. The quality of the solution $\hat{\mathbf{x}}$ can be determined by computing the optimality gap estimate

$$G(\hat{\mathbf{x}}) - \bar{z}_N. \tag{23}$$

The solution $\hat{\mathbf{x}}$ can be obtained as $\hat{\mathbf{x}} \in \arg\min\{G(\mathbf{x}_N^m) : m = 1, ..., M\}$ or from the method proposed in Section 5. Given a small non-negative number $\alpha$, let $t_{M,\alpha}$ denote the (1-$\alpha$) quantile of the Student's t-distribution [38,35] with $M$ degrees of freedom. We use

$$P\left(G(\hat{\mathbf{x}}) - z^* \leq G(\hat{\mathbf{x}}) - \bar{z}_N + \frac{t_{M-1,\alpha}\ \sigma_{z_N^*}}{\sqrt{M}}\right) \approx 1 - \alpha, \tag{24}$$

to construct the one-sided confidence interval of the level $(1 - \alpha)$ for the optimality gap at $\hat{\mathbf{x}}$. That is,

$$\left[\ 0, G(\hat{\mathbf{x}}) - \bar{z}_N + \frac{t_{M-1,\alpha}\ \sigma_{z_N^*}}{\sqrt{M}}\ \right] \tag{25}$$

Instead of developing a confidence interval for the optimality gap by computing $G(\hat{\mathbf{x}})$, we may develop a confidence interval for the optimality gap by the upper-bound estimator. An estimate of the upper bound for $z^*$ can be obtained by evaluating the solution $\hat{\mathbf{x}}$ using a sample $\omega^1, \omega^2, ..., \omega^{N'}$ of $N'$ scenarios, where $N' > N$ [34]. Let $\hat{z}_{N'}(\hat{\mathbf{x}})$ denote the standard sample mean estimator of $G(\hat{\mathbf{x}})$ and $\sigma^2_{\hat{z}_{N'}(\hat{\mathbf{x}})}$ denote the standard sample variance estimator. An approximate $(1 - 2\alpha)$-level confidence interval for the optimality gap at $\hat{\mathbf{x}}$ is

$$\left[\ 0, \hat{z}_{N'}(\hat{\mathbf{x}}) - \bar{z}_N + \frac{t_{M-1,\alpha}\ \sigma_{z_N^*}}{\sqrt{M}} + \frac{t_{N'-1,\alpha}\ \sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}}{\sqrt{N'}}\ \right] \tag{26}$$

The above procedure for determining solution quality in stochastic programs was suggested in [43] and developed in [40].

## 5. Hybrid genetic algorithm

Genetic Algorithm (GA) is a population-based search algorithm that can generates high-quality solution for many mathematical optimisation problems. GA has also received significant attention and has been successfully used in many different applications of stochastic optimisation. For example, [17] propose a two-stage GA for solving a stochastic parallel machine scheduling problem; [11] propose a biased random-key GA for the two-stage capacitated facility location problem; [71] develop a GA for solving large-scale instances of the two-stage stochastic programming for single yard crane scheduling problem. In this section, we propose a Genetic Algorithm enhanced by local search for solving large instances of the considered problem. We refer to the proposed method as the Hybrid Genetic Algorithm (HGA).

Our HGA utilises genetic algorithm for global search and an efficient local search method for intensification purposes. Algorithm 2 presents the pseudocode for the HGA. The input is comprised of the population size ($P$), crossover probability ($\lambda_c$), mutation

**Algorithm 2** Hybrid genetic algorithm.

```
1: **Input**: population size ($P$), crossover probability ($\lambda_c$), mutation probability ($\lambda_m$), termination condition
   ($\text{GEN}_{\max}$)
2: **procedure**
3:     Initialise population consisting of $P$ randomly generated chromosomes
4:     **while** $\text{GEN}_{\max}$ is not satisfied **do**
5:         **for** $i$ from 1 to $P$ **do**
6:             Use the binary tournament to select two parents from the population
7:             With probability $\lambda_c$, apply the half-uniform crossover on the selected parents
8:             With probability $\lambda_m$, apply the uniform mutation on the offspring
9:         **end for**
10:        Use local search to educate or improve each chromosome in the population
11:    **end while**
12:    **return** the best solution found
13: **end procedure**
```

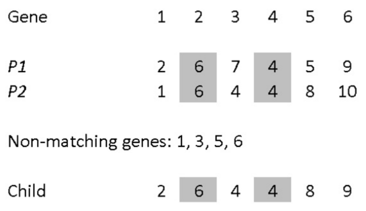
probability ($\lambda_m$), and maximum number of generations ($\text{GEN}_{\max}$). First, the initial population of $P$ chromosomes is created randomly (line 3). Then, the main loop (lines 4 to 11) performs the half-uniform crossover, uniform mutation, and local search on the current population until the maximum number of generations ($\text{GEN}_{\max}$) is reached. The inner loop (lines 5 to 9) uses the binary tournament operator [45] to pick two parents for reproduction. A new offspring is created by applying the half-uniform crossover operator on the two parents with probability $\lambda_c$. When the crossover operator is not applied, the first parent is handled as the offspring. With probability $\lambda_m$, mutation is performed on each of the generated offspring. Finally, local search procedure is used to improve the offspring solution quality (line 10), where it takes a chromosome $\mu$ as an input and returns a chromosome in the neighbourhood of $\mu$ with the smallest value of the objective function. The details of several components of our HGA are given in the remainder of this section.

## *5.1. Representation of chromosome and definition of fitness function*

In GA, the chromosome representation of a solution is important so that it is susceptible to the required genetic operators and fully characterise the solution. In our implementation of GA, we have chosen the solution-based representation, in which a chromosome directly represents a schedule of jobs. For example:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 14 | 19 | 1 | 11 | 5 | 1 | 1 | 10 |

represents a solution where job 1 starts in period 14, job 2 starts in period 19, and so on. The rationale behind this chromosome representation is that no further decoding procedure is needed since each gene of a chromosome corresponds to the job' starting period. Solution-based representation is widely used in the permutation flow shop scheduling problem [56], and the job-shop problem [65] where the chromosome denotes the order in which jobs are processed. In the remainder of this paper, chromosome, solution, and schedule have the same meaning. The three terms are used interchangeably. The fitness



| Gene | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| *P1* | 2 | 6 | 7 | 4 | 5 | 9 |
| *P2* | 1 | 6 | 4 | 4 | 8 | 10 |

Non-matching genes: 1, 3, 5, 6

| Child | 2 | 6 | 4 | 4 | 8 | 9 |
|---|---|---|---|---|---|---|

**Fig. 1.** Example of the half-uniform crossover.

of a chromosome is calculated as $1/G$, where $G$ is the objective function value. The objective function value is computed as described in Section 3.2. The chromosome with a smaller value of the objective function will have higher fitness.

## *5.2. Parent selection and crossover*

The purpose of the selection operator is to decide which solutions will be selected from the population to generate offspring. In our implementation of GA, we have chosen the binary tournament operator [45]. The binary tournament operator picks two randomly selected individuals from the population, and the one with a better fitness is selected for reproduction. Two rounds of the tournament are executed to get two parent chromosomes from which an offspring will be generated.

The purpose of the crossover operator is to take pair of chromosomes and combine them to produce offspring. The selected parent chromosomes will undergo crossover according to the crossover probability $\lambda_c$. In our implementation of GA, we have chosen to use the half-uniform crossover operator [23,22,44]. The half-uniform crossover operator compares the genes from the two parent chromosomes, copies the matching genes, and places them in the same position in the offspring partial solution. Then, the Hamming distance, i.e. the number of non-matching genes, is calculated. The offspring inherits exactly half of the non-matching genes (at random) from the first parent, and the remaining from the second parent. Fig. 1 shows an example of the half-uniform crossover operator. The half-uniform crossover is suitable for the solution based representation because it is able to promote the level of diversification that our HGA needs. The half-uniform crossover is much less likely than the traditional one- or two-point crossover to produce the same offspring twice from the same parents [22]. The half-uniform crossover with solution based representation had been used in [67] to solve the job scheduling problem in grid computing.

## *5.3. Mutation*

The purpose of the mutation operator is to maintain genetic diversity in the population. In our implementation of GA, the uniform mutation is applied at gene level to retain population diversity. Each gene in the child chromosome is assigned a random number sampled from the uniform distribution $U(0, 1)$. If this random number is not greater than $\lambda_m$, the corresponding gene value is replaced by an integer randomly drawn from 1 to $H - p_j^{\max} + 1$, where $j$ corresponds to the index of the chosen gene in the chromosome.

## *5.4. Local search method*

The purpose of the local search is to find a better solution within the neighbourhood of a given solution. For a given solution $\mathbf{s}$, the local search procedure defines a neighbourhood constituting a set of all solutions that can be reached by applying some search operator to $\mathbf{s}$. This paper proposes the operator *shift*$(\tau, \tau')$ which shifts the starting period from $\tau$ to $\tau'$ for a given job. The procedure **Shift Search** is described in the following. In this procedure, a solution $\mathbf{s} = [s_1, ...s_{\mathcal{J}}]$ represents a chromosome, and $G(\mathbf{s})$ represents the objective function value of $\mathbf{s}$ (inverse of $G(\mathbf{s})$ gives the fitness of $\mathbf{s}$).

```
Procedure Shift Search (s, $G(\mathbf{s})$)

Step 1. Job list $JL \leftarrow \{1, 2, ..., \mathcal{J}\}$
Step 2. If $JL$ is not empty, choose a job $j$ in $JL$, remove $j$ from $JL$, and go to step 3. Otherwise, stop and return $\mathbf{s}$ and $G(\mathbf{s})$.
Step 3. Execute operator shift$(\tau, \tau')$ on $\mathbf{s}$ for $\tau = s_j$ and $\tau' = \{1, ..., H - p_j^{\max} + 1\} \setminus \tau$. Compute the objective function value after each shift operator had been applied and choose the best one. If the objective function value of the best solution is better than $G(\mathbf{s})$, let $\mathbf{s}$ be the best solution and $G(\mathbf{s})$ be the objective function value of the best solution, then go to Step 2.
```

In step 3 of the above procedure, when a shift is performed, the chosen job $j$ is first removed from the current solution $\mathbf{s}$, and the distribution of resource consumption is updated for all periods that are affected by this removal, i.e. $\forall t \in \{\tau, ..., \tau + p_j^{\max} - 1\}$. Let $PMF$ be the resultant probability mass function. After assigning a new starting period $\tau'$ to job $j$, the distribution is again updated by convolving $PMF$ with the distribution of job $j$ for all the impacted periods, i.e. $\forall t \in \{\tau', ..., \tau' + p_j^{\max} - 1\}$. The fitness of a solution resulting from a shift can be computed quickly since the distribution of resource consumption is updated by considering one job at a time, leaving all other jobs and all unaffected periods unchanged.

In the following, we discuss the relationship between the features of the problem instances and the performance of the proposed local search method. For a solution $\mathbf{s}$ consisting of $\mathcal{J}$ jobs, as many as $\mathcal{J} \times (H-1)$ shift operators can be applied to this solution, where $H$ is the planning horizon. For each operator, the total number of operations to compute the value of the objective function is $\mathcal{K} \times H$, where $\mathcal{K}$ is the number of resources. Thus, the complexity of the procedure **Shift Search** is $\mathcal{O}(\mathcal{J} \times \mathcal{K} \times H^2)$. The total running time grows linearly with increases in the number of jobs and resources but quadratically with the increase in planning horizon.

However, it is noted that not all $\mathcal{J} \times (H - 1)$ shift operators can improve the solution $\mathbf{s}$. This observation leads to the development of a dominance rule that reduces the neighbourhood size. The dominance rule is described as follows. Consider a solution $\mathbf{s}$

whose value of the objective function is $G(\mathbf{s})$. Let $\mathbf{s}'$ be a solution obtained by applying a specific operator *shift*$(\tau, \tau')$ to a given job $j$. If the sum of $G_1(\mathbf{s}')$ and $G_2(\mathbf{s} \setminus j)$ is greater than or equal to $G(\mathbf{s})$, then the solution $\mathbf{s}'$ is dominated and the operation *shift*$(\tau, \tau')$ can be ignored, since it is better to start job $j$ on period $\tau$ than $\tau'$. Applying the dominance rule described above to the procedure **Shift Search** can significantly reduce the running time of HGA because only the dominant solutions are considered during the local search process, i.e. the size of the search space is reduced. In our implementation of the HGA, the dominance rule is used in the local search procedure.

## 6. Computational results

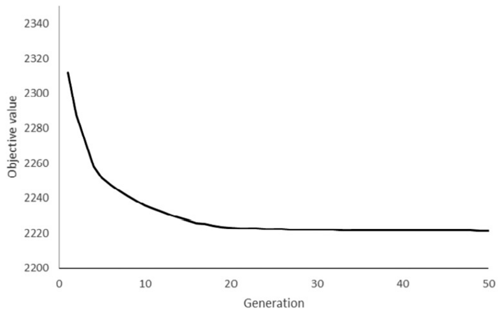
In this section, we report the computational results of the proposed HGA on 54 randomly generated problem instances. We first validate the performance of the HGA against the exact solutions on the set of small problems. Then, we run the HGA on the set of large problems, compare its results with the SAA method and the GA developed by [39]. Next, we report the computational results, and present the 95% confidence interval for the optimality gap at the best solution returned by HGA. Finally, sensitivity analysis on the performance of the proposed HGA and SAA methods is studied.

The proposed hybrid genetic algorithm was implemented in Cython [9]. The testing system was a cluster with Intel Xeon Gold 6150 2.7 GHz 8 cores CPU with 180 GB RAM, running Red Hat Enterprise Linux. The computational facilities were provided by the UTS eResearch High Performance Computer Cluster.

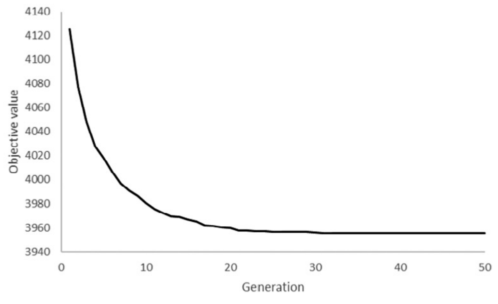
### *6.1. Generation of test instances*

The length of the planning horizon is 50 time periods. In this study, we consider 6 classes of instances: 20 jobs & 5 resources, 40 jobs & 5 resources, 60 jobs & 5 resources, 80 jobs & 5 resources, 100 jobs & 5 resources, and 120 jobs & 5 resources. The random processing time of job (i.e. $p_j$) is assumed to have two possible values with equal probability. The first value is generated from a uniform distribution on the integers $1, .., 50$. The second value is set to the first value plus $\psi$ if the resulting sum is not larger than $H$. Otherwise, the second value is set to the first value minus $\psi$. The value of $\psi$ is chosen to be $\psi = 5$.

For job $j$, the parameter $d_j$ is generated from a uniform distribution on the integers $1, ..., 10$, i.e. $d_j \sim U(1, 10)$. The amount of resource $k$ consumed by job $j$ (i.e. $r_{jk}$) is generated from a uniform distribution on the integers $1, ..., 5$, i.e. $r_{jk} \sim U(1, 5)$. It is assumed that the amount of resource available does not change significantly over the planning horizon, thus we set $R_{tk} = R_k, \forall t \in T$. For the same reason, we set $U_{tk} = U_k, \forall t \in T$. For resource $k$, the parameter $R_k$ is generated from a uniform distribution on the integers between $\bar{p}\bar{r}_k \mathcal{J}/H$ and $\bar{p}\bar{r}_k \mathcal{J}/(0.6H)$, where $\bar{p} = 1/\mathcal{J} \sum_{j=1}^{\mathcal{J}} p_j$ and $\bar{r}_k = 1/\mathcal{J} \sum_{j=1}^{\mathcal{J}} r_{jk}$. For resource $k$, the parameter $U_k$ is set to the greatest integer less than or equal to 10% of $R_k$, i.e. $U_k = \lceil 0.1R_k \rceil$.



(a)



(b)

**Fig. 2.** Change in the objective function value with number of generations of the HGA on the pilot set of (a) small problems and (b) large problems.

For the penalty rates (i.e. $\alpha_{tk}$ and $\beta_{tk}$), it is assumed that they do not vary drastically over the planning horizon, thus we set $\alpha_{tk} = \alpha_k,\ \beta_{tk} = \beta_k, \forall t \in T$. The parameter $\alpha_k$ is generated from a uniform distribution on the integers $1, ..., 10$, i.e. $\alpha_k \sim U(1, 10)$, whereas $\beta_k$ is calculated as $\max\{2\alpha_k, 10\}$. The setting on the above parameters follows the setting in the study by [33].

To test the performance of the proposed solution approaches on problem instances of a varying number of scenarios, we consider two cases:

- Small problem where only ten jobs have two processing times, the remaining jobs have deterministic processing times. The total number of scenario is $2^{10} = 1024$ scenarios.
- Large problem where every job has two processing times resulting in a total of $2^{\mathcal{J}}$ scenarios, where $\mathcal{J}$ is the number of jobs.

## 6.2. HGA parameter setting

For the HGA, we set the values $P = 20$, $\lambda_c = 0.9$, $\lambda_m = 0.01$ for all the instances. The stopping criterion of the proposed HGA was determined using a small pilot set of six problem instances included in the test instances described above. For each instance, ten runs of the algorithm were performed. The results of the experiments are displayed in Fig. 2. The horizontal axis indicates the generation, while the vertical axis indicates the value of the objective function corresponding to the best solution found. From Fig. 2, the algorithm converges rapidly in less than 50 generations. Therefore, $\text{GEN}_{\max} = 50$ is used.

## 6.3. Comparisons between the proposed HGA and CPLEX on small problem instances

As has been discussed in Section 3.1, the stochastic programming problem can be formulated as the model MILP. Solving the model MILP by CPLEX gives an exact

**Table 1**
Comparison between the performance of CPLEX and HGA on small instances.

| Instance | CPLEX | | HGA | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Time | Obj | AvgTime | MinObj | AvgObj | MaxObj | Std | Gap(%) |
| 20-5-1024-S1 | 912 | 1156 | 4 | 1156 | 1156 | 1156 | 0.00 | 0.00% |
| 20-5-1024-S2 | 2264 | 1885 | 5 | 1885 | 1885 | 1885 | 0.00 | 0.00% |
| 20-5-1024-S3 | 6363 | 585 | 8 | 585 | 588 | 591 | 1.85 | 0.51% |
| 40-5-1024-S1 | 3244 | 1093 | 33 | 1093 | 1096 | 1105 | 4.69 | 0.27% |
| 40-5-1024-S2 | 2333 | 1412 | 28 | 1412 | 1413 | 1417 | 1.80 | 0.07% |
| 40-5-1024-S3 | 3386 | 2056 | 25 | 2056 | 2061 | 2068 | 4.40 | 0.24% |
| 60-5-1024-S1 | 13455 | 1553 | 129 | 1557 | 1563 | 1574 | 6.35 | 0.64% |
| 60-5-1024-S2 | 2496 | 1388 | 112 | 1391 | 1394 | 1397 | 1.66 | 0.43% |
| 60-5-1024-S3 | 4298 | 4014 | 70 | 4014 | 4024 | 4043 | 9.01 | 0.25% |
| 80-5-1024-S1 | 5449 | 1821 | 197 | 1822 | 1823 | 1825 | 0.92 | 0.11% |
| 80-5-1024-S2 | 16106 | 1996 | 234 | 1999 | 2001 | 2005 | 2.12 | 0.25% |
| 80-5-1024-S3 | 4921 | 2818 | 208 | 2819 | 2840 | 2878 | 18.01 | 0.78% |

**Notes:** Instance '20-5-1024-S1' means 20 jobs, 5 resources, and 1024 scenarios;
'S1' means first instance for the jobs-resources-scenarios combination.

solution to the stochastic programming problem. Table 1 presents results of the proposed HGA and the CPLEX solver for the small problem instances. For all the instances, CPLEX stopped after reaching a 5-hour limit or when an optimal solution was found. The column "Time" gives the solve time (in seconds) used by CPLEX to obtain the optimal solution (column "Obj"). For each instance, Table 1 also shows the average running time in seconds (AvgTime), the minimum, average, and maximum values of the objective function (MinObj, AvgObj, MaxObj), and the standard deviation (Std) obtained for ten runs of the proposed HGA. Solution quality of the HGA is measured by the percentage relative difference $\text{Gap}(\%) = (\text{AvgObj} - \text{Obj})/\text{Obj} \times 100$, and is reported under the column titled "Gap(%)".

The results in Table 1 show that the HGA can find high-quality solutions with low running time in comparison with the optimal solutions by CPLEX. When $\mathcal{J} \leq 40$, the algorithm obtains optimal solutions in all 6 instances (column "MinObj"). When $\mathcal{J} \geq 60$, the gaps between HGA and CPLEX increase but are not significant. The average percentage relative differences are about 0.44% and 0.38% for the instances with 60 jobs and 80 jobs, respectively. In terms of the running times of the methods, the proposed HGA can solve all the instances with $\mathcal{J} \leq 40$ in less than 40 seconds. For the instances with 80 jobs, HGA takes less than 4 minutes, whereas CPLEX requires as much as 1 hour to solve the problems to within 1% optimality and significantly more time to prove optimality. This shows the limitation of CPLEX and the strength of the proposed HGA for the considered problem. Furthermore, the running times of the proposed HGA do not vary between instances of the same jobs-resources-scenarios combination and only increase with problem scale, whereas the solve time by CPLEX varies significantly between instances of the same jobs-resources-scenarios combination.

## *6.4. Performance evaluation of the proposed HGA on large problem instances*

We further analyse the performance of the HGA by running the algorithm on 42 large problem instances where every job has two processing times. We evaluate its performance in comparison with the SAA method and the GA in [39], which is referred to as GA-Li. We implement the GA-Li in Python and all computational experiments were conducted on the same computer as that of HGA. The parameter setting for GA-Li is the same as in [39] with population size of 10, mutation probability of 0.05, number of scenarios used for evaluation of objective function being 50 and stopping criteria of 100 generations.

Table 2 provides a summary of results. As mentioned in Section 4, solving the SAA problems (by CPLEX) repeatedly produces a statistical estimate for a lower bound on the optimal value of the objective function for the true problem. The parameters for solving the SAA problems are: $M = 30$ and $N = 50$. We report the statistical lower bound under the column titled '$\bar{z}_N$'. For all the instances, CPLEX stopped after reaching a 1-hour time limit or when an optimal solution was found. For the method SAA, we report the average solve time by CPLEX (in seconds) over 30 SAA problems under the column titled 'AvgTime'; and the standard deviation under the column titled 'Std'. For each SAA problem, the solution obtained from SAA is evaluated as described in Section 3.2. The average objective value of the solutions, over the 30 SAA problems, is reported under the column titled 'AvgObj'. As such, $\bar{z}_N$ is different from AvgObj. For the method HGA, we report the average running time in seconds (AvgTime), the average value of the objective function (AvgObj), and the standard deviation (Std) over ten runs. For the method GA-Li, we report the average value of the objective function (AvgObj) over ten runs. For all three methods, the deviation of the average objective value from $\bar{z}_N$ is reported under the columns titled "Gap(%)", and is calculated by $\text{Gap}(\%) = (\text{AvgObj} - \bar{z}_N)/\bar{z}_N \times 100$. To facilitate the reading, the values obtained by the HGA are in bold if they are better than the values obtained by SAA and GA-Li.

The results in Table 2 show that the proposed HGA outperform the SAA and GA-Li methods. Over all the 42 problem instances, the HGA not only obtains better solutions in all of them but also achieves more stable results across the different runs of the algorithm. For the group of instances with large resource violation penalty (e.g., "highPen-80-5-$2^{80}$-S1"), it is observed that when $\alpha_{tk}$ and $\beta_{tk}$ are 100 times larger, the gap is 6 times larger on average. This is reasonable since the total penalties are a large part of the objective function value and SAA cannot have a good approximation with a small sample size limited by the capability of CPLEX. The average gap to $\bar{z}_N$ is about 1.97% for the HGA, whereas it is about 3.07% for the SAA method, and 10.33% for the GA-Li method. The results suggest that the proposed HGA is capable of obtaining high-quality solutions for large instances of the considered problem. From the experiment, we observe that SAA with $N = 100$ produces superior solution quality than SAA with $N = 50$ at the cost of significantly longer running time. For the running time of the methods, the average time required by HGA was 79 seconds, and the increase in running time between the test cases with (20 jobs, $2^{20}$ scenarios) and with (80 jobs, $2^{80}$ scenarios) is moderate. On

**Table 2**
Comparison between the performance of SAA, GA-Li and HGA on large instances.

| Instance | $\bar{z}_N$ | SAA | | | | GA-Li | | HGA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AvgTime | AvgObj | Std | Gap(%) | AvgObj | Gap(%) | AvgTime | AvgObj | Std | Gap(%) |
| 20-5-$2^{20}$-S1 | 853 | 18 | 869 | 4.44 | 1.88% | 886 | 3.90% | 6 | **865** | 1.64 | **1.41%** |
| 20-5-$2^{20}$-S2 | 1318 | 17 | 1356 | 7.60 | 2.88% | 1384 | 4.99% | 6 | **1348** | 0.87 | **2.28%** |
| 20-5-$2^{20}$-S3 | 472 | 23 | 474 | 1.43 | 0.42% | 518 | 9.84% | 9 | **473** | 0.95 | **0.21%** |
| 20-5-$2^{20}$-S4 | 835 | 22 | 843 | 6.37 | 0.96% | 893 | 6.95% | 9 | 843 | 6.40 | 0.96% |
| 20-5-$2^{20}$-S5 | 1307 | 13 | 1320 | 8.46 | 0.99% | 1326 | 1.45% | 6 | **1315** | 3.72 | **0.61%** |
| 20-5-$2^{20}$-S6 | 574 | 20 | 575 | 0.00 | 0.17% | 626 | 9.06% | 8 | 575 | 0.00 | 0.17% |
| 40-5-$2^{40}$-S1 | 4453 | 69 | 4593 | 33.66 | 3.14% | 4711 | 5.80% | 32 | **4541** | 11.18 | **1.98%** |
| 40-5-$2^{40}$-S2 | 7115 | 39 | 7294 | 25.96 | 2.52% | 7373 | 3.63% | 24 | **7244** | 14.55 | **1.81%** |
| 40-5-$2^{40}$-S3 | 2458 | 46 | 2520 | 18.24 | 2.52% | 2601 | 5.82% | 28 | **2492** | 2.73 | **1.38%** |
| 40-5-$2^{40}$-S4 | 1199 | 49 | 1202 | 3.12 | 0.25% | 1331 | 11.01% | 44 | **1200** | 0.03 | **0.08%** |
| 40-5-$2^{40}$-S5 | 1813 | 37 | 1849 | 10.12 | 1.99% | 1951 | 7.61% | 38 | **1834** | 1.48 | **1.16%** |
| 40-5-$2^{40}$-S6 | 1259 | 135 | 1267 | 3.54 | 0.64% | 1398 | 11.04% | 52 | 1267 | 3.13 | 0.64% |
| 60-5-$2^{60}$-S1 | 4626 | 108 | 4832 | 25.62 | 4.45% | 5003 | 8.15% | 83 | **4770** | 15.40 | **3.11%** |
| 60-5-$2^{60}$-S2 | 2043 | 88 | 2060 | 10.87 | 0.83% | 2242 | 9.73% | 78 | **2051** | 1.23 | **0.39%** |
| 60-5-$2^{60}$-S3 | 3216 | 1440 | 3320 | 19.67 | 3.23% | 3533 | 9.87% | 95 | **3286** | 5.76 | **2.18%** |
| 60-5-$2^{60}$-S4 | 2289 | 2732 | 2342 | 45.46 | 2.32% | 2599 | 13.54% | 150 | **2340** | 10.24 | **2.23%** |
| 60-5-$2^{60}$-S5 | 1488 | 544 | 1494 | 1.87 | 0.40% | 1733 | 16.47% | 150 | **1492** | 0.95 | **0.27%** |
| 60-5-$2^{60}$-S6 | 1843 | 76 | 1863 | 38.04 | 1.09% | 1999 | 8.46% | 117 | **1850** | 0.16 | **0.38%** |
| 80-5-$2^{80}$-S1 | 4982 | 105 | 5126 | 18.71 | 2.89% | 5471 | 9.81% | 174 | **5077** | 7.62 | **1.91%** |
| 80-5-$2^{80}$-S2 | 3266 | 3186 | 3359 | 16.57 | 2.85% | 3607 | 10.43% | 228 | **3323** | 6.44 | **1.75%** |
| 80-5-$2^{80}$-S3 | 3387 | 546 | 3507 | 17.01 | 3.54% | 3802 | 12.24% | 185 | **3470** | 9.94 | **2.45%** |
| 80-5-$2^{80}$-S4 | 2074 | 2684 | 2084 | 4.50 | 0.48% | 2391 | 15.28% | 311 | **2079** | 0.13 | **0.24%** |
| 80-5-$2^{80}$-S5 | 5978 | 100 | 6164 | 25.69 | 3.11% | 6422 | 7.43% | 221 | **6097** | 7.27 | **1.99%** |
| 80-5-$2^{80}$-S6 | 4271 | 1563 | 4453 | 24.14 | 4.26% | 4810 | 12.62% | 296 | **4417** | 16.93 | **3.42%** |

**Table 2** (*continued*)

| Instance | $\bar{z}_N$ | SAA | | | | GA-Li | | HGA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AvgTime | AvgObj | Std | Gap(%) | AvgObj | Gap(%) | AvgTime | AvgObj | Std | Gap(%) |
| 100-5-$2^{100}$-S1 | 4030 | 265 | 4093 | 52.18 | 1.56% | 4449 | 10.40% | 466 | **4067** | 3.07 | **0.92%** |
| 100-5-$2^{100}$-S2 | 2785 | 1427 | 2792 | 1.70 | 0.25% | 3177 | 14.06% | 503 | **2790** | 0.44 | **0.18%** |
| 100-5-$2^{100}$-S3 | 2803 | 831 | 2810 | 6.74 | 0.25% | 3172 | 13.16% | 443 | **2807** | 0.65 | **0.14%** |
| 100-5-$2^{100}$-S4 | 2648 | 1406 | 2654 | 1.67 | 0.23% | 3101 | 17.11% | 497 | **2652** | 0.27 | **0.15%** |
| 100-5-$2^{100}$-S5 | 2386 | 3297 | 2392 | 2.24 | 0.25% | 2925 | 22.59% | 711 | **2390** | 0.65 | **0.17%** |
| 100-5-$2^{100}$-S6 | 2467 | 1735 | 2472 | 1.41 | 0.20% | 2949 | 19.54% | 535 | **2470** | 0.19 | **0.12%** |
| 120-5-$2^{120}$-S1 | 4603 | 163 | 4660 | 7.77 | 1.24% | 5066 | 10.06% | 768 | **4638** | 2.50 | **0.76%** |
| 120-5-$2^{120}$-S2 | 5973 | 143 | 6120 | 60.52 | 2.46% | 6488 | 8.62% | 760 | **6054** | 2.03 | **1.36%** |
| 120-5-$2^{120}$-S3 | 4019 | 2047 | 4075 | 8.93 | 1.39% | 4522 | 12.52% | 1154 | **4060** | 4.53 | **1.02%** |
| 120-5-$2^{120}$-S4 | 3745 | 252 | 3756 | 4.46 | 0.29% | 4172 | 11.40% | 957 | **3752** | 1.73 | **0.19%** |
| 120-5-$2^{120}$-S5 | 5062 | 201 | 5199 | 14.18 | 2.71% | 5527 | 9.19% | 874 | **5147** | 2.09 | **1.68%** |
| 120-5-$2^{120}$-S6 | 4417 | 188 | 4516 | 14.94 | 2.24% | 4900 | 10.94% | 939 | **4481** | 2.33 | **1.45%** |
| highPen-80-5-$2^{80}$-S1 | 272439 | 115 | 293357 | 1638 | 7.68% | 309366 | 13.55% | 252 | **288346** | 647 | **5.84%** |
| highPen-80-5-$2^{80}$-S2 | 86381 | 3373 | 99267 | 1582 | 14.92% | 114764 | 32.86% | 381 | **96918** | 314 | **12.20%** |
| highPen-80-5-$2^{80}$-S3 | 86882 | 556 | 98743 | 1907 | 13.65% | 128821 | 48.27% | 286 | **96694** | 791 | **11.29%** |
| highPen-100-5-$2^{100}$-S1 | 119260 | 415 | 131389 | 936 | 10.17% | 143159 | 20.04% | 555 | **128975** | 298 | **8.15%** |
| highPen-100-5-$2^{100}$-S2 | 2803 | 3600 | 3046 | 169 | 8.67% | 3230 | 15.23% | 715 | **2823** | 2.36 | **0.71%** |
| highPen-100-5-$2^{100}$-S3 | 2855 | 2104 | 3229 | 145 | 13.10% | 3575 | 25.22% | 530 | **2956** | 9.21 | **3.54%** |

**Notes:** Instance '20-5-$2^{20}$-S1' means 20 jobs, 5 resources, and $2^{20}$ scenarios;
'S1' means first instance for the jobs-resources-scenarios combination;
Instance 'highPen-80-5-$2^{80}$-S1' is instance '80-5-$2^{80}$-S1' but the values of $\alpha_{tk}$ and $\beta_{tk}$, for all $t$ and $k$ are 100 times larger.

**Table 3**
The 95% confidence interval for the optimality gap at $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the best solution obtained from the HGA.

| Instance | 20-5-$2^{20}$-S1 | 20-5-$2^{20}$-S2 | 20-5-$2^{20}$-S3 | 40-5-$2^{40}$-S1 | 40-5-$2^{40}$-S2 | 40-5-$2^{40}$-S3 |
|---|---|---|---|---|---|---|
| $\bar{z}_N$ | 852.83 | 1317.85 | 472.06 | 4453.38 | 7115.35 | 2457.71 |
| $\sigma_{z_N^*}$ | 12.83 | 29.73 | 1.63 | 61.97 | 93.28 | 36.44 |
| | **CI construction using** (25) | | | | | |
| $G(\hat{\mathbf{x}})$ | 863.75 | 1347.88 | 473.00 | 4527.02 | 7214.22 | 2490.35 |
| $\frac{t_{M-1,0.05}\ \sigma_{z_N^*}}{\sqrt{M}}$ | 3.98 | 9.22 | 0.50 | 19.22 | 28.94 | 11.30 |
| 95% CI | [0, 14.90] | [0, 39.25] | [0, 1.45] | [0, 92.87] | [0, 127.81] | [0, 43.94] |
| | **CI construction using** (26) | | | | | |
| $\hat{z}_{N'}(\hat{\mathbf{x}})$ | 864.43 | 1357.54 | 481.00 | 4521.60 | 7210.75 | 2490.75 |
| $\sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}$ | 128.12 | 235.70 | 3.91 | 481.66 | 602.93 | 253.79 |
| $\frac{t_{M-1,0.025}\ \sigma_{z_N^*}}{\sqrt{M}}$ | 4.79 | 11.10 | 0.61 | 23.14 | 34.83 | 13.61 |
| $\frac{t_{N'-1,0.025}\ \sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}}{\sqrt{N'}}$ | 2.51 | 4.62 | 0.08 | 9.45 | 11.83 | 4.98 |
| 95% CI | [0, 18.90] | [0, 55.42] | [0, 9.63] | [0, 100.81] | [0, 142.06] | [0, 51.63] |

the other hand, although all the SAA problems can be solved to optimality, the average solve time of CPLEX was 474 seconds, which is about six times more than HGA. It is also noted that for the group of instances with 120 jobs, SAA requires less time than HGA on average.

Tables 3 and 4 present the 95% confidence interval (CI) for the optimality gap at $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the best solution found after ten runs of the HGA for each instance. We report the test results for developing the CI based on $G(\hat{\mathbf{x}})$ according to (25) and based on the upper-bound estimator $\hat{z}_{N'}(\hat{\mathbf{x}})$ according to (26) (see Section 4). The upper-bound estimator $\hat{z}_{N'}(\hat{\mathbf{x}})$ is obtained using $N' = 10^4$. From Table 3, it can be observed that tighter confidence interval on the optimality gap can be obtained using the CI construction based on $G(\hat{\mathbf{x}})$ rather than $\hat{z}_{N'}(\hat{\mathbf{x}})$. Indeed, the latter yields confidence interval widths that are within 2.78% and 2.09% from the upper bound $G(\hat{x})$, roughly 1.68 and 1.12 times larger than that obtained from the former, for the instances with 20 and 40 jobs, respectively. This is expected since the random CI width in (26) consists of not only the sampling error from estimating the lower bound but also that from the upper-bound estimator. The same observation can be seen in Table 4, in which constructing the CI using (25) results in sufficiently tight confidence intervals.

## 6.5. Sensitivity analysis

The results in Table 2 indicate that instance 80-5-$2^{80}$-S2 is harder to solve than the other instances. The reason could be that there is a well-balanced share between the expected total tardiness of jobs and the expected total penalty for violating the resource capacity, which results in both SAA and HGA spending a substantial amount of time in an attempt to find the solution that minimises the total cost. In this section, using the large instance 80-5-$2^{80}$-S2, we first analyse the performance of SAA and HGA with the variation of two problem parameters: $H$ and $\psi$. Because the time required by the proposed HGA to solve the instance is at most 30 minutes, we impose a 30-minute time

**Table 4**
The 95% confidence interval for the optimality gap at $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the best solution obtained from the HGA (continue).

| Instance | 60-5-$2^{20}$-S1 | 60-5-$2^{20}$-S2 | 60-5-$2^{20}$-S3 | 80-5-$2^{40}$-S1 | 80-5-$2^{40}$-S2 | 80-5-$2^{40}$-S3 |
|---|---|---|---|---|---|---|
| $\bar{z}_N$ | 4625.95 | 2042.61 | 3215.88 | 4982.28 | 3266.44 | 3386.94 |
| $\sigma_{z^*_N}$ | 57.47 | 15.18 | 46.31 | 44.95 | 47.61 | 50.98 |
| | **CI construction using** (25) | | | | | |
| $G(\hat{\mathbf{x}})$ | 4749.98 | 2050.58 | 3280.43 | 5068.31 | 3314.92 | 3460.76 |
| $\frac{t_{M-1,0.05}\ \sigma_{z^*_N}}{\sqrt{M}}$ | 17.83 | 4.71 | 14.37 | 13.94 | 14.77 | 15.81 |
| 95% CI | [0, 141.86] | [0, 12.68] | [0, 78.91] | [0, 99.97] | [0, 63.25] | [0, 89.63] |
| | **CI construction using** (26) | | | | | |
| $\hat{z}_{N'}(\hat{\mathbf{x}})$ | 4753.09 | 2057.37 | 3277.23 | 5079.33 | 3334.72 | 3475.48 |
| $\sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}$ | 505.62 | 122.74 | 292.91 | 389.73 | 284.11 | 342.30 |
| $\frac{t_{M-1,0.025}\ \sigma_{z^*_N}}{\sqrt{M}}$ | 21.46 | 5.67 | 17.29 | 16.78 | 17.77 | 19.03 |
| $\frac{t_{N'-1,0.025}\ \sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}}{\sqrt{N'}}$ | 9.92 | 2.41 | 5.75 | 7.65 | 5.57 | 6.72 |
| 95% CI | [0, 158.52] | [0, 22.84] | [0, 84.39] | [0, 121.48] | [0, 91.62] | [0, 114.29] |

**Table 5**
Sensitivity analysis on the performance of SAA and HGA with $H$ for instance 80-5-$2^{80}$-S2.

| Instance | $H$ | SAA | | | | HGA | | |
|---|---|---|---|---|---|---|---|---|
| | | AvgTime | AvgObj | Std | AvgMIP-gap(%) | AvgTime | AvgObj | Std |
| 80-5-$2^{80}$-S2 | 50 | 1800 | 3360.70 | 19.17 | 0.60% | 228 | 3323.29 | 6.44 |
| | 60 | 1800 | 2115.50 | 3.32 | 0.11% | 430 | 2109.48 | 0.59 |
| | 70 | 1800 | 2108.32 | 2.73 | 0.19% | 655 | 2104.69 | 0.57 |
| | 80 | 1800 | 2109.44 | 2.41 | 0.21% | 906 | 2104.90 | 0.54 |
| | 90 | 1800 | 2107.40 | 2.05 | 0.18% | 1244 | 2104.69 | 0.43 |
| | 100 | 1819 | 2108.45 | 2.20 | 0.24% | 1521 | 2104.31 | 0.21 |

limit for CPLEX. This ensures a fair comparison. Next, using the large instances, the performance of HGA is analysed with the variation of two GA parameters: $\lambda_m$ and $\lambda_c$.

Table 5 presents the analysis on the performance of SAA and HGA with $H \in \{50, 60, 70, 80, 90, 100\}$ when the remaining problem parameters stay unchanged. In this table, the average optimality gap reported by CPLEX across ten runs is reported under the column titled "AvgMIP-gap(%)". It is observed that the required computation time by HGA increases when $H$ increases. This is expected since increasing $H$ will inevitably increase the number of moves in the local search procedure, which is the most time-consuming component in the proposed HGA. Indeed, the HGA took, on average, 6.67 times longer to solve instance 80-5-$2^{80}$-S2 with $H = 100$ than with $H = 50$. When looking at the average objective value when $H$ is increased from 50 to 60, we note that the AvgObj is improved since the penalty incurred for exceeding the capacity of resources is reduced. As $H$ continues to be increased from 70 to 100, there is little difference in the AvgObj because resource capacity is always sufficient, and the cost incurred due to the violation of resource limits is dominated by the total tardiness of jobs. For the SAA method, it is observed that CPLEX fails to solve the SAA problems to optimality before the 30-minute time limit is reached. The lowest average optimality gap reported by CPLEX was 0.11% when $H = 60$.

**Table 6**
Sensitivity analysis on the performance of SAA and HGA with $\psi$ for instance 80-5-$2^{80}$-S2.

| Instance | $\psi$ | SAA | | | | HGA | | |
|---|---|---|---|---|---|---|---|---|
| | | AvgTime | AvgObj | Std | AvgMIP-gap(%) | AvgTime | AvgObj | Std |
| 80-5-$2^{80}$-S2 | 5 | 1800 | 3360.70 | 19.17 | 0.600% | 228 | 3323 | 6.44 |
| | 10 | 1420 | 8002.33 | 24.79 | 0.170% | 207 | 7884 | 14.34 |
| | 15 | 460 | 11211.83 | 88.86 | 0.008% | 216 | 10990 | 14.79 |
| | 20 | 269 | 12384.08 | 108.30 | 0.009% | 213 | 12198 | 0.00 |

**Table 7**
Sensitivity analysis on the performance of HGA with $\lambda_m$ and $\lambda_c$ for large instances, in terms of solution quality and time.

| Group | $\lambda_m = 0.01$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda_c = 0.85$ | | | $\lambda_c = 0.9$ | | | $\lambda_c = 0.95$ | | |
| | AvgObj | Std | AvgTime | $\%_O$ | $\%_S$ | $\%_T$ | $\%_O$ | $\%_S$ | $\%_T$ |
| 20 jobs | 896 | 1.29 | 7 | -0.03% | -22% | 1% | -0.01% | -12% | 2% |
| 40 jobs | 4763 | 10.05 | 28 | -0.12% | -12% | 1% | -0.10% | -2% | 2% |
| 60 jobs | 3372 | 8.29 | 84 | -0.11% | -21% | 2% | -0.06% | -5% | 2% |
| 80 jobs | 3959 | 6.61 | 186 | -0.11% | -17% | 4% | -0.07% | -1% | 5% |
| Average | 3247 | 6.56 | 76 | -0.09% | -18% | 2% | -0.06% | -5% | 3% |
| Group | $\lambda_m = 0.05$ | | | | | | | | |
| | $\lambda_c = 0.85$ | | | $\lambda_c = 0.9$ | | | $\lambda_c = 0.95$ | | |
| | $\%_O$ | $\%_S$ | $\%_T$ | $\%_O$ | $\%_S$ | $\%_T$ | $\%_O$ | $\%_S$ | $\%_T$ |
| 20 jobs | 0.01% | -11% | 44% | -0.06% | -45% | 46% | -0.05% | -36% | 46% |
| 40 jobs | -0.10% | -18% | 56% | -0.13% | -30% | 61% | -0.12% | -32% | 62% |
| 60 jobs | -0.11% | -25% | 80% | -0.20% | -27% | 81% | -0.17% | -16% | 80% |
| 80 jobs | -0.12% | -16% | 86% | -0.17% | -17% | 89% | -0.14% | -17% | 93% |
| Average | -0.08% | -17% | 66% | -0.14% | -30% | 69% | -0.12% | -25% | 70% |

Table 6 presents the analysis on the performance of SAA and HGA with $\psi \in \{5, 10, 15, 20\}$ when the remaining problem parameters stay unchanged. It is observed that an increase in $\psi$ leads to an increase in the objective value due to violation of the resource limits but has little impact on the solution time of HGA. Also, increasing $\psi$ makes the SAA problems substantially easier to solve. This is because the total tardiness of jobs is negligible when $\psi$ is large. Indeed, the ratios of the expected total penalty for violating the resource capacity to the total tardiness of jobs were 0.48, 2.13, 3.47, and 4.21, respectively, for the four settings of $\psi$.

Table 7 presents the analysis on the performance of HGA using a combination of $\lambda_m \in \{0.01, 0.05\}$ and $\lambda_c \in \{0.85, 0.9, 0.95\}$ when $P = 20$ and $\text{GEN}_{\max} = 50$. In this table, the instances are grouped according to the number of jobs. The first four columns are as follows: the instance group (Group), the average objective value (AvgObj), the standard deviation (Std), the average time taken by HGA to terminate (AvgTime), across the ten runs of the HGA with $\lambda_m = 0.01$ and $\lambda_c = 0.85$. In this table, all the percentage differences are relative to the corresponding results obtained by the HGA with $\lambda_m = 0.01$ and $\lambda_c = 0.85$. It can be clearly seen that the solution quality improves at the cost of longer computation time when $\lambda_m$ increases. This is essentially because more genes in the chromosomes would likely be perturbed when $\lambda_m$ is large. As a result of the differences between the original chromosomes and the mutated chromosomes, more iterations of local search would likely be required to explore the search space. Indeed, the HGA with $\lambda_m = 0.05$ consistently obtains a better solution than the HGA with $\lambda_m = 0.01$. However, the former takes on average 1.7 times longer than the latter. The parameter $\lambda_c$ controls how diversified the chromosomes in a population are. Based on the results in Table 7, increasing or decreasing $\lambda_c$ relative to $\lambda_c = 0.9$ can slightly reduce the solution quality. Since the half-uniform crossover operator gives us a highly disruptive crossover [23], using a large crossover probability ($\lambda_c = 0.95$) may decrease the likelihood that better solutions will be kept in the population.

## 7. Conclusion

This paper examines the problem of scheduling jobs where each job requires several types of resources with uncertain job processing times. A method for calculating the exact distributions of resource consumption resulting from a given schedule was presented. Knowing the distribution enables us to compute the value of the expected cost incurred from exceeding the resource capacity. A genetic algorithm enhanced by local search was then proposed to find a schedule that minimises costs. The HGA incorporates the "standard" implementation of the GA as the global search scheme and a simple but effective shift search procedure as the local search scheme. The computational results on small problem instances show that the proposed HGA yields high-quality solutions with low computation time. Although it is possible to solve these small test problems to optimality by using commercial MIP solvers, the computation time grows rapidly as the number of scenarios increases. For large applications, computational results show that the HGA outperforms the SAA strategy and that changes in $\psi$ have a negligible effect on the computation time with this method. However, the influence of planning horizon on the performance of HGA is, due to the embedded local search procedure, more prominent.

Further studies should consider other modelling extensions. For example, one could include a metric to deal with severe uncertainty of job processing times. In particular, the model can be adjusted to determine the required capacity of the resource types to ensure that capacity will not be exceeded with a certain probability. Other metaheuristics can be developed or the HGA proposed in this paper can be adapted to solve this new problem.

## CRediT authorship contribution statement

All authors contributed equally to this work.

## 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

[1] C. Almeder, R.F. Hartl, A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer, Int. J. Prod. Econ. 145 (1) (2013) 88–95, https://doi.org/10.1016/j.ijpe.2012.09.014.

[2] G. Angulo, S. Ahmed, S.S. Dey, Improving the integer L-shaped method, INFORMS J. Comput. 28 (11) (2016) 483–499, https://doi.org/10.1287/ijoc.2016.0695.

[3] A.N. Arslan, W. Klibi, B. Montreuil, Distribution network deployment for omnichannel retailing, Eur. J. Oper. Res. 294 (3) (2021) 1042–1058, https://doi.org/10.1016/j.ejor.2020.04.016.

[4] S. Atakan, S. Sen, A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs, Comput. Manag. Sci. 15 (2018) 501–540, https://doi.org/10.1007/s10287-018-0311-3.

[5] F. Ballestín, R. Leus, Resource-constrained project scheduling for timely project completion with stochastic activity durations, Prod. Oper. Manag. 18 (4) (2009) 459–474, https://doi.org/10.1111/j.1937-5956.2009.01023.x.

[6] G. Bayraksan, D.P. Morton, Assessing solution quality in stochastic programs, Math. Program. 108 (2–3) (2006) 495–514, https://doi.org/10.1007/s10107-006-0720-x.

[7] G. Bayraksan, D.P. Morton, Assessing solution quality in stochastic programs via sampling, INFORMS Tutor. Oper. Res. (2009) 102–122, https://doi.org/10.1287/educ.1090.0065.

[8] G. Bayraksan, D.P. Morton, A sequential sampling procedure for stochastic programming, Oper. Res. 59 (4) (2011) 898–913, https://doi.org/10.1287/opre.1110.0926.

[9] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, K. Smith, Cython: the best of both worlds, Comput. Sci. Eng. 13 (2) (2011) 31–39, https://doi.org/10.1109/MCSE.2010.118.

[10] M.L. Bentaha, O. Battaia, A. Dolgui, A sample average approximation method for disassembly line balancing problem under uncertainty, Comput. Oper. Res. 51 (2014) 111–122, https://doi.org/10.1016/j.cor.2014.05.006.

[11] F.L. Biajioli, A.A. Chaves, L.A.N. Lorena, A biased random-key genetic algorithm for the two-stage capacitated facility location problem, Expert Syst. Appl. 115 (2019) 418–426, https://doi.org/10.1016/j.eswa.2018.08.024.

[12] L. Bianchi, M. Dorigo, L.M. Gambardella, W. Gurjahr, A survey on metaheuristics for stochastic combinatorial optimization, Nat. Comput. 8 (2009) 239–287, https://doi.org/10.1007/s11047-008-9098-4.

[13] J. Birge, R. Wets, Sublinear upper bounds for stochastic programs with recourse, Math. Program. 43 (1989) 131–149, https://doi.org/10.1007/BF01582286.

[14] J.R. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer, New York, 1997.

[15] M.E. Bruni, L.D.P. Pugliese, P. Beraldi, F. Guerriero, A two-stage stochastic programming model for the resource constrained project scheduling problem under uncertainty, in: Proceedings of the 7th International Conference on Operations Research and Enterprise System (ICORES 2018), 2018.

[16] C. Bugg, A. Aswani, Logarithmic sample bounds for sample average approximation with capacity- or budget-constraints, Oper. Res. Lett. 49 (2) (2021) 231–238, https://doi.org/10.1016/j.orl.2021.01.007.

[17] Z. Cao, C. Lin, M. Zhou, C. Zhou, K. Sedraoui, Two-stage genetic algorithm for scheduling stochastic unrelated parallel machines in a just-in-time manufacturing context, IEEE Trans. Autom. Sci. Eng. (2022), https://doi.org/10.1109/TASE.2022.3178126.

[18] C.C. Carøe, R. Schultz, Dual decomposition in stochastic integer programming, Oper. Res. Lett. 24 (1999) 37–45, https://doi.org/10.1016/S0167-6377(98)00050-9.

[19] C.C. Carøe, J. Tind, L-shaped decomposition of two-stage stochastic programs with integer recourse, Math. Program. 83 (1998) 451–464, https://doi.org/10.1007/BF02680570.

[20] B. Denton, J. Viapiano, A. Vogl, Optimisation of surgery sequencing and scheduling decisions under uncertainty, Health Care Manage. Sci. 10 (2007) 13–24, https://doi.org/10.1007/s10729-006-9005-4.

[21] B.T. Denton, A.J. Miller, H.J. Balasubramanian, T.R. Huschka, Optimal allocation of surgery blocks to operating rooms under uncertainty, Oper. Res. 58 (4) (2010) 802–816, https://doi.org/10.1287/opre.1090.0791.

[22] L.J. Eshelman, J.D. Schaffer, Preventing premature convergence in genetic algorithm by preventing incest, in: Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 115–121.

[23] C. García-Martínez, F.J. Rodriguez, M. Lozano, Genetic algorithms, in: R. Martí, P.M. Pardalos, M.G.C. Resende (Eds.), Handbook of Heuristics, Springer, Cham, 2018, pp. 431–464.

[24] J.C. Geng, Z. Cui, X.S. Gu, Scatter search based particle swarm optimization algorithm for earliness/tardiness flowshop scheduling with uncertainty, Int. J. Autom. Comput. 13 (3) (2016) 285–295, https://doi.org/10.1007/s11633-016-0964-8.
[25] E. Gonzalez-Neira, J.R. Montoya-Torres, D. Barrera, Flow-shop scheduling problem under uncertainties: review and trends, Int. J. Ind. Eng. Comput. 8 (2017) 399–426, https://doi.org/10.5267/j.ijiec.2017.2.001.
[26] H. Gu, H.C. Lam, A genetic algorithm approach for scheduling trains maintenance under uncertainty, in: H.A. Le Thi, T. Pham Dinh, N. Nguyen (Eds.), Advanced Computational Methods for Knowledge Engineering, ICCSAMA 2019, in: Advances in Intelligent Systems and Computing, Springer, Cham, 2019, pp. 106–118.
[27] H. Gu, H.C. Lam, Y. Zinder, Planning rolling stock maintenance: optimisation of train arrival dates at a maintenance centre, J. Ind. Manag. Optim. 18 (2) (2022) 747–772, https://doi.org/10.3934/jimo.2020177.
[28] H. Gu, M. Joyce, H.C. Lam, M. Woods, Y. Zinder, A genetic algorithm for assigning train arrival dates at a maintenance centre, Paper presented at the 9th IFAC Conference on Manufacturing Modelling, Management and Control, 28–30 August, Berlin School of Economics and Law, 2019.
[29] J. Gu, M. Gu, C. Cao, X. Gu, A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem, Comput. Oper. Res. 37 (5) (2010) 927–937, https://doi.org/10.1016/j.cor.2009.07.002.
[30] T. Homem-de-Mell, G. Bayraksan, Monte Carlo sampling-based methods for stochastic optimization, Surv. Oper. Res. Manag. Sci. 19 (1) (2014) 56–85, https://doi.org/10.1016/j.sorms.2014.05.001.
[31] S.C. Horng, S.S. Lin, F.Y. Yang, Evolutionary algorithm for stochastic job shop scheduling with random processing time, Expert Syst. Appl. 39 (3) (2012) 3603–3610, https://doi.org/10.1016/j.eswa.2011.09.050.
[32] A.A. Juan, P. Keenan, R. Martí, S. McGarraghy, J. Panadero, P. Carroll, D. Oliva, A review of the role of heuristics in stochastic optimisation: from metaheuristics to learnheuristics, Ann. Oper. Res. (2021), https://doi.org/10.1007/s10479-021-04142-9.
[33] B. Keller, G. Bayraksan, Scheduling jobs sharing multiples resources under uncertainty: a stochastic programming approach, IIE Trans. 42 (1) (2009) 16–30, https://doi.org/10.1080/07408170902942683.
[34] A.J. Kleywegt, A. Shapiro, T. Homem-de-Mello, The sample average approximation method for stochastic discrete optimization, SIAM J. Optim. 12 (2) (2002) 479–502, https://doi.org/10.1137/S1052623499363220.
[35] K. Krishnamoorthy, Handbook of Statistical Distributions with Applications, Chapman & Hall/CRC, New York, 2006.
[36] S. Küçükyavuz, S. Sen, An introduction to two-stage stochastic mixed-integer programming, Tutor. Oper. Res. (2017), https://doi.org/10.1287/educ.2017.0171.
[37] G. Laporte, F.V. Louveaux, The integer L-shaped method for stochastic integer programs with complete recourse, Oper. Res. Lett. 13 (3) (1993) 133–142, https://doi.org/10.1016/0167-6377(93)90002-X.
[38] A.M. Law, W.D. Kelton, Simulation Modeling and Analysis, 5th edn., McGraw-Hill, New York, 2000.
[39] H. Li, E. Demeulemeester, A genetic algorithm for the robust resource leveling problem, J. Sched. 19 (1) (2016) 43–60, https://doi.org/10.1007/s10951-015-0457-6.
[40] W. Mak, D.P. Morton, R.K. Wood, Monte Carlo bounding techniques for determining solution quality in stochastic programs, Oper. Res. Lett. 24 (1–2) (1999) 47–56, https://doi.org/10.1016/S0167-6377(98)00054-6.
[41] C. Mancilla, R. Storer, A sample average approximation approach to stochastic appointment sequencing and scheduling, Math. Program. 44 (8) (2012) 655–670, https://doi.org/10.1080/0740817X.2011.635174.
[42] S. Mitra, P. Garcia-Herreros, I.E. Grossmann, A cross-decomposition scheme with integrated primal–dual multi-cuts for two-stage stochastic programming investment planning problems, Math. Program. 157 (2016) 95–119, https://doi.org/10.1007/s10107-016-1001-y.
[43] V.I. Norkin, G.C. Pflug, A. Ruszczynski, A branch and bound method for stochastic global optimisation, Math. Program. 83 (1998) 425–450, https://doi.org/10.1007/BF02680569.
[44] D. Peña, A. Tchernykh, B. Dorronsoro, P. Ruiz, A novel multi-objective optimization approach to guarantee quality of service and energy efficiency in a heterogeneous bus fleet system, Eng. Optim. (2022), https://doi.org/10.1080/0305215X.2022.2055007.

[45] F. Pezzella, G. Morganti, G. Ciaschetti, A genetic algorithm for the flexible job-shop scheduling problem, Comput. Oper. Res. 35 (10) (2008) 3202–3212, https://doi.org/10.1016/j.cor.2007.02.014.
[46] E.L. Plambeck, B.R. Fu, S.M. Robinson, R. Suri, Sample-path optimization of convex stochastic performance functions, Math. Program., Ser. A B 75 (2) (1996) 137–176, https://doi.org/10.1007/BF02592150.
[47] C.R. Reeves, Genetic algorithms and neighbourhood search, in: T.C. Fogarty (Ed.), Evolutionary Computing, vol. 865, Springer, Berlin, Heidelberg, 1994, pp. 115–130.
[48] S.M. Robinson, Analysis of sample-path optimization, Math. Oper. Res. 21 (3) (1996) 513–528, https://doi.org/10.1287/moor.21.3.513.
[49] R. Rockafellar, R. Wets, Scenarios and policy aggregation in optimization under uncertainty, Math. Oper. Res. 16 (1) (1991) 119–147.
[50] R.Y. Rubinstein, A. Shapiro, Sensitivity Analysis and Stochastic Optimization by the Score Function Method, 1st edition, John Wiley & Sons, Chichester, England, 1993.
[51] K. Ryan, D. Rajan, S. Ahmed, Scenario decomposition for 0-1 stochastic programs: improvements and asynchronous implementation, https://doi.org/10.1109/IPDPSW.2016.119, 2016.
[52] A. Shapiro, Monte Carlo sampling methods, Handb. Oper. Res. Manag. Sci. 10 (2003) 353–425, https://doi.org/10.1016/S0927-0507(03)10006-0.
[53] A. Shapiro, T. Homem-De-Mello, On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs, SIAM J. Optim. 11 (1) (2000) 70–86, https://doi.org/10.1137/S1052623498349541.
[54] A. Shapiro, D. Dentcheva, A. Ruszczyński, Lectures on Stochastic Programming: Modelling and Theory, Society for Industrial and Applied Mathematics, Philadelphia, 2014.
[55] L.C.R. Soares, M.A.M. Carvalho, Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints, Eur. J. Oper. Res. 285 (3) (2020) 955–964, https://doi.org/10.1016/j.ejor.2020.02.047.
[56] L.Y. Tseng, Y.T. Lin, A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, Eur. J. Oper. Res. 198 (1) (2009) 84–92, https://doi.org/10.1016/j.ejor.2008.08.023.
[57] H.T. Ünal, F. Başçiftçi, Using evolutionary algorithms for the scheduling of aircrew on airborne early warning and control system, Def. Sci. J. 70 (3) (2020) 240–248, https://doi.org/10.14429/dsj.70.15055.
[58] R.M. Van Slyke, R. Wets, L-shaped linear programs with applications to optimal control and stochastic programming, SIAM J. Appl. Math. 17 (4) (1969) 638–663, https://doi.org/10.1137/0117061.
[59] S.A. Vásquez, G. Angulo, M.A. Klapp, An exact solution method for the tsp with drone based on decomposition, Comput. Oper. Res. 127 (2021) 105127, https://doi.org/10.1016/j.cor.2020.105127.
[60] C.R. Vela, R. Varela, M.A. Gonzalez, Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times, J. Heuristics 16 (2) (2010) 139–165, https://doi.org/10.1007/s10732-008-9094-y.
[61] B. Verweij, S. Ahmed, A.J. Kleywegt, G. Nemhauser, A. Shapiro, The sample average approximation method applied to stochastic routing problems: a computational study, Comput. Optim. Appl. 24 (2003) 289–333, https://doi.org/10.1023/A:1021814225969.
[62] K.J. Wang, S.M. Wang, J.C. Chen, A resource portfolio planning model using sampling-based stochastic programming and genetic algorithm, Eur. J. Oper. Res. 184 (1) (2008) 327–340, https://doi.org/10.1016/j.ejor.2006.10.037.
[63] J.P. Watson, D.L. Woodruff, Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems, Comput. Manag. Sci. 8 (2011) 355–370, https://doi.org/10.1007/s10287-010-0125-4.
[64] J.P. Watson, S. Rana, L.D. Whitley, A.E. Howe, The impact of approximate evaluation on the performance of search algorithms for warehouse scheduling, J. Sched. 2 (2) (1999) 79–98.
[65] T. Yamada, R. Nakano, A genetic algorithm applicable to large-scale job-shop problems, Paper presented at the Parallel Problem Solving from Nature 2, Belgium, 28–30 September, 1992, pp. 281–290.
[66] Y. Yoshitomi, R. Yamaguchi, A genetic algorithm and the Monte Carlo method for stochastic job-shop scheduling, Int. Trans. Oper. Res. 10 (6) (2003) 577–596, https://doi.org/10.1111/1475-3995.00429.
[67] M.T. Younis, S. Yang, Hybrid meta-heuristic algorithms for independent job scheduling in grid computing, Appl. Soft Comput. 72 (2018) 498–517, https://doi.org/10.1016/j.asoc.2018.05.032.
[68] F. Zaman, S. Elsayed, R. Sarker, C. Essam, C.A. Coello Coello, An evolutionary approach for resource constrained project scheduling with uncertain changes, Comput. Oper. Res. 125 (2021) 105104, https://doi.org/10.1016/j.cor.2020.105104.

[69] M. Zhang, Y. Hong, N. Balakrishnan, An algorithm for computing the distribution function of the generalized Poisson binomial distribution, J. Stat. Comput. Simul. 88 (8) (2018) 1515–1527, https://doi.org/10.1080/00949655.2018.1440294.

[70] L. Zhen, Tactical berth allocation under uncertainty, Eur. J. Oper. Res. 247 (3) (1981) 928–944, https://doi.org/10.1016/j.ejor.2015.05.079.

[71] F. Zheng, X. Man, F. Chu, M. Liu, C. Chu, A two-stage stochastic programming for single yard crane scheduling with uncertain release times of retrieval tasks, Int. J. Prod. Res. 57 (13) (2019) 4132–4147, https://doi.org/10.1080/00207543.2018.1516903.