Oper Res Int J (2020) 20:927–942
DOI 10.1007/s12351-017-0349-y

ORIGINAL PAPER

Single machine scheduling with sequence-dependent setup times and delayed precedence constraints

Yiyo Kuo[1] · Sheng-I Chen[2] · Yen-Hung Yeh[1]

Received: 10 November 2016 / Revised: 22 March 2017 / Accepted: 18 September 2017 /
Published online: 21 September 2017

Abstract This research deals with the single machine scheduling problem of minimizing the makespan with sequence dependent setup times and delayed precedence constraints. A makespan calculation model is first proposed. When given a feasible job sequence, the proposed model can calculate the makespan. Then a variable neighbourhood search (VNS) with four phases is proposed for optimizing the job sequence. The proposed VNS adopts five operations to search for new solutions, and modifies all solutions to satisfy precedence constraints. The proposed VNS will accept a worse solution over a better solution with a certain probability, in order to escape from a local optimum. The experimental results show that the proposed VNS provides the best results with less than 10 s of computation time. Therefore it is efficient and effective in solving the single machine scheduling problems.

Keywords Single machine scheduling · Sequence-dependent setup times · Delayed precedence constraints · Variable neighbourhood search

1 Introduction

Single machine scheduling is a classical optimization problem that represents multiple real life systems in which a single resource (the machine) represents the whole system or the bottleneck operation of the system (Pereira 2016). This

✉ Yiyo Kuo
yiyo@mail.mcut.edu.tw

[1] Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei City 24301, Taiwan

[2] Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu 300, Taiwan

research aims to minimize the makespan of a single machine scheduling problem, in which sequence-dependent setup times and delayed precedence constraints are taken into consideration. In many manufacturing operations, setup time is needed before processing jobs. When the length of setup is dependent on the immediately preceding operation the system is said to have sequence-dependent setup times (Jula and Kones 2013). Sequence-dependent setup times are commonly observed in various industrial settings including printing, textiles, pharmaceuticals, chemical and metallurgical industries (Choobineh et al. 2006).

Moreover, a job often also needs to be processed before or after other jobs, e.g., due to tool or fixture restrictions, or for other case-dependent technological reason, which lead to precedence constraints (Potts 1985; Lawler 1978; Tanaka and Sato 2013; Davari et al. 2016). Based on precedence constraints, it is possible that a job needs to be processed before or after other jobs by at least a certain period. This usually happens when a product needs to be processed on the machine more than once. For example, i and j are two jobs on a product which are designed to be processed successively on the same machine. Job j cannot be processed immediately and must waits for a period of time when job i is completed, even if the machine has been set up for the new job. That means the release time of job j is dependent on the completion time of job i. The precedence constraints then become delayed precedence constraints. This kind of constraint might arise because the temperature of the product is too high after job i, and it needs a period of time to cool down, or colors are painted in job i, and need a period of time to dry. Therefore the starting time of job j is dependent on the completion time of job i and the corresponding delay time. When other products are assigned to be processed on the same machine, some jobs of other products could be scheduled after the job i to increase the efficiency of the machine and prevent loss of time waiting for job j.

The motivation for solving the single machine scheduling problem with sequence-dependent setup times and delayed precedence constraints is a practical problem in an amplifier assembly company in Taiwan. The assembly process is divided into several parts. The company executes one part at a time. Because different tools and assembly components are required for different parts or different products, setup times are required when executing different part or different products. Moreover, a lot of amplifiers are installed outdoors, and in order to make them waterproof all covers have to be assembled with special glue, and a certain period has to elapse before the glue dries. Therefore delayed precedence constraints are also present in the amplifier assembly process. The company usually assembles other products during the delay time. However, the assembly time of each part is dependent on the batch size and product type. When the number of product types increases and batch size reduces in the future, the scheduling problem will become more complex.

Graham et al. (1979) proposed a well know three-field classification for analyzing all scheduling. The first field specifies the machine environment, the middle field states any special job characteristics and the third field denotes the objective to be minimized (Koulamas 2010). A single machine scheduling problem, in which sequence-dependent setup times and precedence constraints are taken into consideration simultaneously, can be presented as $1\ |s_{ij}, prec(d_{ij})|\ C_{max}$, in which s_{ij}, represents sequence-dependent setup time between job i and j, $prec$ represents the

precedence constraints, d_{ij} represents the corresponding delay times and C_{max} represents the makespan.

Research dealing with delayed precedence constraints is rare in the literature (Wikum et al. 1994; Balas et al. 1995; Finta and Liu 1996; Brucker et al. 1999). The notion of delayed precedence constraints can be used to model the release date of the jobs (Finta and Liu 1996). Bases on the delayed precedence constraints, the release times of some jobs are unknown at the beginning. When given a job sequence which satisfies all precedence constraints, the release times of these jobs can be calculated; the release time of these jobs are dependent on the scheduling results. Wikum et al. (1994) deal with some special cases of single machine scheduling with delayed precedence constraints. These special cases are classified based on the upper bound of delay times, lower bound of delay times and structure of precedence relations. For some special, simple precedence relations, they present polynomial algorithms. For special complex precedence structures, they establish that the problems are NP-hard. Finally they provide heuristics and wort-case bounds for one of the hard problems. In their research, only chain-type precedence relationships are considered. Balas et al. (1995) study the one machine scheduling problem with release and delivery times and delay precedence constraints. They propose a branch-and-bound algorithm in which a modified Longest Tail Heuristic is used to minimize the makespan. Then the proposed algorithm is embedded in a modified version of the Shifting Bottleneck Procedure for solving job shop scheduling problems. Finta and Liu (1996) consider a single-machine scheduling problem with precedence delays for the minimization of makespan. An $O(n^2)$ optimal algorithm is provided when tasks have arbitrary integer execution times and precedence delays have unit length. Brucker et al. (1999) proposed a branch and bound algorithm for a single machine scheduling problem with positive and negative time-lags representing the time between the start times of two jobs. These precedence relationships are called start–start relations, and are different from those that we deal with here. From the review of the above literature it is found that set up times between all pairs of jobs have not been taken into consideration.

A topic that is closely related to the present research is resource dependent release times. Scheduling problems with resource dependent release times have been studied mainly in single machine environments (Li et al. 1995; Ventura et al. 2002). Metallurgy is the most extensively cited application area for scheduling with resource dependent job parameters (Cheng et al. 2006a). Ingot batches must be preheated to the required temperature in a soaking pit before they can be hot-rolled in a blooming mill (Janiak 1998). The preheating time is a non-increasing function of the amount of gas consumed in heating the soaking pit. Thus the preheating time of an ingot may be treated as the release time at which the ingot is available for the job of ingot rolling (Choi et al. 2007). Based on the problem structure of metallurgy, some results have appeared in the literature. They can be found in Janiak (1991, 1998), Cheng and Janiak (1994), Li (1994), Ventura et al. (2002) Cheng et al. (2006a, b) and Choi et al. (2007).

Research that deals with sequence-dependent setup times and delayed precedence constraints to minimize makespan cannot be found in literature. For a single machine scheduling problem, when all jobs are released at the beginning and there

are no sequence-dependent setup times, the makespan is independent of the sequence and equal to the sum of the processing times. However, when there are sequence-dependent setup times, the makespan depends on the schedule and the problem becomes a strongly NP-hard problem (Pinedo 1995). Moreover, most single machine scheduling problems with delayed precedence constraints are also strongly NP-hard (Wikum et al. 1994; Brucker et al. 1999). Variable neighbourhood search (VNS) is a metaheuristic which uses systematic changes of neighbourhood within a possibly randomized local search algorithm to produce a simple and effective approach to combinatorial and global optimization (Hansen and Mladenovic' 2001). In this research, a VNS is proposed as a method for solving the single machine scheduling problem.

The remaining sections of this paper are organized as follows. In Sect. 2, the proposed single machine scheduling problem is formulated and followed by corresponding explanations. In Sect. 3, based on the proposed structure, a makespan calculation model is proposed. In Sect. 4, a VNS is proposed to minimize the makespan. Experimental results are reported in Sect. 5, followed by a summary of results and concluding remarks in Sect. 6.

2 Problem description and formulation

Consider that n jobs are to be processed without preemption on a machine that can process at most one job at a time. Job i (one of the n jobs) has a processing time, p_i. A setup time, s_{ij}, is given for every pair of jobs i and j, and s_{ij} is required before processing job j if job i is an immediate predecessor of job j. Assume that A is the set of all precedence constraints. For any pair of jobs $(i, j) \in A$, the starting time of job j, b_j, not only have to be larger than the completion time of job i, c_i, a delay time, d_{ij}, also needs to be taken into consideration. Thus $b_j \geq c_i + d_{ij}$. If $x_{ij} = 1$ indicate job j is preceded immediately by job i, otherwise $x_{ij} = 0$, the proposed 1 $|s_{ij}$, $prec(d_{ij})|$ C_{max}, can be formulated as follow:

$$\begin{aligned} &\text{Minimize } C_{max} \\ &\text{Subject to} \end{aligned} \tag{1}$$

$$C_{max} \geq c_i \quad i = 0, 1, \ldots, n+1 \tag{2}$$

$$\sum_{j=0}^{n+1} x_{ij} = 1 \quad i = 0, 1, \ldots, n (i \neq j) \tag{3}$$

$$\sum_{i=0}^{n+1} x_{ij} = 1 \quad j = 1, 2, \ldots, n+1 \, (i \neq j) \tag{4}$$

$$\sum_{j=0}^{n} x_{ij} = 0 \quad i = n+1 \tag{5}$$

$$\sum_{i=1}^{n+1} x_{ij} = 0 \quad j = 0 \tag{6}$$

$$b_j \geq \left(c_i + s_{ij}\right)x_{ij} \quad j = 0, 1, \ldots, n+1 \; (i \neq j) \tag{7}$$

$$b_j \geq \left(c_i + d_{ij}\right) \quad \forall (i,j) \in A \tag{8}$$

$$c_i = b_i + p_i \quad i = 0, 1, \ldots, n+1 \tag{9}$$

$$x_{ij} \in \{0, 1\} \quad i = 0, 1, \ldots, n \text{ and } j = 0, 1, \ldots, n \tag{10}$$

Equation (1) is the objective function, in which C_{max} indicates the makespan and is equal to the completion time of the last job. The relationship between C_{max} and completion time of all jobs, c_i, is set in Eq. (2). Equation (3) ensures that each job can be succeeded immediately by just one job. Equation (4) ensures that each job is preceded immediately by just one job. Equation (5) ensures that job $n + 1$ is a dummy job and it is the last job and succeeded immediately by no job. Equation (6) ensures that job 0 is a dummy job and that it is the first job and preceded immediately by no job. Equation (7) ensures that the time interval between the starting time of job j and the completion time of job i is greater than the setup time between job i and j, if job j is preceded immediately by job i. Equation (8) ensure that the time interval between the starting time of job j and the completion time of job i is greater than the delay time between job i and j, if the pair of jobs $(i, j) \in A$. The relationship between starting time, completion time and processing time is set in Eq. (9). Finally the sequence of any two jobs can take only binary values as given in Eq. (10).

3 Makespan calculation

This research aims to minimize the makespan by finding an optimal job sequence for a single machine production system in which sequence-dependent setup times and delayed precedence constraints are taken into consideration. Before the job sequence can be optimized, a makesapn calculation model is firstly proposed as illustrated in Fig. 1. In Fig. 1, $\pi(r)$ indicates the job with sequence r (i.e. $\pi(r) = k$ implies the r-th job to be process is job k) and es_j indicates the earliest starting time of job j. When given a job sequence which satisfies all precedence constraints, the procedure in Fig. 1 follows the sequence of jobs to calculate the completion time of the job with sequence r and the starting time of job with sequence $r + 1$ sequentially.

According to the relationship of Eq. (9), for the job with sequence r, the completion time, $c_{\pi(r)}$, is equal to $b_{\pi(r)} + p_{\pi(r)}$. If the completion time of a job with sequence r is calculated and it is the precedent job in one of the set of precedence constraints, A, then the earliest starting time of the corresponding consequent job can be calculated. i.e. $\pi(r) = i$, and $(i, j) \in A$, then $es_j = c_i + d_{ij}$. Moreover, the

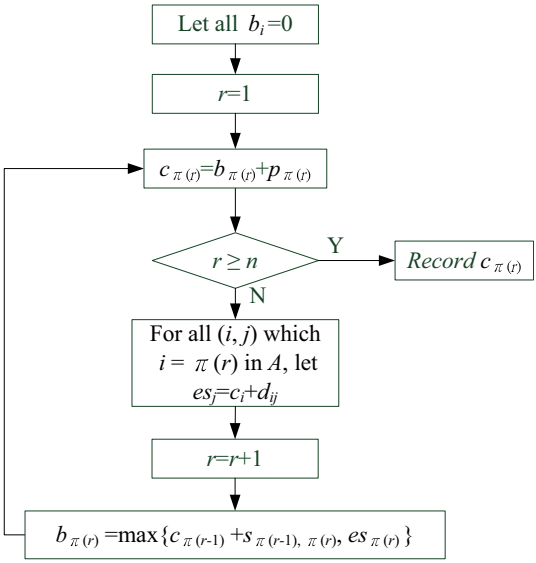


Fig. 1 The proposed makespan calculation model

starting time of a job with sequence r, $b_{\pi(r)}$, is dependent on completion time of the job with sequence $r - 1$, $c_{\pi(r-1)}$, plus the setup time between these two jobs, $s_{\pi(r-1),\,\pi(r)}$, and its earliest starting time, $es_{\pi(r)}$. Finally, the makespan of the job sequence is the completion time of the last job.

4 Variable neighbourhood search

In Sect. 3, a procedure was proposed to calculate the makespan when given a job sequence which satisfies all precedence constraints. The present study adopts a VNS to optimize the sequence. VNS is a metaheuristic that proposes systematic changes to the neighborhood structure within a search to solve optimization problems (Vahdani and Zandieh 2010).

The proposed VNS starts with an initial solution, X, and then keeps finding new neighbourhood solutions by some systematical strategies. However, the newly generated solution may not satisfy all precedence constraints, and may need to be modified. If the performance of the new modified neighbourhood solution, X_{new}, is better (shorter makespan), then the solution, X, will be replaced by the sew solution, X_{new}, to find its neighbourhood solutions. On the other hand, if the performance of the new solution, X_{new}, is worse (longer makespan), then the VNS will keep finding other new neighbourhood solutions based on the original solution X. However, if no better solution is found after a certain number of solutions are searched, the proposed VNS may then accept the new solution even if the performance is worse.

Therefore the proposed VNS for the 1 $|s_{ij}, prec(d_{ij})| C_{max}$, comprises four phases: generation of the initial solution, neighbourhood search, solution modification,

neighbourhood solution acceptance. The details of these phases are described in the following sections.

4.1 Initial solution generation

In this research, the initial solution for the single machine scheduling problem is generated based on the setup times. First, two jobs with the shortest setup time are selected to be a two jobs sequence. For the sequence, there are two ends and each end indicates a job. Then an unselected job which has the shortest setup time with the job at the two ends of the sequence is selected to extend the job sequence. Once when all jobs have been selected into the sequence, the initial solution generation is completed.

4.2 Neighbourhood search

In the present research, five operations are proposed for the search for neighbourhood solutions. They are job exchange, job insertion, job series move I, job series move II and job series exchange. The proposed VNS for searching the neighbourhood for a more economical solution (neighbourhood descent) randomly selects one of the five operations to search the neighbourhood solutions. The details of the five operations are introduced in the following sections.

4.2.1 Job exchange

The job exchange operator selects two jobs at random and then exchanges their position in sequence. An example of the job exchange operator is shown in Fig. 2.

4.2.2 Job insertion

The job insertion operator selects a job at random, and then inserts the selected job in between two other successive jobs. An example of the job insertion operator is shown in Fig. 3.

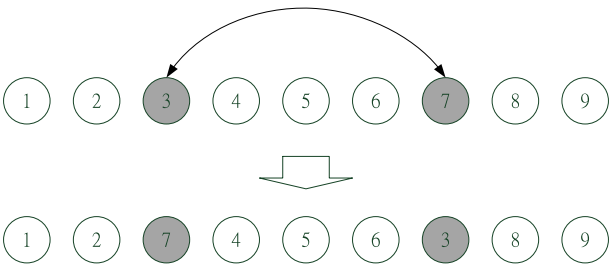


Fig. 2 An example of job exchange operator

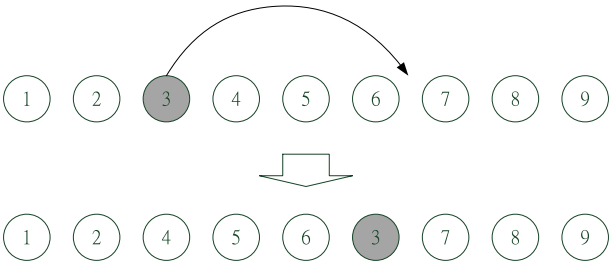


Fig. 3 An example of job insert operator

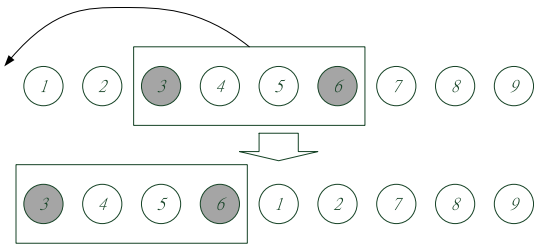
4.2.3 Job series move I

The job series move I operator selects two jobs at random, and then move all jobs (including the two selected jobs) between the selected two jobs to the beginning of the job sequence. An example of the job series move I operator is shown in Fig. 4.

4.2.4 Job series move II

Like the job series move I operator, the series move II operator selects two jobs at random, and then move all jobs (include the two selected jobs) between the selected two jobs to the end of the job sequence. An example of the job series move II operator is shown in Fig. 5.

Fig. 4 An example of job series move I



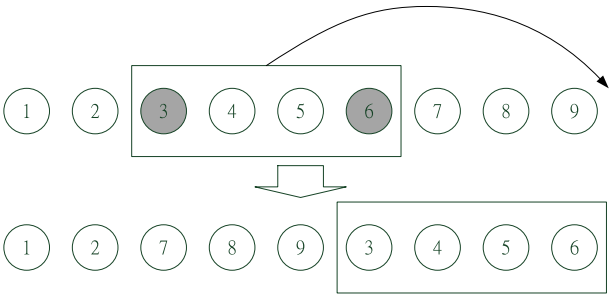


Fig. 5 An example of job series move II

4.2.5 Job series exchange

The job series exchange operator selects one job randomly. Based on the selected job, the job sequence is divided into two series. Then these two series are exchanged. After the job series exchange operator, the selected job becomes the last job of the sequence. An example of the job series exchange operator is shown in Fig. 6.

The use of multiple operations can make the solution space search more extensive (Chen et al. 2010). The above five operations may not make the solution space search totally extensive, but they provide the opportunity to search neighbourhoods by changing job sequences over a small or wide range.

4.3 Solution modification

For each solution that is generated by the initial solution generation and neighbourhood solution search, the solution may not be feasible due to delayed precedence constraints. For any pair of jobs $(i, j) \in A$, the position of job i in the job sequence must be arranged in front of job j. This research proposed a procedure to modify the solutions based on the newly generated solutions (initial solutions and neighbourhood solutions). When there are W pair of jobs $(i, j) \in A$, the procedure is shown in Fig. 7.

The proposed solution modification deals with all pair of jobs $(i, j) \in A$. The proposed procedure selects a pair of jobs $(i, j) \in A$ randomly and tests if the selected (i, j) satisfy the precedence constraint. If the selected (i, j) satisfies the precedence constraint, then the procedure keeps testing other pairs (i, j). However, if the selected (i, j) does not satisfy the precedence constraint, then the position of job j will be changed to a random position behind job i. The change of job position, ensures that the selected (i, j) satisfy the precedence constraint, but it may cause other pair of jobs $(i, j) \in A$ which have passed the test to violate precedence constraint. For example, in a four-job sequence, 1–2–3–4, with two precedence constraints (2, 3) and (4, 2). The job sequence satisfies the precedence constraints (2, 3), but violates precedence constraint (4, 2). When the position of job 2 is moved behind job 4, the job sequence then becomes 1–3–4–2 which violates precedence

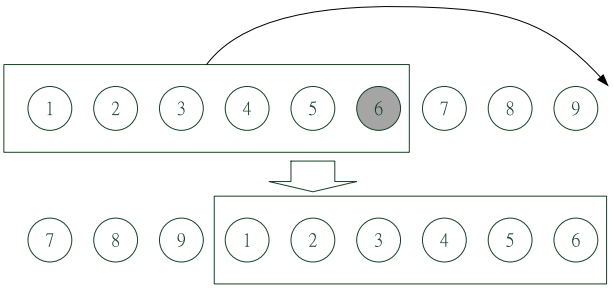


Fig. 6 An example of job series exchange

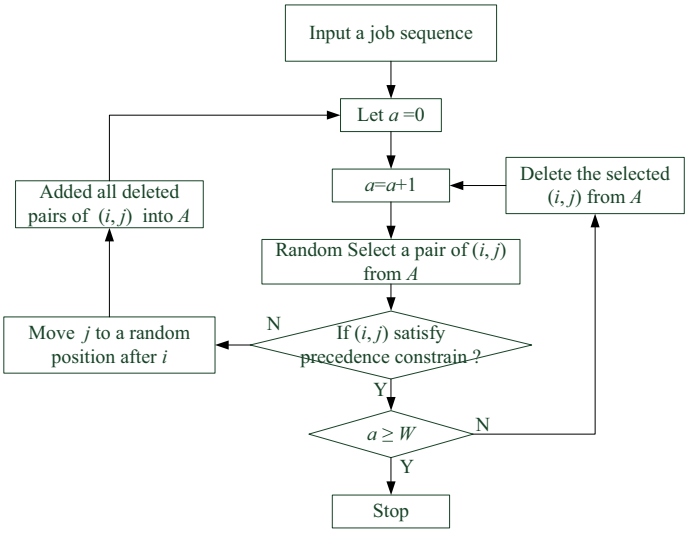


Fig. 7 The procedure of solution modification

constraint (2, 3). Therefore, if the position of any job changed, the proposed solution modification procedure will restart to test all precedence constraint at randomly. The main purpose of solution modification is not only to make the solution feasible, but also to provide the opportunity to make the solution space more extensive.

4.4 Neighbourhood solution acceptance

According to the proposed neighborhood search and solutions modification method, it is possible that no better solution is found after a certain number of neighborhood solutions are searched. In this situation, the proposed VNS may then accept new neighborhood solutions based on the performance of the new solution, even if it is worse. This research adopts the concept of simulated annealing (SA), which accepts worse solutions to escape from a local optimum (Kirkpatrick et al. 1983).

This research aims to minimize makespan for the single machine scheduling problem. If no better solution is found after a number of solutions, m, are searched, and the next new searched solution, X_{new}, is also worse, the new worse solution may be accepted based on a probability which dependent on its performance, $Z(X_{new})$ and the convergent requirement. The acceptance probability can be calculated by Eq. (11).

$$P(X_{new}) = e^{(-\Delta Z/T)} \tag{11}$$

In Eq. (11), $P(X_{new})$ is the is the probability of acceptance for solution X_{new}, ΔZ is the difference between the performances of the solution X, and X_{new}, and it can be calculated by Eq. (12).

$$\Delta Z = Z(X_{new}) - Z(X) \tag{12}$$

Moreover, in Eq. (11), T is the gradually reducing value. The more solutions are searched, the lower is the value of T. In this research, when given the upper bound and lower bound of T, the reducing speed can be controlled by a parameter α ($0 < \alpha < 1$). It is assumes that the T_s and T_e indicate the upper bound and lower bound of the T, while T_s is the initial value of T and ΔT is the difference between T_s and T_e. Each time when a new solution is searched, the ΔT is reduce to$\alpha\Delta T$, and then T is reduce to $T_e + \alpha\Delta T$. Gradually, T will keep gradually reduce until it is very close to the T_e.

Therefore, according to the Eq. (11), the probability that a worse solution will be accepted reduces as its performances or number of solutions searched increases. The VSN is terminated when K solutions have been searched. The entire procedure of the proposed VNS is illustrated in Fig. 8.

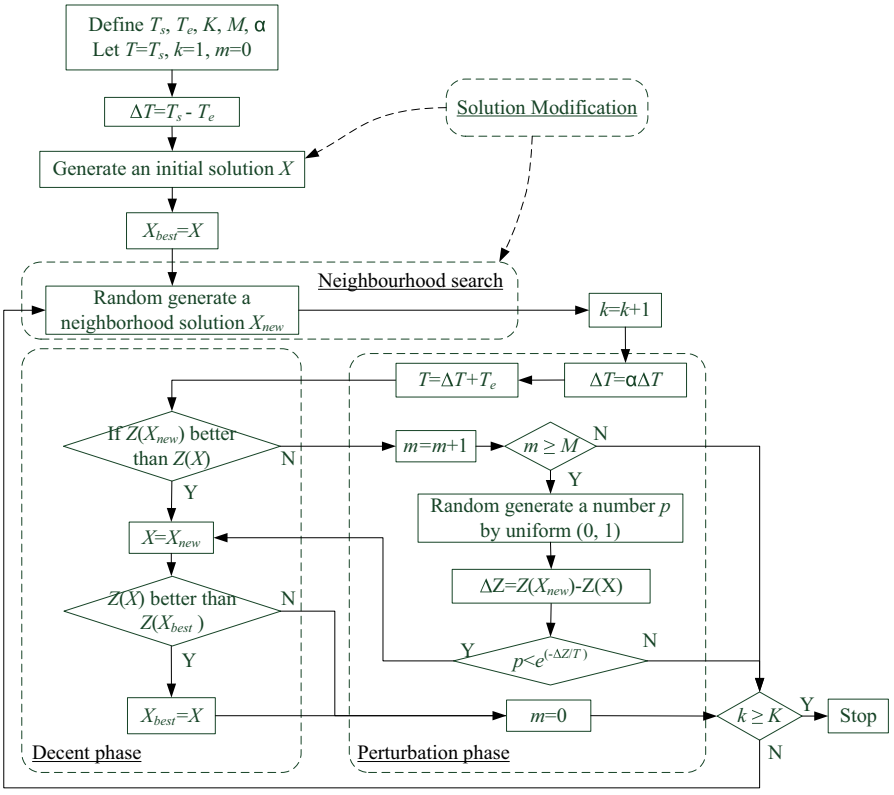


Fig. 8 The procedure of proposed VNS

5 Computational experiments

According to the structure of the proposed single machine scheduling problem, the number of jobs, n, processing times of all jobs, p_i, setup times between all pairs of jobs, s_{ij}, all pairs of jobs $(i, j) \in A$ and their corresponding delay time, d_{ij}, should be given before testing the performance of the proposed VNS. Assuming that there are 20 jobs, this research randomly generated 5 problems for each combination of precedence constraints, processing times of all jobs, setup times between all pairs of jobs, and delay times of corresponding precedence constraints. Each combination is called a scenario. Table 1 shows the total of 16 scenarios, which produced a total of 80 problems for testing. To specify the precedence constraints, all pairs of jobs are selected. Sets of constraints which are impossible are excluded. For example, three precedence constraints (1, 2), (2, 3) and (3, 1) are impossible in reality, and are not considered when generating a feasible solution. Thus, for all randomly selected pairs of precedence constraints (i, j), $i < j$ is required.

In relation to VNS, Yang et al. (2005) proposed a procedure to decide T_s and T_e. Then the acceptance probability in the beginning of VNS will be greater than 50%, and the acceptance probability in the end of VNS will be adequately small. Based on the procedure, Kuo (2010) proposed Eqs. (13) and (14) to calculate T_s and T_e.

$$T_s = -(Z_{max} - Z_{min}) / \log_e 0.5 \tag{13}$$

$$T_e = -(Z_{max} - Z_{min}) \times 0.0001 / \log_e 0.05 \tag{14}$$

Table 1 The structure of testing problems

No.	Number of precedence constraints	Processing times	Setup times	Delay times for precedence constraint
1	10	[10, 20]	[5, 10]	[20, 40]
2	10	[10, 20]	[5, 10]	[40, 60]
3	10	[10, 20]	[10, 15]	[20, 40]
4	10	[10, 20]	[10, 15]	[40, 60]
5	10	[20, 30]	[5, 10]	[20, 40]
6	10	[20, 30]	[5, 10]	[40, 60]
7	10	[20, 30]	[10, 15]	[20, 40]
8	10	[20, 30]	[10, 15]	[40, 60]
9	15	[10, 20]	[5, 10]	[20, 40]
10	15	[10, 20]	[5, 10]	[40, 60]
11	15	[10, 20]	[10, 15]	[20, 40]
12	15	[10, 20]	[10, 15]	[40, 60]
13	15	[20, 30]	[5, 10]	[20, 40]
14	15	[20, 30]	[5, 10]	[40, 60]
15	15	[20, 30]	[10, 15]	[20, 40]
16	15	[20, 30]	[10, 15]	[40, 60]

In both equation, Z_{max} is the likely worst performance of all solutions and Z_{min} is the likely best performance of all solutions. Therefore ($Z_{max} - Z_{min}$) is likely to be the upper bound of ΔZ. In this research, the likely worst solution is calculated assuming that all processing times, setup times and delay time are the maximum values, and the likely best solution is calculated assuming that all processing times and setup times are the minimum values and all delay times for all corresponding precedence constraints are ignored. Therefore Z_{max} and Z_{min} are calculated by Eqs. (15) and (16) in which n is the number of jobs, $\overline{p}$ is the upper bound of processing times, $\overline{s}$ is the upper bound of setup times, W is the number of precedence constraints, $\overline{d}$ is the upper bound of delay times, $\underline{p}$ is the lower bound of processing times and $\underline{s}$ is the lower bound of setup times. Moreover, in this research k, m, and α are set as 1,000,000, 50 and 0.99, respectively. The average results for all 16 scenarios are shown in the second column of Table 2.

$$Z_{max} = n \times \overline{p} + (n-1) \times \overline{s} + W \times \overline{d} \tag{15}$$

$$Z_{max} = n \times \underline{p} + (n-1) \times \underline{s} \tag{16}$$

In addition to the VNS, this research also adopts SA for the proposed single machine scheduling problem. SA was proposed by Kirkpatrick et al. (1983). Like VNS, the SA algorithm has been widely used in solving sophisticated optimization

Table 2 The experimental results

No.	VNS	SA1	SA2	SA3	SA4	SA5
1	393.6	398.0	400.6	406.4	422.8	472.0
2	392.2	396.0	394.8	399.8	451.8	548.2
3	500.0	507.4	506.4	509.6	539.4	569.0
4	505.6	512.6	511.2	513.6	536.4	584.2
5	582.8	584.2	587.4	586.6	597.4	688.2
6	595.4	597.6	602.8	608.2	623.6	744.4
7	697.0	699.8	700.8	707.6	728.0	753.8
8	697.2	701.8	704.6	717.0	783.4	848.0
9	438.4	442.6	451.0	445.0	498.4	584.0
10	463.8	468.2	477.8	499.4	607.6	589.4
11	534.6	545.2	551.8	562.2	588.2	676.4
12	547.2	554.4	553.2	560.6	639.2	656.6
13	603.4	608.4	616.4	623.6	651.4	738.0
14	662.6	673.8	679.8	683.6	791.6	876.2
15	690.2	697.6	696.6	701.4	743.2	840.2
16	744.4	751.0	753.2	782.8	831.0	820.8
Average	565.5	571.2	574.3	581.7	627.1	686.8
Improvement (%)		0.99	1.52	2.78	9.82	17.65

Table 3 ANOVA summary for the VNS results

Factor	Sun of square	Degree of freedom	Mean square	*F*
A	32,160	1	32,160	12.99
B	700,877	1	700,877	283.12
C	192,080	1	192,080	77.59
D	8862	1	8862	3.58
Error	185,337	75	2476	
Total	1,119,646	79		

A: number of precedence constraints

B: processing times

C: setup times

D: delay times for precedence constraint

problems (Lee et al. 2007).The SA algorithm adopted in this research is similar to the proposed VNS. The difference is that the SA uses a single operator for searching neighborhood solutions. Because the proposed VNS uses five operators for searching neighborhood solutions, this research adopts five SA algorithms, called SA1, SA2, SA3, SA4 and SA5 respectively, for comparison. The results are shown in the third to seventh columns of Table 2.

In Table 2, it is shown that the proposed VNS provides the best average results for all scenarios. The average performance improves on the SA algorithms by between 0.99 and 17.65%. All problems were tested on an Intel Core i7 3.6 GHz personal computer and the computation times were less than 10 s.

This study also determines the VNS performance. According to the experimental structure in Table 1, there are 4 factors. The analyses of variance for all factors are shown in Table 3. It can be seen that the scheduling result for VNS clearly affected by number of precedence constraints, processing times and setup times (Factors A, B, and C). However, it is found that the effect of delay times (Factors D) is not significant at the $\alpha = 0.05$. That means the proposed VNS can provide a robust schedule, no matter what the length of delay times is, when precedence constraints are changed.

6 Conclusion

This research deals with a single machine scheduling problem in which two constraints are taken into consideration. The first is sequence-dependent setup times, which are common in the literature. The second is delayed precedence constraints which are rare in literature. In order to optimize the proposed single machine scheduling problem, a makesapn calculation model is proposed. When given a feasible sequence of jobs, the proposed model can calculate completion times of all jobs. Then a VNS which comprises four phases is proposed for optimizing the jobs sequence. The first phase uses the data of setup times to generate an initial solution,

and then selects one of five operators to search neighbourhood solutions in the second phase. Because the initial solution and the newly searched neighbourhood solution may not be feasible, all solutions have to be modified in the third phase. If the modified neighbourhood solution is better, then it will replace the original solution. However, in phase four a worse modified neighbourhood solution may still replace the original solution. By testing 16 randomly generated scenarios, the results show that the performance of the proposed VNS is better than that of the other five SA algorithms, and it also provides robust schedules for different lengths of delay time. Moreover the results are found in less than 10 s. Therefore the proposed VNS is efficient and effective in solving the single machine scheduling problems.

Acknowledgements This work was supported, in part, by the Ministry of Science and Technology of Taiwan, under Grant MOST 105-2221-E-131-025-.

References

Balas E, Lenstra JK, Vazacopoulos A (1995) The one-machine problem with delayed precedence constraints and its use in job scheduling. Manage Sci 41:94–109

Brucker P, Hilbig T, Hurink J (1999) A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. Discrete Appl Math 70:247–266

Chen P, Huang HK, Dong XY (2010) Iterated variable neighbourhood descent algorithm for the capacitated vehicle routing problem. Expert Syst Appl 37:1620–1627

Cheng TCE, Janiak A (1994) Resource optimal control in some single-machine scheduling problems. IEEE Trans Autom Control 39:1243–1246

Cheng TCE, Kovalyov MY, Shakhlevich NV (2006a) Scheduling with controllable release dates and processing times: makespan minimization. Eur J Oper Res 175:751–768

Cheng TCE, Kovalyov MY, Shakhlevich NV (2006b) Scheduling with controllable release dates and processing times: total completion time minimization. Eur J Oper Res 175:769–781

Choi BC, Yoon SH, Chung SJ (2007) Single machine scheduling problems with resource dependent release times. Comput Oper Res 34:1988–2000

Choobineh FF, Mohebbi E, Khoo H (2006) A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. Eur J Oper Res 175:318–337

Davari M, Demeulemeester E, Leus R, Nobibon FT (2016) Exact algorithms for single-machine scheduling with time windows and precedence constraints. J Sched. doi:10.1007/s10951-015-0428-y

Finta L, Liu Z (1996) Single machine scheduling subject to precedence delays. Discrete Appl Math 70:247–266

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discret Math 5:287–326

Hansen P, Mladenovic' N (2001) Variable neighbourhood search: principles and applications. Eur J Oper Res 130:449–467

Janiak A (1991) Single machine scheduling problem with a common deadline and resource dependent release dates. Eur J Oper Res 53:317–325

Janiak A (1998) Single machine sequencing with linear models of release dates. Naval Res Logist 45:99–113

Jula P, Kones I (2013) Continuous-time algorithms for scheduling a single machine with sequence-dependent setup times and time window constraints in coordinated chains. Int J Prod Res 51(12):3654–3670

Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Koulamas C (2010) The single-machine total tardiness scheduling problem: review and extensions. Eur J Oper Res 202:1–7

Kuo Y (2010) Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. Comput Ind Eng 59:157–165

Lawler EL (1978) Sequencing jobs to minimize total weighted completion time subject to precedence constraints. Algorithmic Asp Comb 2:75–90
Lee DH, Cao Z, Meng Q (2007) Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. Int J Prod Econ 107:115–124
Li CL (1994) Scheduling with resource-dependent release dates—a comparison of two different resource consumption functions. Naval Res Logist 41:807–819
Li CL, Swell EC, Cheng TCE (1995) Scheduling to minimize release-time resource consumption and tardiness penalties. Naval Res Logist 42:949–966
Pereira J (2016) The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective. Comput Oper Res 66:141–152
Pinedo M (1995) Scheduling: theory, algorithms, and systems. Prentice Hall, Englewood Cliffs
Potts CN (1985) A lagrangean based branch and bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time. Manage Sci 31(10):1300–1311
Tanaka S, Sato S (2013) An exact algorithm for the precedence-constrained single-machine scheduling problem. Eur J Oper Res 229(2):345–352
Vahdani B, Zandieh M (2010) Scheduling trucks in cross-docking systems: robust meta-heuristics. Comput Ind Eng 58:12–24
Ventura JA, Kim D, Garriga F (2002) Single machine earliness-tardiness scheduling with resource-dependent release dates. Eur J Oper Res 142:52–69
Wikum ED, Llewellyn DC, Nemhauser GL (1994) One-machine generalized precedence constrained scheduling problems. Oper Res Lett 16:87–99
Yang T, Peters BA, Tu M (2005) Layout design for flexible manufacturing systems considering single-loop directional flow patterns. Eur J Oper Res 164:440–455