Flexible Services and Manufacturing Journal
https://doi.org/10.1007/s10696-019-09357-4

Integrated preventive maintenance and flow shop scheduling under uncertainty

Javad Seif[1,3] · Mohammad Dehghanimohammadabadi[2] · Andrew Junfang Yu[1]

Abstract

This paper is concerned with stochastic scheduling of production and maintenance activities in a permutation flow shop setting. We present a two-stage stochastic mixed-integer program (SMIP) that adapts the conventional permutation flow shop scheduling problem for incorporating multiple preventive maintenance activities with various meter-based intervals. The model handles uncertainties in both processing times and the duration of maintenance activities. The concept of combining maintenance activities in scheduling problems is introduced and formulated, along with other practical considerations. The objective is to minimize the total expected cost associated with lateness penalties and maintenance resources. We use simulation–optimization (SO) for solving large-scale instances of the problem, and for validating the SMIP model. Through extensive computational experiments, we show that the SO method is superior in terms of efficiency and effectiveness and evaluate its sensitivity to the input data. Finally, a case study in earth-moving operations is presented, followed by managerial implications.

Keywords Preventive maintenance · Flow shop scheduling · Stochastic mixed-integer programming (SMIP) · Simulation–optimization · Genetic algorithms · Construction equipment

✉ Andrew Junfang Yu
ajyu@utk.edu

Javad Seif
jseif@utk.edu

Mohammad Dehghanimohammadabadi
m.dehghani@northeastern.edu
http://www.mie.neu.edu/people/dehghani-mohammad

[1] Department of Industrial and Systems Engineering, The University of Tennessee-Knoxville, Knoxville, TN, USA

[2] Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, USA

[3] Reliability and Maintainability Center, Tickle College of Engineering, The University of Tennessee-Knoxville, Knoxville, TN, USA

Published online: 01 June 2019

1 Introduction

This study introduces an integrated stochastic operations and maintenance scheduling problem with a wide range of applications in industries from classic manufacturing settings to construction projects. In the conventional scheduling problems, it is assumed that the machines can continuously process the jobs (Pinedo 2012) and the information is complete and certain. However, in practice the machines must stop for a preventive or corrective maintenance, and the information available to the planners can be both incomplete and uncertain (Berry 1993). Ortiz and Diaz (2018) conducted a literature review specifically on non-deterministic duration of activities in project scheduling. The integration of maintenance and production scheduling has appeared in the literature in the last two decades (Xu et al. 2015; Yu and Seif 2016). Although incorporating maintenance decisions into production scheduling problems increases the practicality of existing models and solution methods, it also increases their complexity, and considering uncertainty makes the problem even more complex. This paper introduces an integrated preventive maintenance and flow shop scheduling problem under uncertainty as a stochastic mixed integer program (SMIP). Simulation–optimization (SO) is also employed as an alternative solution approach that is capable of handling the computational complexity of the problem.

Flow shop scheduling has been studied by many researchers after Johnson introduced the problem for two machines in 1954 (Johnson 1954), yet the number of studies in this area has been growing in time (Rossit et al. 2018; Yenisey and Yagmahan 2014). The main goal in the flow shop scheduling is to find a sequence for n jobs that are to be processed by m machines to optimize an objective function. Minimizing the completion time of the very last job (the makespan), the overall completion time, and the tardiness of the jobs are some examples of such an objective. Maintenance costs cover a big percentage of the total operating costs (Ángel-Bello et al. 2011; Yip et al. 2014). Therefore, it is reasonable to include minimizing the maintenance cost in the objective function. We consider minimization of the costs associated with both tardiness and maintenance activities.

Yoo and Lee (2016) classify scheduling problems with maintenance activities (MAs) incorporated, as *fixed* and *coordinated*. The first class of problems, *scheduling with machine availability constraints,* considers maintenance as a constraint not a decision. For instance, Choi et al. (2010) consider a number of maintenance periods in ordered and proportionate flow shop scheduling. They assume that these maintenance periods have been scheduled in advance with known start and finish times. Therefore, the maintenance schedule is incorporated to their model as a constraint not a decision. Other researchers use the same approach to model machine unavailability in a two-machine flow shop scheduling (Cheng and Wang 1999, 2000; Kubiak et al. 2002; Kubzin et al. 2009; Lee 1997, 1999).

In the second class of problems, scheduling of maintenance and job processing are considered simultaneously. Aggoune (2004) is one of the few papers that studies the coordinated variant of flow shop scheduling, while allowing a decision for performing maintenance within a time window. Performing maintenance

in a time window has been modeled in different types of scheduling problems, yet they are not precise in timing of MAs. Bock et al. (2012) study the computational complexity of single machine scheduling problems where each machine has a maintenance level and processing of the jobs deteriorates it. An MA needs to be performed in order to restore or increase the maintenance level before it becomes negative. Seif et al. (2018) and Yu and Seif (2016) adapt the concept of maintenance level for flow shop scheduling when multiple types of maintenance levels are involved. In this paper, we formulate and solve a permutation flow shop scheduling problem under uncertainty and incorporate the concept of combining different types of MAs in the problem.

Knezevic (1997) classifies maintenance tasks as *simultaneous*, *sequential*, or *combined*. A simultaneous task is composed of activities that are mutually independent and can be performed concurrently. A sequential task includes mutually independent activities that are performed in a predetermined order. A combined task includes some activities that can be performed simultaneously, and some activities that are sequential. Therefore, a combined maintenance task is a generalization of the other two types. To prevent any confusion, by MA, we refer to a set of tasks that have a common meter-based periodic interval. Two different MAs may have some similar tasks, so the combination of two or more MAs could prevent the repetition of the common tasks. We model the concept of combined maintenance activities in scheduling problems for the first time. In doing so, we cover all of the possible scenarios. The case study in Sect. 5 shows a practical example for better understanding of the problem.

Over the past few years, the importance of considering uncertainty in the scheduling problems has been highlighted by researchers and industrial practitioners; however, the methods used to deal with uncertainty do not seem to be very effective (Zheng et al. 2015). Gourgand et al. (2000) and González-Neira et al. (2017) conduct comprehensive reviews of research papers that involve flow shop scheduling under uncertainty. The latter review found that most of the papers in the literature consider processing times as a stochastic parameter, but maintenance has never been included as a stochastic process; while *Mean Time to Repair (MTTR)*, which is a well-known term in the maintenance and reliability literature (Ben-Daya et al. 2009; Hastings 2009) implies uncertainity in the durations of MAs.

González-Neira et al. (2017) report the superiority of the stochastic optimization approach in modeling uncertainty. They also mention stochastic programming and simulation–optimization as the most promising methods in stochastic flow shop scheduling. We consider two stochastic parameters to capture the uncertainties: (i) the processing times of production jobs, and (ii) the durations of MAs. We apply both stochastic programming and simulation–optimization to model and solve the problem. The use of both methods facilitates their validation, and performance evaluation in computational experiments.

Table 1 identifies the gap in the literature by comparing this paper with the most relevant publications. The contributions and features of this paper are:

- the conventional permutation flow shop scheduling problem is extended to a stochastic flow shop scheduling integrated with maintenance planning,

Table 1 Comparing this paper with the most relevant publications

Publication	Flow shop scheduling	Uncertainty	Maintenance cost included in OF	Integrated production and maintenance scheduling	Various types of maintenance	Maintenance-dependent processing times	Combined maintenance activities
Bertolini et al. (2019)				✓			
Seif et al. (2018)	✓		✓	✓	✓	✓	
Yu and Seif (2016)	✓		✓	✓	✓		
Zheng et al. (2015)	✓	✓					
Gourgand et al. (2000)	✓	✓					
Aggoune (2004)	✓ (m = 2)	✓		✓			
Bock et al. (2012)				✓			
Lee and Leon (2001)				✓		✓	
Allaoui and Artiba (2004)	✓	✓					
Aghezzaf et al. (2007)			✓				
Dekker (1995)		✓	✓				✓
Dekkert et al. (1991)		✓	✓				✓
Kroep (2017)			✓				✓
This paper	✓	✓	✓	✓	✓	✓	✓

- the concepts of combining MAs and prolonged (maintenance-dependent) processing times are introduced and incorporated to the model in order to increase its practicality,
- two methodologies (simulation–optimization and stochastic programming) are developed and contrasted for validating the results and evaluating their performances.

The rest of this paper is laid out as follows. First, we formulate the problem as a two-stage *Stochastic Mixed-Integer Program (SMIP)* in Sect. 2. In Sect. 3, we present a SO algorithm as an alternative approach for modeling and solving the problem, which also allows the validation of the SMIP. In Sect. 4, we will evaluate and report in details the performance of the SO in comparison with one of the commercial solvers through extensive computational experiments. In Sect. 5, the case study is presented. Conclusions and remarks along with directions for future research are discussed in Sect. 6.

2 Problem definition and mathematical formulation

In this section, we formulate a mixed-integer stochastic program to define the problem. To do this, first, we define and formulate the concepts of combining maintenance activities and discuss the prolonged processing times. The SMIP model is presented as the last part of this section.

2.1 Combined maintenance activities

Maintenance activities are made up of various tasks or steps, and it is not uncommon that two different maintenance activities share certain tasks. For example, "oil change" and "oil filter change" of a vehicle are two different preventive maintenance activities that share certain tasks such as "driving the vehicle onto two ramps", or "locating and unscrewing the oil drain plug." For such activities it may be optimal (in terms of time or cost) to combine and perform the activities together, which explains why the oil change and oil filter change are usually executed as one *combined* activity. Figure 1 illustrates a general case in which Activities 1 and 2 share Tasks A and C. We can see that the combined activity is shorter than the sum of two separate activities because Tasks A and C are performed only once. The case study in Sect. 5 includes more examples of preventive maintenance activities with shared tasks.

We will incorporate the concept of combined maintenance activities into the mixed-integer program that will be presented later, as follows. For l maintenance activities, there exists $o = 2^l$ possible combinations. Note that one of these combinations includes no maintenance activity; l of them include only one activity, $\binom{l}{2}$ of them include only

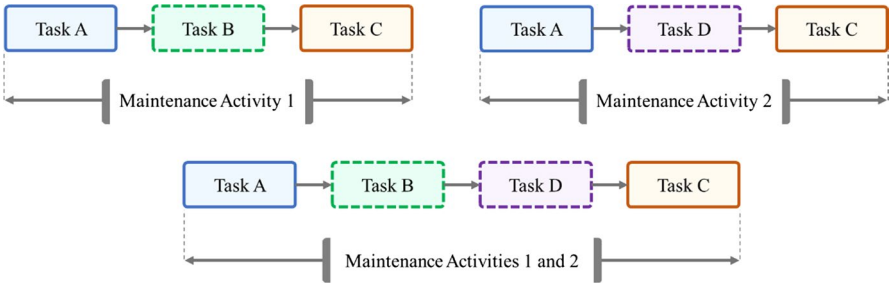

Fig. 1 Maintainability block diagram for combined maintenance activities

two activities, $\binom{l}{n}$ of them include only n activities ($n < l$), and finally, one of them comprises all l activities. The time and cost associated with each one of these combinations is less than or equal to the sum of the time and cost of the activities included in that combination. If the activities in a combination do not share any tasks, the time and cost of performing them separately will be equal to the case where they are performed consecutively (combined).

Consider a job to be processed on a specific machine in a flow shop. The binary variables $y_k \in \{0,1\},\ k = 1,\ldots,l$ determine whether the maintenance activity type k is scheduled before processing the job. The binary variables $\phi_r \in \{0,1\},\ r = 1,\ldots,o$ determine whether combination r is forming in that position, which is a function of $y_k,\ k = 1,\ldots,l$. The following equation will map the binary decision variables corresponding to individual maintenance activities that can be performed before a job ($y_k,\ k = 1,\ldots,l$) to binary decision variables corresponding to the combinations of maintenance activities ($\phi_r,\ r = 1,\ldots,o$).

$$\sum_{r=1}^{o=2^l} b_r \phi_r = \sum_{k=1}^{l} a_k y_k, \quad \phi_r, y_k \in \{0,1\},\ \forall k = 1,\ldots,l \tag{1}$$

where the coefficients $a_k,\ k = 1,\ldots,l$ and $b_r,\ r = 1,\ldots,o$ are chosen such that only one of the combinations (variables) in the left-hand side is chosen (takes the value 1). But there is no limitation on the number of non-zero binary variables on the right-hand side. In other words, none, one, or more than one of the variables on the right-hand side can take the value 1 (we can decide to perform none, one, or more than one type of maintenance activity before processing a certain job on a certain machine), but only one of the variables on the left-hand side must take the value 1 (only one combination can represent the chosen set of maintenance activities). "Appendix" includes a method for generating the coefficients.

2.2 Prolonged processing times

An time-based maintenance activity allows a machine to operate only for certain number of hours, called maintenance interval. As soon as the machine's cumulative operating times equals the maintenance interval, it must be stopped for performing the maintenance activity. Let r_k and R_k be the residual operating time and the age-based interval of the kth maintenance activity of a machine, respectively. The variable r_k equals R_k after the maintenance activity type k is performed on the machine and it approaches 0 as the machine processes production jobs. We expect that in practice the processing time of a job prolongs as the residual operating times approach 0. This is because: (1) machine performance declines as it gets closer to its maintenance due date, which can lead to a slower processing, and (2) when both tardiness and maintenance costs exist in a minimization objective function, it motivates the solution algorithms to schedule maintenance activities as early as possible to reduce the risk of failures, but not too early that causes excessive tardiness and maintenance costs. Here, we adapt the model proposed by Seif et al. (2018) for prolonged processing times.

The value $f_k = r_k/R_k$ represents the remaining/residual operating time as a fraction of the respective maintenance interval. Obviously, $0 \leq f_k \leq 1$, $\forall k = 1, \ldots, l$, and $0 \leq F \leq 1$ where $F = \sum_{k=1}^{l} f_k/l$ is the average of fractional residual operating times and represents the machine's health. The prolonged processing time of a job, ρ, is defined as

$$\rho = \begin{cases} \lambda^1 p, & A < F \leq 1 \\ \lambda^2 p, & B < F \leq A \\ \lambda^3 p, & 0 < F \leq B \end{cases}, \quad 0 \leq B \leq A \leq 1 \leq \lambda^1 \leq \lambda^2 \leq \lambda^3 \tag{2}$$

where p is the nominal processing time of the job. If the machine's health, F, falls between the constant A and 1, the nominal processing time is multiplied by the coefficient λ^1 which can be greater than or equal to 1 with the potential to prolong the processing time. If the machine's health is between A and B, processing time is multiplied by the coefficient λ^2 which can be greater than λ^1, and prolong the processing time, and if it is between 0 and B, multiplied by λ^3 which can be greater than λ^2 prolonging the processing time even more. Here we are considering a special case in which the health of a machine has only three states. The generalized form will be considered in the SMIP formulation.

An illustrative example shown in Fig. 2 summarizes the problem definition. This example involves a single machine (for simplicity), two types of maintenance activities (MA_1 and MA_2, with time intervals R_1 and R_2), three production jobs (J_1, J_2, and J_3), and two probabilistic scenarios. The only difference in the data of these two scenarios is the processing time of J_1. We also assume that in Scenario 1, we are not allowed to combine the MAs. In Scenario 1, MA_1 is required after J_1 because otherwise we will miss the due time of MA_1 ($R_1 < 0$) while processing J_2. We can also see that processing time of J_2 is shorter compared to Scenario 2 because MA_1 was just executed. Note that processing time of J_3 is the same in both scenarios for the same reason (neither one has

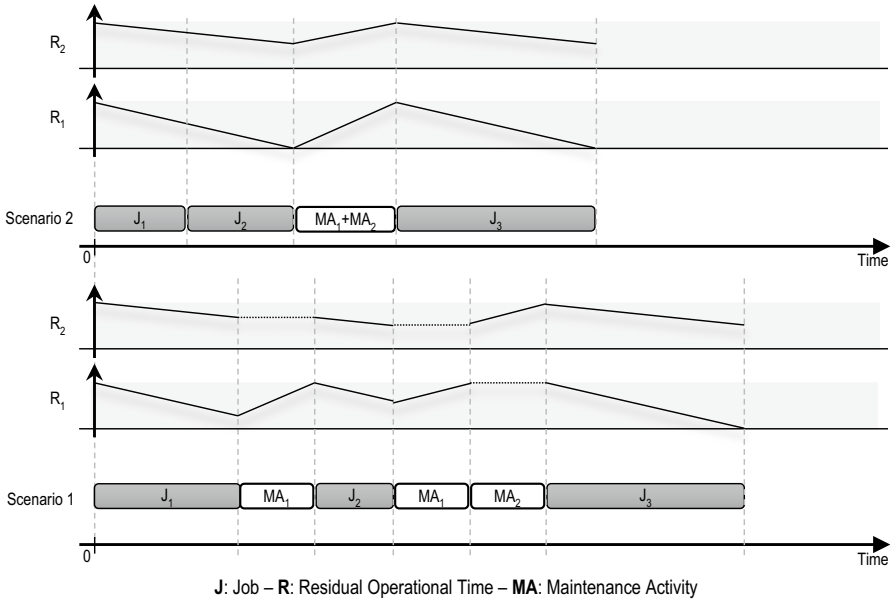

Fig. 2 An illustrative example of the problem (single machine)

been prolonged). Finally, it takes less time to perform MA_1 and MA_2 in a row when combining them is allowed (in Scenario 2).

2.3 The mixed-integer stochastic program (SMIP)

We model a permutation flow shop scheduling problem in which a number of jobs will be processed by a series of machines in the same order. All machines have a number of meter-based maintenance activities. The MAs take place only after a machine completes processing of a job and before starts processing the next job (preemption is not allowed). The residual operating time of a machine with respect to any of the MAs linearly decreases as the machine processes the jobs. The residual operating times cannot be negative. Therefore, an MA must be scheduled before processing a job, if the respective residual operating time is going to become negative while processing the job. The durations of MAs and the processing times of jobs are all uncertain and follow certain probability distributions. There are several scenarios in each of which the MA durations and processing times are sampled from respective distributions. Under each scenario, the maintenance durations can change if the MAs are combined.

2.3.1 Notations

Let *m, n, l,* and *S* be the number of machines, jobs, types of maintenance activities, and scenarios, respectively. The following indices, parameters, and variables are used in the formulation of the problem.

Indices

i Represents machines where $i = 1, \ldots, m$.
j Represents individual production jobs where $j = 1, \ldots, n$.
q Represents job positions in a sequence where $q = 1, \ldots, n$.
k Represents a certain type of maintenance where $k = 1, \ldots, l$.
h Represents the health state of a machine $h = 1, \ldots, H$.
r Represents maintenance combinations where $r = 1,2, \ldots, o$.
s Represents a specific scenario where $s = 1, \ldots, S$.

Parameters (Input Data)

p_{ij}^{s} Nominal processing time of job j on machine i under scenario s.
e_{ik}^{s} Nominal duration of MA type k on machine i under scenario s.
e_{ir}^{s} Duration of MA combination type r on machine i under scenario s.
$R_{i,k}$ Frequency of time-based preventive maintenance activity type k for machine i.
SP_{ik} Cost of required spare parts and materials for MA type k on machine i.
$SP_{ik}^{'}$ Cost of required spare parts and materials for MA combination type r on machine i.
WF Cost of skilled workforce per time unit for performing maintenance activities.
d_j The due date of job j.
π_j Penalty cost associated with each time unit delay in completion of job j.
λ^{h} The coefficient which is multiplied by the nominal processing times of the jobs to prolong them, when a machine is in the health state h.
$\Pr(s)$ Probability of scenario s being realized.
K A sufficiently large number.

Decision Variables

Z_s Total cost, the value of the objective function under scenario s.
x_{jq} First-stage decision variable that takes the value 1 if job j is processed as the qth job in the sequence, and 0 otherwise.
y_{iqk} First-stage binary decision variable that takes the value 1 if MA type k is scheduled for machine i before processing the qth job, and 0 otherwise.
ϕ_{iqr} First-stage binary decision variable that takes the value 1 if MA combination type r is scheduled for machine i before processing the qth job, and 0 otherwise.
r_{iq}^{ks} Residual operating time with respect to the MA type k of machine i before processing the qth job, under scenario s.
ST_{iq}^{s} Start time of the qth job on machine i, under scenario s.
FT_{iq}^{s} Finish time of the qth job on machine i, under scenario s.
t_q^{s} Tardiness of the qth job, under scenario s.
ρ_{iq}^{s} Processing time of the qth job on machine i, under scenario s.

γ_{ijq}^{s} Processing time of job j on machine i when it is processed as the qth job, under scenario s.

Π_{jq}^{s} Penalty (cost) associated with job j as a result of it being processes as the qth job, under scenario s.

Λ_{iq}^{hs} 1 if machine i is in the health state h before processing the qth job, under scenario s, and 0 otherwise.

u_{ijq}^{hs} 1 if machine i is in the health state h before processing job j when it is processed as the qth job, under scenario s, and 0 otherwise.

2.3.2 The model

The objective function (OF) of the model is to minimize the total *expected* cost which comprises the penalty cost incurred because of lateness in completion of the jobs (tardiness), and the maintenance cost (spare parts and man-hours).

$$\text{minimize } E\left[Z_s\right] = \sum_{s=1}^{S} \Pr(s)\left[\sum_{j=1}^{n}\sum_{q=1}^{n} \Pi_{jq}^{s} + \sum_{i=1}^{m}\sum_{q=1}^{n}\sum_{r=1}^{o=2^{l}-1} \phi_{iqr}\left(SP_{ir}^{'} + e_{ir}^{'s} WF\right)\right], \tag{3}$$

Subject to:

$$\sum_{q=1}^{n} x_{jq} = 1, \quad j = 1, \ldots, n \tag{4}$$

$$\sum_{j=1}^{n} x_{jq} = 1, \quad q = 1, \ldots, n \tag{5}$$

$$ST_{11}^{s} = 0, \quad s = 1, \ldots, S \tag{6}$$

$$ST_{i1}^{s} = \sum_{i'=1}^{i-1} \rho_{i'1}^{s}, \quad i = 2, \ldots, m,\ s = 1, \ldots, S \tag{7}$$

$$ST_{1q}^{s} = FT_{1(q-1)}^{s} + \sum_{r=1}^{o} \phi_{1qr} e_{1r}^{'s}, \quad q = 2, \ldots, n,\ s = 1, \ldots, S \tag{8}$$

$$ST_{iq}^{s} \geq FT_{i(q-1)}^{s} + \sum_{r=1}^{o} \phi_{iqr} e_{1r}^{'s}, \quad i = 2, \ldots, m,\ q = 2, \ldots, n,\ s = 1, \ldots, S \tag{9}$$

$$ST_{iq}^{s} \geq FT_{(i-1)q}^{s}, \quad i = 2, \ldots, m,\ q = 2, \ldots, n,\ s = 1, \ldots, S \tag{10}$$

$$FT_{iq}^{s} = ST_{iq}^{s} + \rho_{iq}^{s}, \quad i = 1, \ldots, m,\ q = 1, \ldots, n,\ s = 1, \ldots, S \tag{11}$$

$$r_{i1}^{ks} = R_{i,k}, \quad i = 1, \ldots, m,\ s = 1, \ldots, S,\ k = 1, \ldots, l \tag{12}$$

$$r_{iq}^{ks} \geq \sum_{j=1}^{n} \gamma_{ijq}^{s}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n,\ k = 1, \ldots, l \tag{13}$$

$$r_{iq}^{ks} \geq r_{i(q-1)}^{ks} - \sum_{j=1}^{n} \gamma_{ij(q-1)}^{s} - y_{iqk}K, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 2, \ldots, n,\ k = 1, \ldots, l \tag{14}$$

$$r_{iq}^{ks} \leq r_{i(q-1)}^{ks} - \sum_{j=1}^{n} \gamma_{ij(q-1)}^{s} + y_{iqk}K, \quad s = 1, \ldots, S, i = 1, \ldots, m, q = 2, \ldots, n, k = 1, \ldots, l \tag{15}$$

$$r_{iq}^{ks} \geq R_{i,k} - K\left(1 - y_{iqk}\right), \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 2, \ldots, n,\ k = 1, \ldots, l \tag{16}$$

$$r_{iq}^{ks} \leq R_{i,k} + K\left(1 - y_{iqk}\right), \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 2, \ldots, n,\ k = 1, \ldots, l \tag{17}$$

$$t_{q}^{s} \geq FT_{mq}^{s} - \sum_{j=1}^{n} x_{jq}d_{j}, \quad s = 1, \ldots, S,\ q = 1, \ldots, n \tag{18}$$

$$\Pi_{jq}^{s} - \pi_{j}t_{q}^{s} \geq -K\left(1 - x_{jq}\right), \quad s = 1, \ldots, S,\ j = 1, \ldots, n,\ q = 1, \ldots, n \tag{19}$$

$$\Pi_{jq}^{s} - \pi_{j}t_{q}^{s} \leq K\left(1 - x_{jq}\right), \quad s = 1, \ldots, S,\ j = 1, \ldots, n,\ q = 1, \ldots, n \tag{20}$$

$$\Pi_{jq}^{s} \geq -Kx_{jq}, \quad s = 1, \ldots, S,\ j = 1, \ldots, n,\ q = 1, \ldots, n \tag{21}$$

$$\Pi_{jq}^{s} \leq Kx_{jq}, \quad s = 1, \ldots, S,\ j = 1, \ldots, n,\ q = 1, \ldots, n \tag{22}$$

$$\rho_{iq}^{s} = \sum_{j=1}^{n} \sum_{h=1}^{H} u_{ijq}^{hs} \lambda^{h} p_{ij}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n \tag{23}$$

$$u_{ijq}^{hs} \leq \Lambda_{iq}^{hs}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ q = 1, \ldots, n,\ h = 1, \ldots, H \tag{24}$$

$$u_{ijq}^{hs} \leq x_{jq}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ q = 1, \ldots, n,\ h = 1, \ldots, H \tag{25}$$

$$u_{ijq}^{hs} \geq x_{j\mathrm{q}} + \Lambda_{iq}^{hs} - 1, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ q = 1, \ldots, n,\ h = 1, \ldots, H \tag{26}$$

$$\sum_{h=1}^{H} \Lambda_{iq}^{hs} = 1, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n \tag{27}$$

$$\sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} > \frac{H-1}{l} \Lambda_{iq}^{1s} - K \sum_{h=2}^{H} \Lambda_{iq}^{hs}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n \tag{28}$$

$$\sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} \leq \frac{h}{l} \Lambda_{iq}^{hs} + K \left(\sum_{h'=1}^{H} \Lambda_{iq}^{h's} - \Lambda_{iq}^{hs} \right), \quad h = 2, \ldots, H-1, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n \tag{29}$$

$$\sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} \geq \frac{h-1}{l} \Lambda_{iq}^{hs} - K \left(\sum_{h'=1}^{H} \Lambda_{iq}^{h's} - \Lambda_{iq}^{hs} \right), \quad h = 2, \ldots, H-1, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n \tag{30}$$

$$\sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} < \frac{1}{l} \Lambda_{iq}^{Hs} + K \sum_{h=1}^{H-1} \Lambda_{iq}^{hs}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ q = 1, \ldots, n \tag{31}$$

$$\gamma_{ijq}^{s} \leq x_{jq} K, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ q = 1, \ldots, n \tag{32}$$

$$\gamma_{ijq}^{s} \leq \rho_{iq}^{s}, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ q = 1, \ldots, n \tag{33}$$

$$\gamma_{ijq}^{s} \geq \rho_{iq}^{s} + \left(x_{jq} - 1\right) K, \quad s = 1, \ldots, S,\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ q = 1, \ldots, n, \tag{34}$$

$$\sum_{r=1}^{o} b_r \phi_{iqr} = \sum_{k=1}^{l} a_k y_{iqk}, \quad i = 1, \ldots, m,\ q = 1, \ldots, n \tag{35}$$

$$x_{jq}, y_{iqk}, \Lambda_{iq}^{h}, u_{ijq}^{hs}, \phi_{iqr} \in \{0, 1\}, \quad s = 1, \ldots, S,\ j = 1, \ldots, n, \quad q = 1, \ldots, n,\ i = 1, \ldots, m,\ h = 1, \ldots, H \tag{36}$$

$$r_{iq}^{ks}, t_{q}^{s}, \Pi_{jq}^{s}, ST_{iq}^{s}, FT_{iq}^{\mathrm{s}}, \rho_{iq}^{s}, \gamma_{ijq}^{s} \geq 0, \quad s = 1, \ldots, S,\ j = 1, \ldots, n, \quad q = 1, \ldots, n,\ i = 1, \ldots, m,\ h = 1, \ldots, H \tag{37}$$

The OF in Eq. (3) is comprised of two parts; the penalty cost, and the maintenance cost. The penalty cost for each job is presented as a variable that has two indices, j and q. For each job j, the variable is set equal to 0 in Constraints (21) and (22) if it does not occupy the position q in the sequence of the jobs. Otherwise, it is set equal to the penalty cost for job j times the tardiness value, as expressed in Constraints (19) and (20). The maintenance cost itself is comprised of two parts; the cost of spare parts and the cost of the cost of workforce. The binary variable ϕ_{iqr} determines whether maintenance combination r is scheduled before processing the qth job on machine i. It is multiplied by the cost of spare part for that combination plus the unit cost of workforce times the duration of that combination. The MAs are considered individually in the scheduling process. Constraint (35), which is a generalization of Eq. (1), chooses the combination that correctly represents the scheduled MAs.

Constraints (5) and (6) ensure that each job is assigned to only one position in the sequence of the jobs, and each position is filled by only one job. These two constraint sets are only concerned with the first stage variables. However, the rest of the constraints involve at least one second-stage variable with s in their superscripts. Therefore, the rest of the constraints must be feasible for every scenario, otherwise the solution is infeasible.

Constraints (6–11) together determine the start and finish time of the jobs on every machine. The Start Time (ST) and Finish Time (FT) variables are first calculated for the first job in the sequence and the first machine in the flow shop, and then they will be calculated for the rest of the jobs and machines. Constraint (6) sets 0 as the ST of the first job on the first machine. Constraint (7) sets the ST of the first job on each machine equal to the sum of its processing times on the previous machines. Constraint (8) sets the ST of each job on the first machine equal to the finish time of the previous job on the first machine, plus the duration of MAs. As was already explained, instead of nominal durations of the individual MAs, the duration of maintenance combinations is considered in timings. Constraints (9) and (10) are the linear form of $ST^s_{iq} = \max\left(FT^s_{i(q-1)} + \sum_{r=1}^{o} \phi_{iqr} e'^s_{1r}, FT^s_{(i-1)q}\right)$; the ST of every job on a machine is equal to the maximum of its FT on the previous machine plus the maintenance time, and the FT of the previous job on the machine. Constraint (11) sets the FT of all jobs on every machine equal to the respective ST plus the processing time of the job.

Constraint (12) sets the residual operating times of all machines equal to the maintenance interval, before processing the first job. Constraint (13) sets the residual operating time of machine i before processing the qth job to be greater than or equal to the time it takes to process the job. This constraint ensures that the machines do not operate while their maintenance requirements are overdue. Constraints (14–17) are the linearizes form of the following equation.

$$r^{ks}_{iq} = \begin{cases} r^{ks}_{i(q-1)} - \sum_{j=1}^{n} \gamma^s_{ij(q-1)}, & y_{iqk} = 0 \\ R_{i,k}, & y_{iqk} = 1 \end{cases}, \quad \forall s, i, q, k \tag{38}$$

The residual operating time of a machine before processing a job with respect to MA k is equal to the respective maintenance interval, if the MA is performed, and otherwise it is equal to its value before processing the previous job minus the processing time of the previous job. Tardiness for each job in the sequence is calculated in Constraint (18). The prolonged processing times of the jobs are calculated as expressed in Constraint (23); the nominal processing time of job j on machine i times the coefficient of state h, times the binary variable u_{ijq}^{hs} that takes the value 1 if machine i is in state h before processing the qth job and if the qth job is job j, and 0 oherwise. Constraints (24–26) are the linearization form of $u_{ijq}^{hs} = \Lambda_{iq}^{hs} x_{jq}$.

Constraint (27) ensures that machine i is in only in one of the predefined health states before processing the qth job. Constraints (28–31) are the generalized and linearized form of the following equations when the machines have only three states ($H = 3$).

$$\Lambda_{iq}^{1s} = \begin{cases} 1, \ \sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} > 0.66 \\ 0, \ otherwise \end{cases}, \quad \forall s, i, q \tag{39}$$

$$\Lambda_{iq}^{2s} = \begin{cases} 1, \ 0.33 \leq \sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} \leq 0.66 \\ 0, \ otherwise \end{cases}, \quad \forall s, i, q \tag{40}$$

$$\Lambda_{iq}^{3s} = \begin{cases} 1, \ \sum_{k=1}^{l} \frac{r_{iq}^{ks}}{l \cdot R_{i,k}} < 0.33 \\ 0, \ otherwise \end{cases}, \quad \forall s, i, q \tag{41}$$

Constraints (32–34) are the linearized form of $\gamma_{ijq}^{s} = x_{jq} \rho_{iq}^{s}$ for obtaining the actual processing time of job j on machine i, if it is scheduled as the qth job. Constraint (35) ensures that the correct maintenance combination is chosen based on the scheduling of MAs. The one that corresponds to the set of maintenance activities that are decided to be performed. This is a generalization of Eq. (1) for flow shop scheduling with multiple types of MAs. Constraints (36–37) ensure that all the decision variables are within their bounds.

3 Simulation optimization

Simulation–optimization (SO) is a promising avenue of research to tackle stochastic problems with uncertain parameters (Dehghanimohammadabadi 2016; Dehghanimohammadabadi et al. 2017). Among all the available options, we use a *simheuristic* approach to obtain desirable solutions with a reasonable

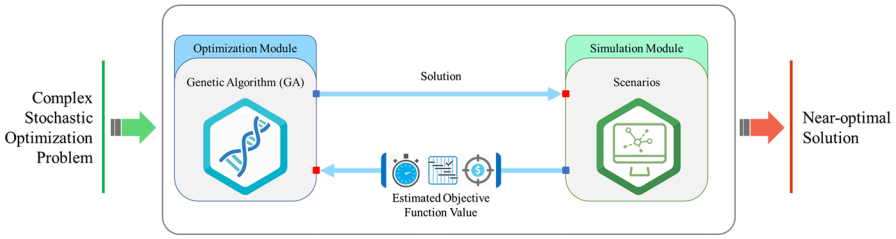

Fig. 3 Schematic framework of the Simheuristic approach

computational effort. A *simheuristic* approach is an integration of simulation and a metaheuristic-driven framework (Juan et al. 2015). Simheuristics are capable of solving complex stochastic combinatorial optimization problems, where the simulation component deals with the uncertainty of the model (Chica et al. 2017). This approach is particularly favorable when combination of discrete and continuous decision variables are present in the optimization problem (Jalali and Nieuwenhuyse 2015). The simheuristic model in this study is a combination of Genetic Algorithms (GA) and Monte Carlo Simulation (MCS) models. As shown in Fig. 3, the simulation module is a *cost function* for the solutions that are generated by the metaheuristic algorithm. Since the model is stochastic, in each iteration of GA, the generated solutions are evaluated in a simulated environment to calculate the expected value of the objective function (the cost). This procedure continues until GA meets the stopping criteria.

Following the SO strategies recommended by Barton and Meckesheimer (2006), the MCS is used to generate a number of possible scenarios based on the probability distributions of the stochastic parameters. The scenarios are inputs to the optimization module. Although each one of these scenarios represents a deterministic instance of the problem, in which the value of the stochastic parameters is certain, they collectively represent the stochastic nature of the problem.

As depicted in Fig. 4, after the scenarios are generated via the Monte-Carlo simulation, the optimization module generates a solution (X), recursively, and the objective function value is calculated for each scenario (Y_s). The expected value (average) will determine the ultimate value of the solution. The new solution is generated based on the internal operators and search methods of the specific meta-heuristic method that is being used. This cycle repeats until the optimization module satisfies some stopping criteria. Depending on the user's preferences, these criteria could be running model for a certain number of iterations or achieving a desirable performance measure (such as time).

We use Genetic Algorithm (GA) as the meta-heuristic technique. The goal of metaheuristics is to efficiently explore the search space in order to find (near-)optimal solutions (Blum and Roli 2003). Among all metaheuristic methods, we selected GA because GA has proved its robustness in searching complex solution spaces in many SO problems (Carson and Maria 1997), and it is a good alternative for solving real-time operations (Rani and Moreira 2010). GA is being successfully used as an optimization module in a wide range of SO problems, including scheduling

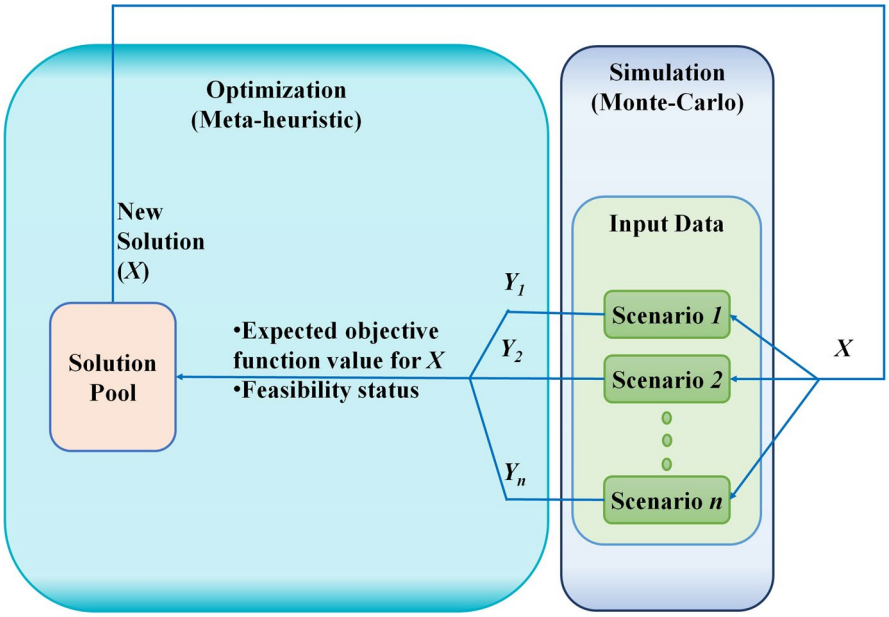

Fig. 4 The mechanism of the simulation–optimization method

(Dehghanimohammadabadi and Keyser 2017), supply chain design (Göçken et al. 2015; Truong and Azadivar 2003), and healthcare operations (Baesler and Sepúlveda 2001). Moreover, the parallel nature of GA allows for parallel processing, which is essential in saving computational time when solving complex stochastic problems (Rani and Moreira 2010).

Ortiz and Diaz (2018) found that GA was the main solution technique employed by researchers for project scheduling when the duration of activities is non-deterministic. Rossit et al. (2018) also reported GA as the single leading meta-heuristic method used for solving flow shop scheduling (followed by Tabu Search, Simulated Annealing, and Ant Colony Optimization algorithms). Lin et al. (2017) applied GA to a SO scheduling problem that was close to this work in that the processing times were uncertain. GA is an algorithm inspired by the basic mechanism of natural evolution, introduced by Holland (1975).

The GA procedure initializes from a randomly generated population of solutions, and evolves good local solutions by mimicking the process of natural selection using mechanisms such as mutation to generate variants and crossover to improve combinations (Trevino and Falciani 2006). GA is a population-based algorithm and employs random choices to have a highly exploitative search, keeping a balance between exploration of the feasible domain and exploitation of good solutions. Yu and Seif (2016) employed a factorial design of computational experiments for studying the parameters of a GA that was designed for solving a similar problem: a deterministic flow shop scheduling problem with multiple maintenance activities. In this work, we have tested and adapted the results of their study for tuning the parameters

Table 2 The values for the GA parameters

Parameters	Value
Initial population size	200
Crossover percentage	0.8
Mutation percentage	0.8
Mutation rate	0.03

Table 3 Solution represtaion of the proposed GA

Jobs	1	2	3	4
Jobs sequence	4	1	3	2
The required combination of MAs on:				
Machine 1	7	0[a]	4	0
Machine 2	0	0[a]	6	3
Machine 3	2	0[a]	0	5

[a]No MA is required prior to processing the first job in the sequence

of the GA, which has yielded similar results (see Sect. 4.1). The ultimate values of the GA parameters are listed in Table 2.

We use a new strategy to represent solutions generated by GA. This solution representation determines (i) the sequence of jobs in the flow shop system, and (ii) the number of MAs needed to be performed on each machine prior to each job by choosing one of the MA combinations. Table 3 shows the solution representation for a flow shop system with 4 jobs, 3 machines, and 3 MAs. The second row in the solution matrix indicates the sequence of jobs that go through processes in all machines (permutation flow shop). The numbers provided in the last three rows indicate the combination of MAs that needs to be performed on each machine before processing each job. We use integers 0–7 in Table 3 for representing $2^l = 2^3 = 8$ possible combinations of three MAs. Combination 0 includes no MA, and Combination 1, 2 and 3 include only the first, second and third MAs, respectively. Combination 4 includes the first and second MAs, Combination 5 includes the first and third MAs, Combination 6 includes the second and third MAs, and finally Combination 7 includes all three MAs.

The number of variables used to represent a solution for a given problem with m machines, n jobs, and l MAs is $(n + nm)$, which is independent of l. The chromosome representations proposed by Seif et al. (2018) and Yu and Seif (2016) for deterministic flow shop and maintenance scheduling has $(n + nml)$ integer variables. Therefore, the solution space created by their chromosome representation, unlike the one proposed in this paper, grows in size with the number of maintenance activities. We use real numbers for chromosome representation, as shown in Fig. 5. The chromosomes are then parsed to obtain the sequence of the jobs, and the required combination of MAs before processing each job on each machine. Without combining the maintenance activities we would need a binary variable that determines the execution of each maintenance activity.

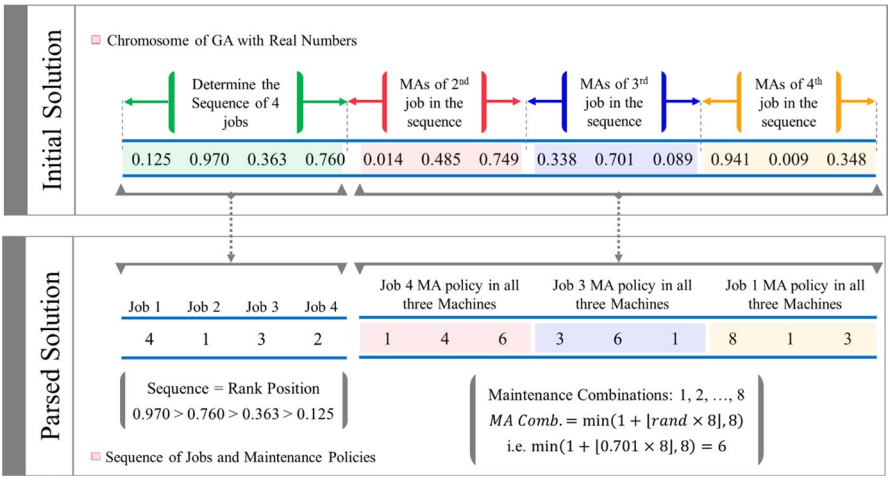

Fig. 5 Representation of GA chromosomes with real numbers

In Fig. 5, for 4 jobs, 3 machines, and 3 MAs, each chromosome is coded as an array of $(4 + (4 - 1) \times 3 = 13)$ real numbers. The rank position of the first four numbers indicate the sequence of the jobs. The rest of the numbers are converted to an integer value $(1, 2, \ldots, 8)$ to determine the combination of MAs that are required before processing each job on each machine, similar to Table 3. For instance, the generated real number for the MA policy of the 3rd job in the sequence (Job 3) on Machine 2 is 0.701. By mapping this number to the range of 1–8, combination 6 is obtained.

The only infeasibility that can occur in the algorithm is due to the violation of maintenance requirements. Feasibility of the solutions is checked by ensuring that the remaining time to the next maintenance activity (after processing each job) is a positive value. Each chromosome has an attribute which is an $m \times n \times l$ (three-dimensional) array. The value in the (i, q, k) cell of this array shows the remaining (residual) time of the preventive maintenance type k on machine i before processing the qth job. Presence of any negative values in the array implies failing to execute at least one of the preventive maintenance activities before it was due. In other words, the violation of Eq. (38). If this happens, a sufficiently large value is considered as the cost of such chromosome (solution), which makes it extremely unlikely to pass over. The rest of the variables are dependent on the job sequence and maintenance schedule. They are calculated for a given solution according to the presented model and will not cause any infeasibility.

4 Computational experiments

In this section, we present the computational experiments that we designed to tune the SO algorithm, and to evaluate the performances of the algorithm in comparison with a commercial solver. First, we introduce the methods used to generate test

problems for the experiments. Then, we discuss tuning of the algorithm. Finally, we present the main experiments that evaluate the SO algorithm by comparing its performance against a commercial solver. We coded the algorithm in MATLAB and used IBM ILOG CPLEX Optimization Studio (Version: 12.5.1.0) to solve the problem formulated in Eqs. (3–37) in Sect. 2. All algorithms and the CPLEX solver were run on an i7-3770 @ 3.40 gigahertz Intel processor with 8.00 gigabytes of system memory. Throughout the experiments, we solve 30 test problems with various settings. Solving 30 test problems allows us to draw reliable inferences about the performance of the solution methods.

While the metaheuristic component of a simheuristic method is responsible for exploring and exploiting the solution space, we depend on the simulation component for obtaining the objective function value. The simulated optimal objective function value can be far from the actual optimal value (Hong et al. 2015). This is because we are unable to test a solution in all the possible scenarios. However, our certainty increases with the number of scenarios (replications) used in the simulation model. We included 30 scenarios (replications) in each test problem. We chose this number because it is large enough to minimize noise in the objective function value (hence, maximize the reliability of optimality for a solution given by the metaheuristic component) and it is small enough for CPLEX to be able to at least find a feasible solution for the problem sizes used in computational experiments. These scenarios are generated via Monte-Carlo simulation. The comparison is between two different solution approaches, namely the exact solution methods built in the commercial solver and a simheuristic method. Both the solver and the simheuristic check the feasibility of a solution against all of the constraints defined in the stochastic program that was presented in Sect. 2.

4.1 Test problem generation

A test problem must contain the values of all the parameters introduced in Sect. 2. We used a test problem generator, which is a function with the following arguments: number of jobs (n), Due Date Tightness Factor ($DDTF$), and Maintenance Interval Factor (MIF). The values of the parameter in each test problem are generated as follows. Number of machines, $m = 3$, number of MAs, $l = 3$, the health states of the machines, $H = 3$, number of scenarios, $S = 30$, maintenance intervals, $R_1 = 4 \times MIF, R_2 = 5 \times MIF, R_3 = 6 \times MIF$, spare part costs, $SP_{ik} \sim U(\$150, \$450)$, the workforce cost, $WF = \$20/hour$, the due dates, $d_j \sim U\left(240, \left\lfloor \frac{240n}{DDFT} \right\rfloor\right)$, penalty costs, $\pi_j \sim U(10, 20)$, the probability of scenarios, $\Pr(s) = 1/S$, the coefficients for prolonging the processing times, $\lambda^1 = 1.0, \lambda^1 = 1.5, \lambda^1 = 2.0$, nominal processing times, $p^s_{ij} \sim TRI(20, 35, 70)$, and the nominal duration of MAs, $e^s_{ik} \sim TRI(5, 15, 25)$, where $\sim U(a, b)$ denotes a random value that follows the Uniform distribution in the range $[a, b]$, and $\sim TRI(a, b, c)$ denotes a random value that follows the Triangular distribution with a, b, and c as the minimum, most likely, and the maximum values that the random variable can take, respectively. All the test problems (as CPLEX files) can be retrieved online at https://www.dropbox.com/sh/wn7u776g4fwzrcn/AADC0LdeITjWuF9WWTZDUmdMa?dl=0.

Table 4 The impact of population size on the performance of the SO algorithm

Number of jobs (n)	Population size	Average improvement in the OFV (cost) (%)	Average increase in the solution time (%)
4	25	0	0
	50	7	77
	100	15	255
	200	22	546
	400	27	1215
6	25	0	0
	50	6	104
	100	18	399
	200	22	756
	400	17	1394
8	25	0	0
	50	10	131
	100	17	359
	200	22	764
	400	23	1720

Although these might be close to the values used in some of applications, the sole purpose of generating them within the presented bounds is to form test problems that are both feasible and challenging to solve. We will present a case study in Sect. 5 in which the values are chosen such that they represent an application that is similar to one of the real-world problems. We keep the number of machines, types of MAs, and the health state of the machines as constants in all of the test problems, but we will increase the number of jobs in order to test the performance of the algorithm in dealing with large-scale instances of the problem. In practice and for a particular application, the values that are fixed do not usually change significantly yet the number of jobs is usually subject to change and will increase. Therefore, we decided to only increase the number of jobs in the forthcoming experiments.

4.2 Computational experiment for the population size

Yu and Seif (2016) use a GA for solving a flow shop scheduling problem with diverse maintenance activities and showed that only the population size is statistically significant in improving the quality of solutions. They also showed that increasing the population size up to a certain point increases the quality of solutions. After that point, the quality does not improve significantly, yet the solution time keeps increasing. Table 4 shows the results of the experiment we performed in order to find an appropriate population size for the problem presented in this paper. First, we generated a test problem, solved it with the GA, and then recorded the objective function value (OFV) and the solution while the population size was 25. We used these values as the baseline. Then, we solved the same problem with the

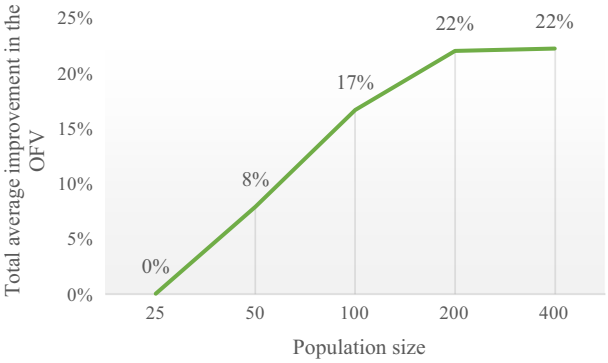

Fig. 6 The impact of increasing the population size on the objective function value (OFV)

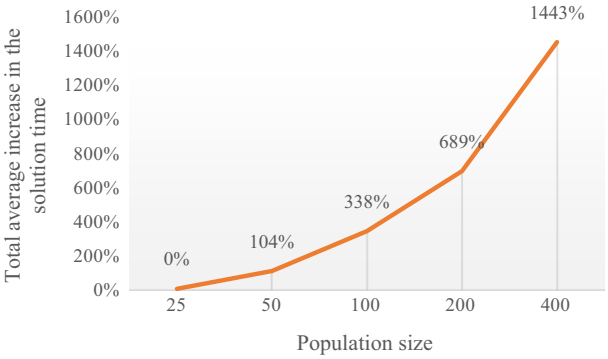

Fig. 7 The impact of increasing the population size on the solution time

same settings, yet with a larger population size, and recorded the *improvement* in the OFV and *increase* in the solution time. We repeated this experiment three times for three different problem sizes. The average improvements and increases are reported in Table 4.

Figure 6 summarizes the results of Table 4, by plotting the total average of improvements and increases against the population size. Increasing the population size beyond 200 does not lead to any improvement in the OFV, yet the solution time keeps increasing, as Fig. 7 shows. This result agrees with the findings of Yu and Seif (2016).

4.3 Computational experiment for performance evaluation

In this section, we present computational experiments that evaluate the performance of the presented simulation–optimization (SO) method compared with CPLEX as a commercial solver that uses exact solution algorithms for the presented SMIP formulation. First, we test the quality of solutions by comparing the results of SO with

those of CPLEX on small-size test problems, in which the global optimal is known. Then we evaluate the performance of the SO method for larger problem sizes under a limited solution time. Finally, we test how sensitive the performance is to the input data. In all of the experiments, we use the following stopping conditions for the SO algorithm and CPLEX. The algorithm will stop and return the best solution after I number of iterations, or after the OFV does not improve for $\lfloor 0.2 \times I \rfloor$ iterations. We chose $I = 100$, but it can take any positive integer value. Obviously, a higher value of I is more likely to result in a lower OFV and higher solution time and depends on user's preference. For CPLEX, we observed that when the optimal solution cannot be found within an hour, there is a possibility that it cannot be found even within several hours. Therefore, we set a time limit of 3000 s for CPLEX. However, if the solution time of the algorithm becomes greater than 3000 s, we use a time limit greater than the solution time of the algorithm, as the CPLEX time limit. We generated a randomized test problem and solved it with both algorithms, SO and CPLEX solver, to ensure these algorithms are consistent and accurate.

Table 5 shows the results of solving 30 test problems solved once via CPLEX and once via SO. The number of jobs (n) is 4 in all of these test problems and the number of scenarios is 30 in each problem. These problems are considered as small-scale problems. In some of the test problems (bold-faced and underlined) the SO method finds the global optimal solution, and in some problems (bold-faced) the gap between the two methods is less than 1.00% for the OFV. However, the solution time of the SO is considerably larger than that of CPLEX. Next, we want to see how the results of the comparison changes when the problem size (number of jobs) increases.

At the bottom of Table 5, the results are summarized by reporting the average, minimum, and maximum values of each column. Table 6 summarizes the results for the average of 30 test problems for different number of jobs. A negative gap means that the SO algorithm has performed better than CPLEX. One observation is that, as the number of jobs (problem size) increases, the quality of CPLEX solutions decreases under a limited solution time. Also, the gap between the SO algorithm and CPLEX decreases with respect to both OFV and solution time. The maximum gaps in the 7th and the 10th columns show the worse performance of the SO algorithm compared to CPLEX. When the gap for the worst-case scenario is negative, it means that the SO algorithm has consistently performed better than CPLEX within a time limit. These values are indicated in boldfaced. As a result, it can be concluded that under a limited solution time, the proposed SO algorithm outperforms a commercial solver as the problem size increases.

In Table 6 we increased the number of jobs up to a point where all 30 test problems can be solved with CPLEX. Table 7 shows the results for $n = 10$. We increased the solution time limit to 5000 s which is considerably higher than the average solution time of the SO algorithm. For the bold-faced test problems, CPLEX was unable to find any feasible solution for the problem. Again, even in the worst-case scenario, the SO algorithm finds a better solution with a lower OFV than CPLEX.

As the last part of the experiments, we want to make sure that the quality of solutions of the algorithm (as measured by the gap in OFV), as well as the time to find a solution, are not dependent on how we generate the input data of the test problems.

Table 5 Comparing the simulation–optimization method with CPLEX for n = 4

Test problem	CPLEX (SMIP)		Simulation–optimization (SO)			Gap	
	OFV	Time (s)	OFV	Time (s)	Iterations	OFV (%)	Time (%)
1	6210.44	96	6263.10	414	45	0.85	332.04
2	6534.53	111	6938.77	443	47	6.19	298.53
3	5556.52	121	5577.18	504	55	0.37	315.17
4	6598.37	98	7166.55	450	48	8.61	357.97
5	5710.22	113	5710.27	455	49	0.00	302.32
6	5211.41	89	5360.73	550	60	2.87	521.18
7	5604.63	136	5604.68	332	36	0.00	144.24
8	7235.49	114	7288.27	751	84	0.73	559.98
9	6993.4	85	7932.77	417	45	13.43	388.73
10	5470.29	86	5728.43	389	42	4.72	353.86
11	6325.52	138	6740.75	451	48	6.56	226.21
12	5700.31	113	6338.80	450	49	11.20	298.14
13	7250.16	86	7616.83	429	46	5.06	396.56
14	5425.85	171	5427.85	525	58	0.04	207.04
15	5291.15	179	6153.85	596	65	16.30	232.12
16	6798.6	143	6814.17	561	60	0.23	291.54
17	5243.63	103	6486.17	390	41	23.70	279.57
18	7035.76	114	7074.84	416	45	0.56	265.91
19	6048.12	112	6092.67	426	46	0.74	279.50
20	7329.23	144	7771.63	555	60	6.04	286.41
21	6832.7	132	7229.67	541	58	5.81	308.26
22	5176.95	114	5177.00	531	58	0.00	365.59
23	5388.92	96	6005.27	526	56	11.44	446.69
24	5831.19	153	6687.63	389	41	14.69	153.71
25	7286.2	94	7527.65	582	64	3.31	519.06
26	5079.85	180	5079.90	430	47	0.00	138.78
27	6122.64	103	6664.98	411	44	8.86	300.46
28	5452.83	124	5452.88	586	64	0.00	374.55
29	7393.03	145	7734.09	441	48	4.61	203.53
30	6329.74	117	6329.80	601	67	0.00	412.40
Average	6148.92	120	6465.91	485	52	5.23	318.67
Minimum	5079.85	85	5079.90	332	36	0.00	138.78
Maximum	7393.03	180	7932.77	751	84	23.70	559.98

In other words, we want to examine the impact of the input data on the performance of the proposed algorithm. We changed the arguments (*DDTF* and *MIF*) of the test problem generator function introduced in Sect. 4.1 and solved 30 test problems for each setting. Table 8 shows a summary of the results. Table 9 provides the average, minimum, and maximum solution times in CPLEX and the SO algorithm, for each setting. In order to examine whether the gap or solution time are significantly

Table 6 Comparing the simulation–optimization method with CPLEX when problem size increases

N of jobs (n)	CPLEX Avg. Time	SO Avg. Time	Avg. N of Iter.	Gap in the solution time			Gap in the OFV		
				Avg. (%)	Min. (%)	Max. (%)	Avg. (%)	Min. (%)	Max. (%)
4	120	485	52	318.67	138.78	559.98	5.23	0.00	23.70
5	2383	777	68	−63.78	−79.81	−21.89	9.43	0.00	32.46
6	3002	1091	79	−63.66	−78.43	−53.80	0.99	−15.18	21.19
7	3001	1430	88	−52.34	−73.58	−44.63	−10.57	−34.84	17.08
8	3001	1828	98	−39.09	−52.89	−35.11	−23.38	−43.39	−6.48
9	3000	2081	98	−30.63	−44.06	−25.03	−26.64	−42.29	−8.22

Table 7 Comparing the simulation–optimization method with CPLEX for n = 10

Test problem	CPLEX (SMIP)		Simulation Optimization			Gap	
	OFV	Time (s)	OFV	Time (s)	Iterations	OFV (%)	Time (%)
1	46,517.1	5001	4934	31969.8	100	− 31.27	− 1.33
2	46,070.3	5000	4862	34,726.5	100	− 24.62	− 2.76
3	48,594	5001	4840	36,507.0	100	− 24.87	− 3.21
4		4999	4778	32,680.8	100		
5	53,894.7	5000	4878	35,050.7	100	− 34.96	− 2.44
6	42,035.5	5000	4796	32,312.5	100	− 23.13	− 4.09
7	46,760	5000	4852	32,919.7	100	− 29.60	− 2.95
8	48,287.1	5000	5011	35,255.7	100	− 26.99	0.22
9		5000	4882	30,531.8	100		
10	37,772.4	5000	4907	25,120.3	100	− 33.50	− 1.86
11	49,068.5	5002	4998	35,048.1	100	− 28.57	− 0.08
12		5000	4996	40,068.8	100		
13	47,049.4	5001	4988	32,131.5	100	− 31.71	− 0.26
14	41,246.2	5000	4864	23,863.0	100	− 42.15	− 2.72
15	43,165.8	5001	4842	34,974.3	100	− 18.98	− 3.18
16	40,944.7	5001	4946	27,929.2	100	− 31.79	− 1.10
17	47,965.9	4999	4819	35,923.4	100	− 25.11	− 3.59
18	46,645	5001	4857	39,589.5	100	− 15.13	− 2.88
19	42,654.6	5000	4900	23,057.4	100	− 45.94	− 2.00
20	47,003.6	5000	4928	39,221.8	100	− 16.56	− 1.46
21	41,515.4	5000	4917	34,909.3	100	− 15.91	− 1.66
22	44,415.9	5000	4878	32,601.5	100	− 26.60	− 2.45
23	50,445	5000	4866	31,690.7	100	− 37.18	− 2.70
24	37,282.9	4999	4825	31,712.7	100	− 14.94	− 3.47
25	33,822.2	5000	4838	30,608.1	100	− 9.50	− 3.25
26		5000	4827	34,578.4	100		
27	36,036.3	5000	4824	33,809.9	100	− 6.18	− 3.52
28	51,582.1	5000	4877	32,966.8	100	− 36.09	− 2.47
29	44,263.5	5000	4764	32,498.6	100	− 26.58	− 4.72
30	49,271	5002	4812	35,931.3	100	− 27.07	− 3.79
Average	44,781.1	5000	32,781.9	4878	100	− 26.34	− 2.45
Minimum	33,822.2	4999	23,057.4	4764	100	− 45.94	− 4.72
Maximum	53,894.7	5002	39,589.5	5011	100	− 6.18	0.22

affected by the input data we performed analysis of variance (ANOVA) on samples drawn from the data used for Tables 8 and 9. Tables 10 and 11 are the ANOVA tables in which five treatments ($a = 5$, the way test problems are generated, the settings), a sample size of seven ($n = 7$), and a confidence interval of $\alpha = 0.01$ is used.

As the results suggest, the input data has no statistical significance in the solution time or the quality of the solutions (the gaps). This is intuitive when looking

Table 8 Sensitivity of the gap to the input data, n = 4

Setting	*DDTF*	*MIF*	No. of problems solved	Gap in the OFV		
				Avg. (%)	Min. (%)	Max. (%)
1	3	50	30	10.42	0.00	26.63
2	4	50	30	9.41	0.00	32.29
3	5	50	30	11.82	0.29	30.77
4	4	40	30	15.39	0.49	32.40
5	4	60	30	11.92	0.00	37.32

at the results in Tables 8 and 9. This finding implies that the results that were shown previously are independent of the problem instances and the conclusions about the performance of the SO method are robust.

5 Case study

In this section we show one of the applications of the presented problem and solution method. Construction projects typically have multiple work zones, or sites at separate distant locations [see Güden and Süral (2014), as an example in railroad construction]. Because the equipment operate in series to complete specific operations (such as the earth-moving) in each location, finding the optimal sequence for routing the equipment between the sites can be treated as a flow shop scheduling problem in which the locations are equivalent to production jobs. On the other hand, before departure from one location to another, the equipment are required to travel to a maintenance station if their usage-based (meter-based) preventive maintenance activities are due.

Due to uncertainty in lead times, and significant varying weather conditions, we can see that stochasticity is a highly pronounced component of scheduling in construction projects (Kerkhove and Vanhoucke 2017). Therefore, the focus of this case study is on stochastic optimization of a realistic problem applicable to operations and maintenance scheduling in construction projects. The input data and description of this case study are adapted from the case study by Yu and Seif (2016) that is designed for a deterministic flow shop scheduling with multiple maintenance activities (MAs) in which combining the MAs was not considered and processing times were fixed regardless of the machines' health state. After presentation of the data and describing the case study, we will discuss the solution and draw managerial implications.

One of the main activities in the early stages of a heavy construction project is earthmoving. A simplified version of the earthmoving process described by Fu (2013) is as follows. The first step is called *preparation*. Excavators are used in this step; they dig natural form of material from the earth. Next, in the *loading* step, wheel loaders can load the removed and prepared soil into haul trucks. Finally, in

Table 9 Sensitivity of the solution time to the input data, n = 4

Setting	*DDTF*	*MIF*	The solution time of CPLEX				The solution time of the SO algorithm			
			Avg. (s)	Min. (s)	Max. (s)	Variance	Avg. (s)	Min. (s)	Max. (s)	Variance
1	4	40	74.20	5.58	259.03	4358.92	576.47	316.50	1013.11	26,187.84
2	4	50	198.48	42.80	418.69	8263.82	551.35	323.34	759.26	11,789.68
3	4	60	203.72	96.48	426.07	5990.88	510.35	228.56	670.08	7919.88
4	3	40	300.70	42.40	1179.03	54,610.11	565.92	361.50	831.70	12,494.47
5	5	40	199.73	48.13	464.04	9319.93	524.53	352.58	722.45	8576.37

Table 10 Analysis of variance for the gap sensitivity experiment

Source of variation	Sum of squares	Degree of freedom	Mean square	F_0	*P*-value
Setting	0.0558	4	0.0140	2.6731	0.0510
Error	0.1567	30	0.0052		
Total	0.2126	34			

Table 11 Analysis of variance for the solution time sensitivity experiment

Source of variation	Sum of squares	Degree of freedom	Mean square	F_0	*P*-value
Setting	19,196.26,013	4	4799.0650	0.4119	0.7986
Error	349,524.7108	30	11,650.8237		
Total	368,720.9709	34			

Table 12 Maintenance intervals (in hours) recommended by the equipment manufacturer (Caterpillar 2010a, b, c). Reproduced from (Yu and Seif 2016)

Machine	10	50	100	250	500	1000
Excavators	✓	✓	✓	✓	✓	✓
Wheel Loaders	✓	✓	✓	✓	✓	✓
(Haul) Trucks	✓	✓			✓	✓

the *hauling* step, haul trucks transport earth to a deposit point by travelling through routes.

Typical (preventive) maintenance activities for construction machinery are usually based on the operating hours of the machinery. In Table 12, maintenance intervals (R_k, $k = 1, \ldots, 6$) recommended by one of the manufacturers of heavy construction equipment is listed for the machinery that are required for the simplified earthmoving process (Caterpillar 2010a, b, c). Different tasks are included in each MA. For example, the tasks included in the 50-h MA of excavators shown are lubrication of boom, stick and bucket linkage, drive shaft universal joint, etc. Table 13 shows the task lists of the 250-h, 500-h, and 1000-h MAs for the excavator. Tasks Numbers 1–4 for the 500-h MA are shared in the 1000-h MA, as shown in boldfaced. Therefore, when combined, the total duration of these two MAs will be approximately 75% of the sum of the durations of the two MAs because 25% of the tasks listed under the two MAs will be redundant when they are combined (assuming that the tasks have the same duration). Although the 250-h MA does not share any tasks with the other two MAs, after checking the details of some of the tasks we noticed that their share certain steps within their tasks. Table 14 shows the steps for performing Task Number 11 of the 250-h MA and Task Number 4 of the 500-h MA. Steps 1, 4, 5, and 6 of the first task are the same as Steps 6, 8, 9, and 10 of the second task. We assumed the duration of the combined tasks to be 60% of the sum of the three durations. In practice, these values can be calculated precisely after a time study is conducted on the MAs.

Table 13 Task list of each MA for the excavator (excerpts from Caterpillar (2010c))

Task No.	Maintenance activity interval (h)		
	250	500	1000
1	Air Conditioner—Test	Axle Oil (Front)—Change	Axle Oil (Front)—Change
2	Axle Bearings (Front)—Lubricate	Axle Oil (Rear)—Change	Axle Oil (Rear)—Change
3	Axle Oil Level (Front)—Check	Final Drive Oil—Change	Battery Hold-Down—Tighten
4	Axle Oil Level (Rear)—Check	Transmission Oil—Change	Drum Brakes—Inspect
5	Braking System—Test	Drive Shaft Support Bearing Lubricant—Check	Final Drive Oil—Change
6	Condenser (Refrigerant)—Clean	Fuel System Priming Pump—Operate	Overhead Guard—Inspect
7	Cooling System Hoses——Inspect	Fuel System Secondary Filter—Replace	Transmission Oil—Change
8	Engine Oil and Filter—Change	Fuel Tank Cap and Strainer—Clean	
9	Final Drive Oil Level—Check	Fuel System Primary Filter/Water Separator-	
10	Swing Bearing—Lubricate	Element—Replace	
11	Transmission Oil Level—Check		
12	V-Belts—Inspect/Adjust/Replace		

Table 14 Two tasks with similar steps (excerpts from Caterpillar (2010c))

Task	Steps
Transmission Oil Level—Check	1. Remove filler plug (1) 2. Check the lubricant level. The lubricant level should be at the bottom of the opening for filler plug (1) 3. If necessary, fill the gearbox with lubricant to the bottom of the opening for filler plug (1) 4. Clean filler plug (1) 5. Inspect the O-ring seal. If damage or wear is noticed on the O-ring seal, replace the seal 6. Install filler plug (1)
Transmission Oil—Change	1. Remove the dirt that is around filler plug (1) and around drain plug (2) 2. Remove drain plug (2). Drain the lubricant into a suitable container 3. Clean drain plug (2) 4. Inspect the O-ring seal. If damage or wear is noticed on the O-ring seal, replace the seal 5. Install drain plug (2) 6. Remove filler plug (1) 7. Fill the gearbox with lubricant to the bottom of the filler plug opening. 8. Clean filler plug (1) 9. Inspect the O-ring seal. If damage or wear is noticed on the O-ring seal, replace the seal 10. Install filler plug (1)

Table 15 Processing times, due dates, and penalty costs for the jobs. Adapted from Yu and Seif (2016)

Location (jobs)	Processing times (no. of days × hours/day)			Due date (days)	Penalty/day
	Excavator	Wheel Loader	Truck		
L_1	$\sim$ TRI$(20, 22, 25) \times 8$	$\sim$ TRI$(25, 28, 30) \times 8$	$\sim$ TRI$(5, 7, 9) \times 16$	90	\$211
L_2	$\sim$ TRI$(15, 20, 25) \times 8$	$\sim$ TRI$(20, 25, 30) \times 8$	$\sim$ TRI$(7, 10, 14) \times 16$	100	\$118
L_3	$\sim$ TRI$(10, 15, 20) \times 8$	$\sim$ TRI$(15, 20, 25) \times 8$	$\sim$ TRI$(3, 9, 14) \times 16$	80	\$118
L_4	$\sim$ TRI$(18, 23, 28) \times 8$	$\sim$ TRI$(15, 20, 25) \times 8$	$\sim$ TRI$(9, 11, 13) \times 16$	70	\$346

We consider a project with four locations, in which earth moving operations need to be done. There are three machines allocated for earthmoving operations of these locations; one excavator, one-wheel loader, and one truck. The locations are too far from each other for the machines to be able to simultaneously work in more than one location. In Table 15, the operation requirements in each location are shown. Due dates are also shown along with the penalty for each day of delay (adapted from Yu and Seif 2016). Table 15 shows the processing times as the number of days a machine is expected to work in a location multiplied by the number of hours worked per day. The due date and penalty costs for completing a job after the due date are also presented in this table. We assumed a 10-h shift for the working days in the last two columns.

Average cost of performing a preventive maintenance activity on a wheel loader is approximately \$234 (Azadeh et al. 2014). We have used this value to approximate

Table 16 The optimal solution for the case study

Sequence of jobs	Scheduled location	Scheduled maintenance activities		
		Excavator	Loader	Truck
1	L_3	–	–	–
2	L_4	MA_{250}	MA_{250}	–
3	L_1	MA_{250}, MA_{500}	$MA_{250}, MA_{500}, MA_{1000}$	MA_{500}
4	L_2	MA_{250}	MA_{250}	–
Total expected cost: \$4078	Expected maintenance cost: \$3076		Expected penalty cost: \$1002	

Table 17 The optimal solution, when the durations of the MAs do not change in combinations

Sequence of jobs	Scheduled location	Scheduled maintenance combinations		
		Excavator	Loader	Truck
1	L_3	–	–	–
2	L_4	MA_{250}	MA_{500}, MA_{1000}	–
3	L_1	MA_{250}, MA_{500}	MA_{250}, MA_{500}	MA_{500}
4	L_2	MA_{250}	MA_{250}	–
Total expected cost: \$4188	Expected maintenance cost: \$3170		Expected penalty cost: \$1018	

the overall cost of each MA's spare part cost, i.e. $\sim U(\$200, \$300)$. We consider \$25/h as the workforce cost. Because the first three MAs (10, 50, 100 h) are usually done in a fraction of an operational day, and usually by the operators, where the machine is operating, and because 2000 h MAs and above are not going to be reached within the scheduling process for this case study, we have considered only the 250-h, 500-h, and 1000-h MAs (denoted as MA_{250}, MA_{500}, and MA_{1000}). Because the trucks do not have the 250-h MA, we set its maintenance interval equal to infinity, $R_{3,1} = +\infty$, in order to nullify it. Although these MAs can be performed ideally in one day, we consider the triangular distribution $\sim TRI(1, 2, 5)$ for the maintenance durations because in practice the machines might wait in the maintenance station for a few days due to spare part unavailability, no empty spot being available in the maintenance stations, etc.

The optimal solution is presented in Table 16. This solution provides a schedule for routing of the machines between the construction locations and a maintenance station, as well as the maintenance plan. For example, the truck goes to L_1 first, then goes to L_4, then to the maintenance station because MA_{500} is scheduled before processing the third job (L_1). Note that the truck does not require to undergo MA_{250}. After maintenance, it goes to L_1, and then to L_2. The excavator and loader need to go to stop for maintenance before operating in any of the locations (except for the first location). When two or more MAs are scheduled in a row, the total maintenance time is shortened due to the concept of combining MAs in Sect. 2.1. Note that the sequence of MAs (when combined) does not affect tardiness or maintenance

costs. In practice, when combined, MAs are performed in a certain order according to maintenance procedures.

We solved the problem again after changing the input data for the durations of the MAs. This time we used the sum of MAs for combinations, instead of a portion (75% or 60%) of the sum. Table 17 shows the solution for the new problem. The only changes in the optimal schedule are the MAs of the loader before going to L_4 and L_1. With the new data, the 500-h and the 1000-h MAs (MA_{500} and MA_{1000}) are prescribed before L_4, and 250-h and the 500-h MAs (MA_{500} and MA_{1000}) are prescribed before going to L_1. This means performing an excessive 500-h MA compared to the original solution. The reason is that in the new data, performing the MAs takes longer which leads to an increase in tardiness. The solver tries to compensate for this increase in the duration of the MAs by performing more MAs so that the processing times of the jobs do not get prolonged (due to the poor health of the machine). However, the value of the objective function is still worse than the original problem.

6 Conclusion

In this paper, we introduced a stochastic maintenance and flow shop scheduling problem adapted to a realistic production or service environment. We incorporated the concept of combined maintenance activities in the permutation flow shop and considered the impact of the health of machines on the processing times of jobs. The objective was to minimize the total cost of maintenance activities and lateness penalties. We formulated the problem as a two-stage stochastic mixed-integer program in which the first-stage decision variables determined both the sequence of the jobs and a combination of maintenance activities to be scheduled before processing the jobs. Because the commercial solvers were not able to solve large-scale instances of the problem in a reasonable time, we proposed a simulation–optimization solution method that can efficiently solve these instances. We designed a series of computational experiments in order to tune the algorithm, evaluate its performance in comparison with CPLEX, and assess the sensitivity of its performance to the input data. We concluded that:

- an increase in the population size in the algorithm improves the quality of the solutions only up to a certain point, after which only the solution time increases,
- although for small-sized instances of the problem we recommend the use of commercial/exact solvers, for medium to large-scale instances and under a limited time frame, the proposed solution method outperforms these computationally and financially expensive solvers, and
- the quality of the solutions and solution time of the presented simulation–optimization method is not sensitive to the input data under a limited solution time, which alludes to the robustness of the method.

We demonstrated an application of the presented problem through a case study in construction projects. The results of the case study showed that considering the decrease maintenance time, when the activities are combined, leads to savings and improvement in the objective function value. Taking random failures into the consideration is highly desirable and can be studied as an extension of this paper. Also, the concept of combined maintenance activities can be incorporated in other production settings such as flexible flow shop and job shop scheduling.

Appendix: Generating the coefficients

Consider the set $\boldsymbol{a} = \{a_1, a_2, \ldots, a_l\}, \forall k = 1, \ldots, l, a_k \in \mathbb{R}$. $\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{2^l}\}$ is the set of all the possible subsets of $\boldsymbol{a}$ that are not null, and $\boldsymbol{b} = \{b_1, b_2, \ldots, b_{2^l}\} \ni b_r = \sum_{a' \in \boldsymbol{s}_r} a', \forall r = 1, \ldots, 2^l$. We want to find the elements of $\boldsymbol{a}$ such that the elements of $\boldsymbol{b}$ are unique. For example, for $\boldsymbol{a}_1 = \{1, 2, 3\}$, $l = 3$, $\boldsymbol{S}_1 = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}. \therefore \boldsymbol{b}_1 = \{1, 2, 3, 3, 4, 5, 6\}$, where the members of $\boldsymbol{b}_1$ are not unique. But for $\boldsymbol{a}_2 = \{1.1, 1.2, 1.3\}, l = 3$, $\boldsymbol{S}_2 = \{\{1.1\}, \{1.2\}, \{1.3\}, \{1.1, 1.2\}, \{1.1, 1.3\}, \{1.2, 1.3\}, \{1.1, 1.2, 1.3\}\}. \therefore \boldsymbol{b}_2 = \{1.1, 1.2, 1.3, 2.3, 2.4, 2.5, 3.6\}$, where the members of $\boldsymbol{b}_2$ are unique. The following proposition shows one way of generating $\boldsymbol{a}$.

Proposition 1 *If* $\boldsymbol{a} = \left\{1.1, 1.01, \ldots, 1.0\underbrace{0 \ldots 0}_{l-1}1\right\}, |\boldsymbol{a}| = l$, *the uniqueness of the elements in* $\boldsymbol{b}$ *is guaranteed.*

Proof For $\boldsymbol{a} = \left\{1.1, 1.01, \ldots, 1.\underbrace{00 \ldots 0}_{l-1}1\right\}, |\boldsymbol{a}| = l$:

$$\boldsymbol{S} = \left\{\{1.1\}, \{1.01\}, \ldots, \left\{1.\underbrace{00 \ldots 0}_{l-1}1\right\}, \{1.1, 1.2\}, \ldots, \left\{1.1, 1.01, \ldots, 1.\underbrace{00 \ldots 0}_{l-1}1\right\}\right\},$$

$$|\boldsymbol{S}| = 2^l - 1. \therefore \boldsymbol{b} = \left\{1.1, 1.01, \ldots, 1.\underbrace{00 \ldots 0}_{l-1}1, 2.11, 2.101, \ldots, l.\underbrace{11 \ldots 1}_{l}\right\}, |\boldsymbol{b}| = 2^l - 1.$$

Assume $\exists b_i, b_j \in \boldsymbol{b}$ *and* $\boldsymbol{s_i}, \boldsymbol{s_j} \in \boldsymbol{S} \ni b_i = b_j$ *and* $\boldsymbol{s_i} \neq \boldsymbol{s_j}$. Because $b_i = b_j$, the integer and decimal parts of the two numbers are equal. Because the integer parts of all of the elements in $\boldsymbol{a}$ are the same (1), the integer part of b_i and b_j indicate the number of elements in their respective subsets, i.e. $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$. Because all elements in each subset have 1 as their integer part and each element has a unique number of zeros in the

decimal part, the elements have to be equal for b_i and b_j to be equal. If the elements of s_i and s_j are identical, $s_i = s_j$. which contradicts the assumption. Therefore, for $\boldsymbol{a} = \left\{1.1, 1.01, \ldots, 1.\underbrace{00\ldots0}_{l-1}1\right\}, |\boldsymbol{a}| = l$ the elements of $\boldsymbol{b}$ are unique. □

References

Aggoune R (2004) Minimizing the makespan for the flow shop scheduling problem with availability constraints. Eur J Oper Res 153(3):534–543. https://doi.org/10.1016/S0377-2217(03)00261-3

Aghezzaf EH, Jamali MA, Ait-Kadi D (2007) An integrated production and preventive maintenance planning model. Eur J Oper Res 181(2):679–685. https://doi.org/10.1016/j.ejor.2006.06.032

Allaoui H, Artiba A (2004) Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. Comput Ind Eng 47(4):431–450. https://doi.org/10.1016/j.cie.2004.09.002

Ángel-Bello F, Álvarez A, Pacheco J, Martínez I (2011) A single machine scheduling problem with availability constraints and sequence-dependent setup costs. Appl Math Model 35(4):2041–2050. https://doi.org/10.1016/j.apm.2010.11.017

Azadeh A, Asadzadeh SM, Seif J (2014) An integrated simulation-analysis of variance methodology for effective analysis of CBM alternatives. Int J Comput Integr Manuf 27(7):624–637

Baesler FF, Sepúlveda JA (2001) Multi-objective simulation optimization for a cancer treatment center. Paper presented at the simulation conference, 2001. Proceedings of the Winter

Barton RR, Meckesheimer M (2006) Metamodel-based simulation optimization. Handb Oper Res Manag Sci 13:535–574

Ben-Daya M, Ait-Kadi D, Duffuaa SO, Knezevic J, Raouf A (2009) Handbook of maintenance management and engineering, vol 7. Springer, Berlin

Berry PM (1993) Uncertainty in scheduling: probability, problem reduction, abstractions and the user. Paper presented at the IEE Colloquium on advanced software technologies for scheduling

Bertolini M, Mezzogori D, Zammori F (2019) Comparison of new metaheuristics, for the solution of an integrated jobs-maintenance scheduling problem. Expert Syst Appl 122:118–136

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv (CSUR) 35(3):268–308

Bock S, Briskorn D, Horbach A (2012) Scheduling flexible maintenance activities subject to job-dependent machine deterioration. J Sched 15:1–14

Carson Y, Maria A (1997) Simulation optimization: methods and applications. Paper presented at the Proceedings of the 29th conference on Winter simulation

Caterpillar (2010a) 725 and 730 OEM Articulated trucks—maintenance intervals. In: Operation and maintenance manual excerpt. Retrieved March 30, 2015, from https://safety.cat.com

Caterpillar (2010b) 988 Wheel loader—maintenance intervals. In: Operation and maintenance manual excerpt. Retrieved March 30, 2015, from https://safety.cat.com

Caterpillar (2010c) M312 and M315 Excavators—maintenance intervals. In: Operation and maintenance manual excerpt. Retrieved March 30, 2015, from https://safety.cat.com

Cheng TE, Wang G (1999) Two-machine flowshop scheduling with consecutive availability constraints. Inf Process Lett 71(2):49–54

Cheng TCE, Wang G (2000) An improved heuristic for two-machine flowshop scheduling with an availability constraint. Oper Res Lett 26(5):223–229. https://doi.org/10.1016/S0167-6377(00)00033-X

Chica M, Juan Pérez AA, Cordon O, Kelton D (2017) Why simheuristics? Benefits, limitations, and best practices when combining metaheuristics with simulation. Available at SSRN: https://ssrn.com/abstract=2919208 or https://doi.org/10.2139/ssrn.2919208

Choi B-C, Lee K, Leung JYT, Pinedo ML (2010) Flow shops with machine maintenance: ordered and proportionate cases. Eur J Oper Res 207(1):97–104. https://doi.org/10.1016/j.ejor.2010.04.018

Dehghanimohammadabadi M (2016) Iterative optimization-based simulation (IOS) with predictable and unpredictable trigger events in simulated time. Western New England University, Springfield

Dehghanimohammadabadi M, Keyser TK (2017) Intelligent simulation: integration of SIMIO and MATLAB to deploy decision support systems to simulation environment. Simul Model Pract Theory 71:45–60

Dehghanimohammadabadi M, Keyser TK, Cheraghi SH (2017) A novel iterative optimization-based simulation (IOS) framework: an effective tool to optimize system's performance. Comput Ind Eng 111(Supplement C):1–17. https://doi.org/10.1016/j.cie.2017.06.037

Dekker R (1995) Integrating optimisation, priority setting, planning and combining of maintenance activities. Eur J Oper Res 82(2):225–240

Dekkert R, Smit A, Losekoot J (1991) Combining maintenance activities in an operational planning phase: a set-partitioning approach. IMA J Manag Math 3(4):315–331

Fu J (2013) Logistics of earthmoving operations: simulation and optimization. KTH Royal Institute of Technology, Stockholm

Göçken M, Boru A, Dosdoğru AT, Geyik F (2015) (R, s, S) inventory control policy and supplier selection in a two-echelon supply chain: an optimization via simulation approach. Paper presented at the Proceedings of the 2015 Winter simulation conference

González-Neira E, Montoya-Torres J, Barrera D (2017) Flow-shop scheduling problem under uncertainties: review and trends. Int J Ind Eng Comput 8(4):399–426

Gourgand M, Grangeon N, Norre S (2000) A review of the static stochastic flow-shop scheduling problem. J Decis Syst 9(2):1–31

Güden H, Süral H (2014) Locating mobile facilities in railway construction management. Omega 45:71–79. https://doi.org/10.1016/j.omega.2014.01.001

Hastings NAJ (2009) Physical asset management. Springer, Berlin

Holland J (1975) Adaption in natural and artificial systems. The University of Michigan Press, Ann Arbor

Hong LJ, Nelson BL, Xu J (2015) Discrete optimization via simulation. In: Fu M (ed) Handbook of simulation optimization. Springer, Berlin, pp 9–44

Jalali H, Nieuwenhuyse IV (2015) Simulation optimization in inventory replenishment: a classification. IIE Trans 47(11):1217–1235

Johnson SM (1954) Optimal two-and three-stage production schedules with setup times included. Naval Res Logist (NRL) 1(1):61–68

Juan AA, Faulin J, Grasman SE, Rabe M, Figueira G (2015) A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems. Oper Res Perspect 2:62–72

Kerkhove LP, Vanhoucke M (2017) Optimised scheduling for weather sensitive offshore construction projects. Omega 66:58–78. https://doi.org/10.1016/j.omega.2016.01.011

Knezevic J (1997) Systems maintainability, vol 1. Springer, Berlin

Kroep S (2017) *Clustering of aircraft maintenance jobs at AIS airlines*. University of Twente, Enschede

Kubiak W, Błażewicz J, Formanowicz P, Breit J, Schmidt G (2002) Two-machine flow shops with limited machine availability. Eur J Oper Res 136(3):528–540. https://doi.org/10.1016/S0377-2217(01)00083-2

Kubzin MA, Potts CN, Strusevich VA (2009) Approximation results for flow shop scheduling problems with machine availability constraints. Comput Oper Res 36(2):379–390. https://doi.org/10.1016/j.cor.2007.10.013

Lee C-Y (1997) Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. Oper Res Lett 20(3):129–139. https://doi.org/10.1016/S0167-6377(96)00041-7

Lee C-Y (1999) Two-machine flowshop scheduling with availability constraints. Eur J Oper Res 114(2):420–429. https://doi.org/10.1016/S0377-2217(97)00452-9

Lee CY, Leon VJ (2001) Machine scheduling with a rate-modifying activity1. Eur J Oper Res 128(1):119–128. https://doi.org/10.1016/S0377-2217(99)00066-1

Lin JT, Chiu C-C, Chang Y-H (2017) Simulation-based optimization approach for simultaneous scheduling of vehicles and machines with processing time uncertainty in FMS. Flex Serv Manuf J 1:1. https://doi.org/10.1007/s10696-017-9302-x

Ortiz NR, Diaz FJ (2018) The project scheduling problem with non-deterministic activities duration: a literature review. J Ind Eng Manag 11(1):116

Pinedo M (2012) Scheduling. Springer, Berlin

Rani D, Moreira MM (2010) Simulation–optimization modeling: a survey and potential application in reservoir systems operation. Water Resour Manag 24(6):1107–1138

Rossit DA, Tohmé F, Frutos M (2018) The non-permutation flow-shop scheduling problem: a literature review. Omega 77:143–153. https://doi.org/10.1016/j.omega.2017.05.010

Seif J, Yu AJ, Rahmanniyay F (2018) Modeling and optimization of a bi-objective flow shop scheduling with diverse maintenance requirements. Int J Prod Res 56(9):3204–3225

Trevino V, Falciani F (2006) GALGO: an R package for multivariate variable selection using genetic algorithms. Bioinformatics 22(9):1154–1156. https://doi.org/10.1093/bioinformatics/btl074

Truong TH, Azadivar F (2003) Simulation optimization in manufacturing analysis: simulation based optimization for supply chain configuration design

Xu D, Wan L, Liu A, Yang DL (2015) Single machine total completion time scheduling problem with workload-dependent maintenance duration. Omega (United Kingdom) 52:101–106. https://doi.org/10.1016/j.omega.2014.11.002

Yenisey MM, Yagmahan B (2014) Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega 45:119–135. https://doi.org/10.1016/j.omega.2013.07.004

Yip H-L, Fan H, Chiang Y-H (2014) Predicting the maintenance cost of construction equipment: comparison between general regression neural network and Box–Jenkins time series models. Autom Constr 38:30–38. https://doi.org/10.1016/j.autcon.2013.10.024

Yoo J, Lee IS (2016) Parallel machine scheduling with maintenance activities. Comput Ind Eng 101(Supplement C):361–371. https://doi.org/10.1016/j.cie.2016.09.020

Yu AJ, Seif J (2016) Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA. Comput Ind Eng 97:26–40

Zheng Y, Lian L, Fu Z, Mesghouni K (2015) Evolutional algorithm in solving flexible job shop scheduling problem with uncertainties. In: LISS 2013. Springer, pp. 1009–1015

Javad Seif holds three degrees in industrial engineering (B.S., M.S., and Ph.D.) and a graduate certificate in reliability and maintainability engineering from the University of Tennessee. Prior to his graduate studies, Dr. Seif worked in industry for several years designing, developing, and implementing industrial information systems for various service and manufacturing companies. His main research interests are within the broad fields of operations research and data science applied to problems in maintenance planning and scheduling, flight and maintenance planning, physical asset management, and reliability analysis. Dr. Seif is now a Post-Doctoral Research Associate at the Reliability and Maintainabiity Center (RMC) of the University of Tennessee at Knoxville (UTK).

Mohammad Dehghanimohammadabadi is an Assistant Teaching Professor of Mechanical and Industrial engineering at Northeastern University (Boston, USA). He received his Ph.D. in Engineering Management from Western New England University (USA) in 2016. His research is mainly focused on developing and generalizing simulation and optimization frameworks in different disciplines such as healthcare, supply chain, and manufacturing.

Andrew Junfang Yu received his B.S. in Electrical Engineering and M.S. in Management Science from Shanghai Jiao Tong University and earned his M.S. in Systems Science and Ph.D. in Industrial Engineering from Louisiana State University. He worked as a supply chain and logistics solution architect at i2 Technologies (JDA) and as an Assistant Professor at Louisiana State University and Southern Methodist University. He also worked on the faculty at Shanghai Jiao Tong University. Dr. Yu is now an Associate Professor at the University of Tennessee. Dr. Yu's main research interests are in the areas of supply chain and logistics management, maintenance planning and scheduling, data analytics and science, and systems engineering.