Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Simultaneous selection and scheduling with sequence-dependent setup times, lateness penalties, and machine availability constraint: Heuristic approaches

Mohammad Hossein Zarei[a], Mehdi Davvari[b], Farhad Kolahan[c] and Kuan Yew Wong[d*]

[a]*Escuela Técnica Superior de Ingenieros Industriales, Universidad Politécnica de Madrid, C/Jose´ Gutie´rrez Abascal, 2, 28006 Madrid, Spain*
[b]*Department of Automotive Engineering, Islamic Azad University, Khomeini Shahr Branch, Isfahan, Iran*
[c]*Department of Mechanical Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran*
[d]*Department of Manufacturing and Industrial Engineering, Faculty of Mechanical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia*

C H R O N I C L E

Article history:
Received April 22 2015
Received in Revised Format
June 29 2015
Accepted June 30 2015
Available online
July 2 2015

Keywords:
Job selection
Job scheduling
Earliness
Tardiness
Lateness
Sequence-dependent setup time
Scatter search
Simulated annealing

A B S T R A C T

Job selection and scheduling are among the most important decisions for production planning in today's manufacturing systems. However, the studies that take into account both problems together are scarce. Given that such problems are strongly NP-hard, this paper presents an approach based on two heuristic algorithms for simultaneous job selection and scheduling. The objective is to select a subset of jobs and schedule them in such a way that the total net profit is maximized. The cost components considered include jobs' processing costs and weighted earliness/tardiness penalties. Two heuristic algorithms; namely scatter search (SS) and simulated annealing (SA), were employed to solve the problem for single machine environments. The algorithms were applied to several examples of different sizes with sequence-dependent setup times. Computational results were compared in terms of quality of solutions and convergence speed. Both algorithms were found to be efficient in solving the problem. While SS could provide solutions with slightly higher quality for large size problems, SA could achieve solutions in more reasonable computational time.

1. Introduction

When several jobs or projects are put forward, a manufacturing company may logically tend to choose the ones which deliver the highest return. However, selecting the jobs solely according to their revenues does not guarantee the company's profitability. Selected jobs must be scheduled with respect to the limited resources of the firm such that the deadlines are met and the lateness penalties are avoided or minimized. Therefore, the methods that can consider different scheduling combinations of jobs at the time of selection are crucial. Such methods enable the managers to choose the best subset of jobs with the consideration of their due dates and also provide a resolution to investigate two interrelated problems in a single context.

* Corresponding author. Tel: +607-5534691
E-mail: wongky@fkm.utm.my (K. Y. Wong)

doi: 10.5267/j.ijiec.2015.7.001

Lateness is usually defined as the algebraic difference between the due date and actual completion time of a job, regardless of their mathematical sign. It is either in the form of tardiness, which considers only positive difference (completion after due date), or earliness, which corresponds to negative deviation from completion time (ahead of due date) (Conway et al., 2012). According to the just-in-time (JIT) philosophy, a product or service should be produced at the time needed and in the quantities required; otherwise a penalty is incurred in proportion to the amount of lateness (Shingo, 1989). Assigning such penalties prevents extra costs to be imposed to the company by decreasing the amount of inventory of finished products (for earliness) as well as avoiding the loss of goodwill and customer dissatisfaction (for tardiness) (Behnamian & Zandieh, 2013).

Setup time is an integral part of processing time which is defined as the period of time that it takes to unload a job from the machine, adjusting the machine for processing the next job, and installing the next job on the machine (Kolahan & Liang, 1998). For simplification purposes, the setup times are considered to be sequence-independent in many studies. In reality, however, a large percentage of production schedulers reported that they have frequently encountered sequence-dependent setup times (Luo & Chu, 2007). Hence, managing the jobs with sequence-dependent setups is a critical factor in enhancing the performance of manufacturing systems.

Solving the sole problem of selection or scheduling has been the purpose of numerous studies. For the selection problem, most of the studies have taken advantage of exact methods such as integer programming (Yavuz & Captain, 2002), scoring methods (Henriksen & Traynor, 1999), and multiple criteria decision making tools such as analytic network process (Meade & Presley, 2002). The scheduling problem for single machine has been addressed using both exact and heuristic approaches. In a study by Chen et al. (2007), the makespan of a single machine together with the delivery time to a single customer area are minimized using a two-phase integer programming approach. A branch-and-bound algorithm has been presented by Luo and Chu (2007) that could deal with the problem of scheduling N jobs on a single machine with sequence-dependent setup times to minimize the maximum tardiness. They have implemented their approach on different instances of jobs to evaluate its efficiency. The main shortcoming of their method was that it considerably lost its credibility in solving large size problems (97.3% solved for 15-job instances and only 37.3% solved for 30-job instances).

Most enumeration methods trying to solve large scale scheduling examples encounter the same problem as the previous study. Since the problem of scheduling jobs on a single machine with sequence-dependent setups to minimize the lateness is shown to be strongly NP-hard (Du & Leung, 1990; Low et al., 2008; Baker, 1974), classical methods such as integer programming or branch-and-bound are unable to achieve an optimal solution in a reasonable running time for such problems. Consequently, application of heuristic optimization algorithms that can produce optimal (or near-optimal) solutions at a considerably lower computational time has become widespread in the past decades. In the study of Chen et al. (2007), after proposing the integer programming approach, two heuristic algorithms have been presented to solve the problem. The authors have concluded that for large size problems their heuristics are more efficient compared to their proposed integer programming model. A comprehensive review of the literature about the scheduling problem with setup times or costs constraints can be found in Allahverdi et al. (2008).

Few studies have been conducted on the combined problem of selection and scheduling. Kyparisis and Douligeris (1993) were the first who have taken these two problems into account simultaneously. They have extended the branch-and-bound scheduling algorithm of Emmons (1975) to include the optimal selection of jobs. The objective of their study was to minimize the total flow time with minimum number of tardy jobs. The problem with their modified branch-and-bound was that it could not be applied once the maximum number of non-tardy jobs exceeds the number of selected jobs. In addition, the amount of branching increased significantly by growing the size of the problem. In the same vein, Ahonen et al. (2009) have modeled the problem of organizing customer tasks in a virtual organization as a flexible flow shop problem in which one machine among the available machines in a service group must be selected and the execution order of jobs assigned to the machine must be determined. The objective

function was aimed at minimizing the total cost of selection as well as the makespan of the jobs sequence. In order to solve the problem, they have proposed a tabu search and a simulated annealing algorithm with variable neighborhood search and applied them to a cutting stock example.

This research casts a light upon the problem of simultaneous job selection and scheduling for a single machine to maximize the net profit. The problem has been solved by two heuristic algorithms, scatter search (SS) and simulated annealing (SA). In formulating the problem, lateness penalties, sequence-dependent setup times, and machine availability time are considered as the main constraints. Our survey of literature shows that the problem has not, to date, been discussed with the scope of this paper, despite evidence of its increasing use in the manufacturing systems.

The rest of the paper is organized as follows. Next section states and formulates the problem under consideration. SS and SA, the two solution procedures used in this research are introduced in Section 3. In section 4, the computational results of the study are presented. Section 5 provides a comparison and discussion on the results of investigated problems with different sizes. Finally, section 6 concludes the paper with future research recommendations.

2. Problem Statement and Formulation

2.1. Problem Statement

Consider J jobs are offered to a company for processing on a single machine. The company is free to choose any number of jobs. Once a job is selected, it should be processed on the machine and should be completed on a specific due date. Any earliness or tardiness results in penalty and thus is unfavorable. A sequence-dependent setup time is considered for each job depending on the immediate previous job. This period of time is required for changing the tools, parts, dies and adjusting the machine feed rate, speed, etc. The sequence-dependent setup times are assumed asymmetrically. Moreover, the machine is available at a limited extension of time.

Completion of each job earns a specific amount of revenue for the company while it incurs a processing cost. In addition to the processing cost, probable earliness and tardiness penalties are to be paid. The net profit is calculated by subtracting all the costs and possible penalties of a job from its revenue. The objective is to maximize the net profit resulted from scheduling a selected subset of jobs.

For the system under study, the following features and assumptions are considered.

- A number of jobs are available to be processed on the machine.
- Preemption is not allowed which means the jobs cannot be interrupted once started.
- Any job that is selected to be processed first has no setup time. In other words, all of the jobs are ready to be processed at the beginning.
- Each job has a set of specific sequence-dependent setup times, each of which pertains to the job that would precede it in the sequence. The setup times are asymmetric.
- The machine can process one job at a time only.
- Idle time is not permitted during the availability time of the machine.

2.2. *Mathematical Model*

In order to formulate the model, the following notations are used:

j: Index of job number
$[j]$: Index of position for the j-th job in the sequence
Z: Net profit
n: Number of jobs

t_j: Processing time of job j
d_j: Due date of job j
α_j: Earliness penalty of job j per unit of time
β_j: Tardiness penalty of job j per unit of time
mc_j: Processing cost of job j per unit of time
be_j: Completion revenue for job j
E_j: Actual earliness of job j
T_j: Actual tardiness of job j
c_j: Actual completion time of job j
A: Availability time of the machine
X_j: 1, if job j is selected; 0, otherwise

A maximization objective function is exploited for the problem as shown in Eq. (1). The objective function consists of three cost components for each job, including total processing cost ($mc_j t_j$), total earliness penalty ($\alpha_j E_{[j]}$), and total tardiness penalty ($\beta_j T_{[j]}$). These costs are subtracted from the job's revenue (be_j) to give the net profit. Since the completion revenue of each job has a fixed value, the function tries to maximize the net profit by reducing the cost components (processing cost and lateness penalties).

$$max\ Z = \sum_{j=1}^{n} (\ be_j - \ (mc_j t_j + \alpha_j E_{[j]} + \beta_j T_{[j]}))\ X_j \quad (1)$$

subject to:

$$t_{[1]}X_1 + t_{[2]}X_2 + \cdots + t_{[n]}X_n \leq \mathrm{A} \quad (2)$$

$$E_{[j]} \geq 0 \quad (3)$$

$$T_{[j]} \geq 0 \quad (4)$$

$$X_j = 0 \text{ or } 1, j = 1,..., n \quad (5)$$

where E_j and T_j are calculated as follows:

$$E_j = \max(0\ , d_j - c_j) \quad (6)$$

$$T_j = \max(0\ , c_j - \ d_j) \quad (7)$$

As the constraints of this model, Eq. (2) guarantees that the sum of processing time for the selected subset of jobs does not exceed the availability time of the machine. Eq. (3) and Eq. (4) ensure that the values for earliness and tardiness are not negative and Eq. (5) represents the type of decision variable. Solving the problem stated and formulated above using exact methods is cumbersome even for small size problems. Hence, we propose two heuristic procedures in the following section to solve the problem.

3. Solution Procedures

3.1. Scatter Search

Scatter search (SS) is a heuristic algorithm that was first introduced by Glover (1977) as an integer programming heuristic. After the application was extended to nonlinear, binary and permutation problems in 1994 by Glover, the algorithm became popular and its application has spread to a wide variety of optimization problems (Glover, 1994; Martí et al., 2006). The search method of SS has a systematic structure which differentiates it from other algorithms with random search design such as genetic algorithm (Martí, 2006). The algorithm searches through the solution space based on a diversification approach which enables it to escape from local optimums and presents optimal (or in the vicinity of the optimum) solutions. The mechanisms of SS are not limited to a single uniform design but

help to explore the effective strategic possibilities for a particular implementation. The fundamental structure of the algorithm consists of five methods explained in the following subsections.

3.1.1. Diversification Generation Method

The diversification generation method generates a collection of initial solutions. An arbitrary solution, sometimes called as seed solution, is used as an input at the beginning of the run. Diversification generation method also refreshes the reference set when the algorithm is restarted. The method is used together with improvement method which is described in the next subsection.

3.1.2. Improvement Method

The purpose of improvement method is generating higher quality solutions by exploring and evaluating neighbor solutions. It uses the solutions achieved from the diversification generation or combination method to transform each solution into one or more improved ones. Note that neither the input nor the output solutions are necessarily feasible. Amongst the five methods of SS, improvement is the only arbitrary method.

3.1.3. Reference Set Update Method

Reference set update method is employed to construct and keep a reference set containing a definite number of best solutions found. The size of the reference set, b, is one of the adjusting parameters of the algorithm and is chosen relatively small (e.g. about 20). Whilst different update methods have been used for the algorithm (see Martí et al., 2006), 2-tier reference set is used in this research. According to this type of design, two reference sets are constructed: $RefSet_1$ and $RefSet_2$. The latter keeps a number of high quality solutions, while the former contains diverse solutions with the furthest distance from high quality solutions. By doing this, not only high quality solutions are maintained, but also the reference set is updated with highly diverse solutions which impedes the algorithm to become homogenous by admitting merely similar high quality solutions.

3.1.4. Subset Generation Method

This method uses the existing solutions in the reference set to generate a subset of solutions that can be used as a basis for the solution combination method (next method). Since the combination method is not confined to the combination of just two solutions, the subset generation method should also be able to generate subsets of different sizes.

3.1.5. Solution Combination Method

Having generated appropriate subsets through subset generation method, the subsets must be transformed into combined solution vectors using solution combination method. The design of SS employed in this research uses a competitive solution combination method based on the objective function. This type of combination assigns a higher choice probability to the combinations that can produce better solutions according to their objective function values. A general sketch of the SS algorithm is presented in Fig. 1.

3.2. Simulated Annealing

Simulated Annealing (SA) is an optimization heuristic algorithm that was first proposed by Kirkpatrick (1984). The idea behind the algorithm is the annealing process used in metallurgy. During this process, a metal is heated until it reaches the temperature of liquefying, and then cooled down slowly such that the metal atoms find a more stable state than their initial situation. The process goes on in a controlled manner until the metal solidifies back completely. The real world optimization problems can be solved in the same way. At each stage, the SA algorithm generates a new state and compares its energy with the

energy of the current state. The algorithm moves to the new state if it is found better; otherwise, a transition probability equation shown in Eq. (8) is used to change the current state (Chen & Chien, 2011).

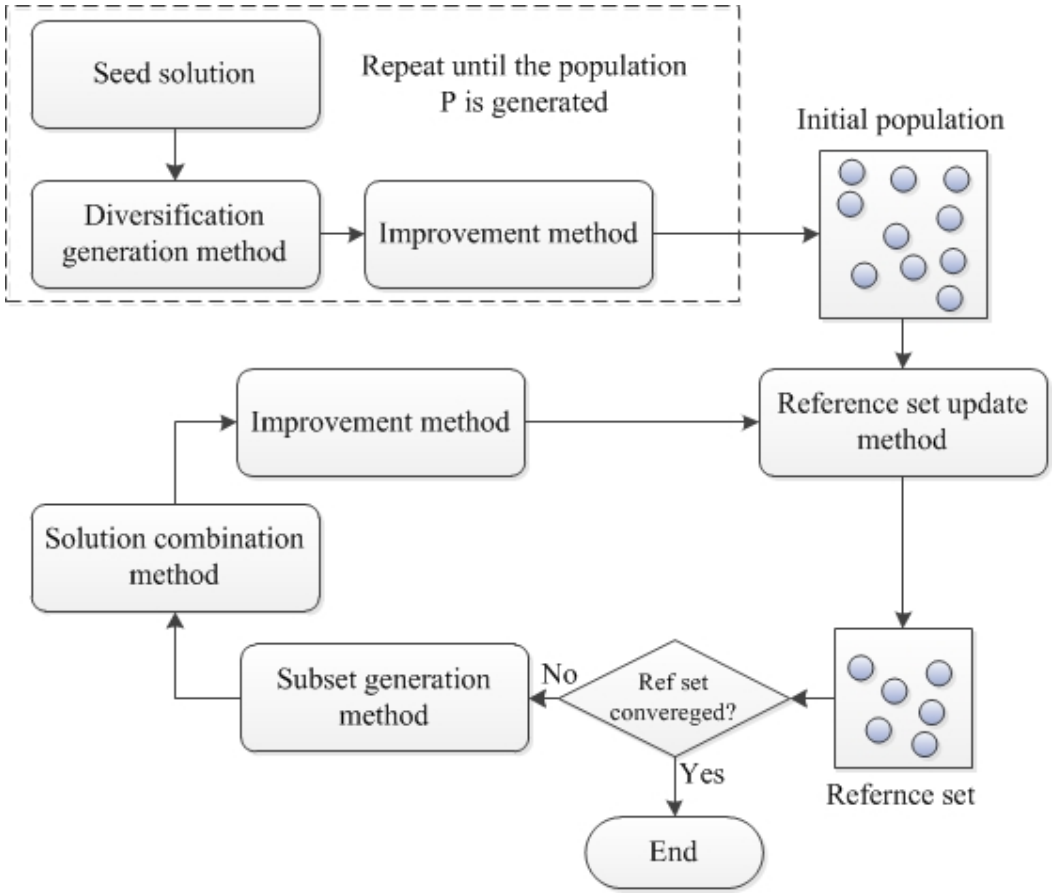

Fig. 1 Flowchart of SS Algorithm

$$P_a = e^{-\frac{k\Delta E}{T}} \tag{8}$$

where k is the Boltzmann constant, T is the current temperature of the system, and ΔE is the difference between the energy levels of the system that can be obtained using Eq. (9).

$$\Delta E = E(S') - E(S) \tag{9}$$

In Eq. (9), S represents the current state and $S'$is the new state of the system. As the algorithm proceeds, the temperature decreases and the search through the solution area is narrowed down. Although the algorithm accepts solutions that result in improvement, for a comprehensive search of the solution space, it is capable of adopting bad solutions as well (Arif, 2012; Mosavi & Shiroie, 2012). The termination criterion can be running a certain number of iterations, reaching a specified running time or cooling down to a predetermined temperature. We have modified the algorithm to accommodate the requirements of the twofold selection and scheduling problem.

3.2.1. Generating an Initial Solution

The algorithm generates the random permutation (randperm(n)), where n is the number of jobs. Then, the jobs are selected from the beginning of the permutation until job j. The selection is performed such that the machine availability constraint is not violated.

3.2.2. Neighbor Generation

For generating neighbors, the algorithm makes two by two replacements for all the jobs existing in the random permutation (randperm(n)). Then again, the jobs are selected from the beginning of the new permutation until job j so that the machine availability constraint is kept. The flowchart of the SA algorithm is presented in Fig. 2.

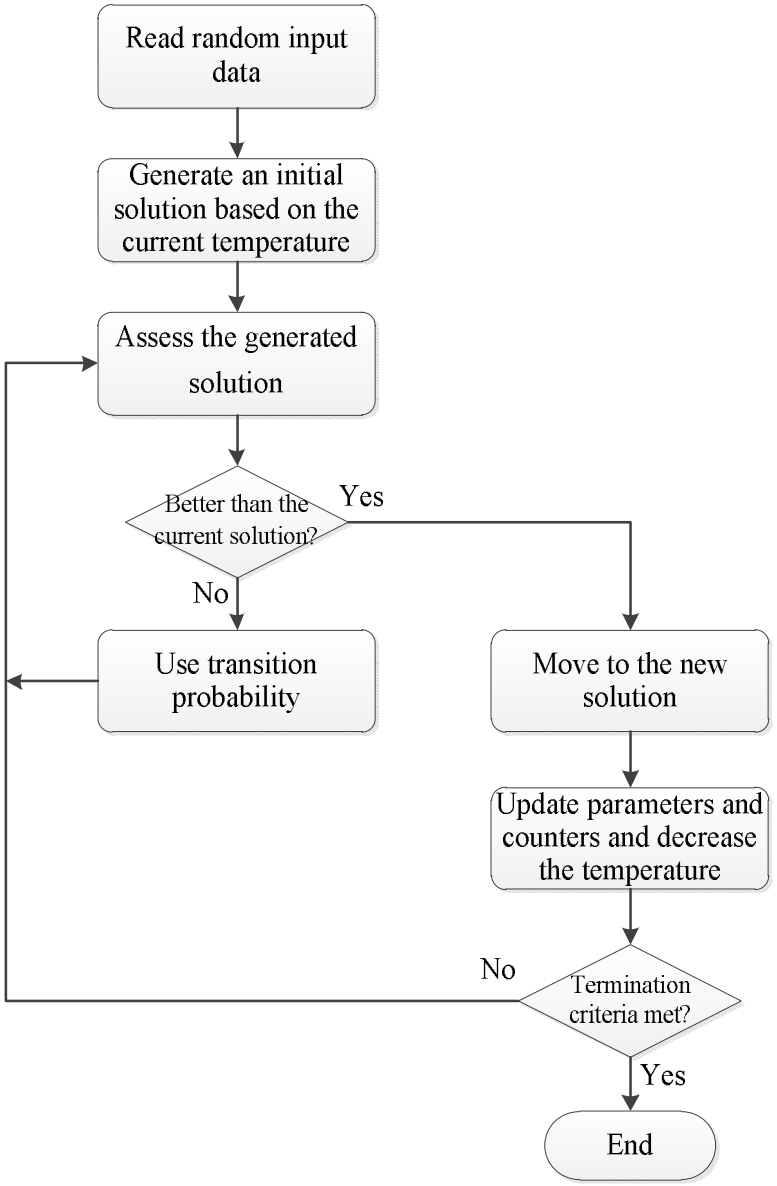

Fig. 2 Flowchart of SA algorithm

4. Computational Results

In order to evaluate the proposed approaches, the SS and SA algorithms were implemented in *MATLAB R2009a* computer software and applied to the problems of different sizes. In this section, the details of a 10-job instance problem are described to provide a walk through on the problem specifications and the performance of the proposed approaches. Then, the 80-job instance problem is presented to show the performance of the proposed heuristics to solve large size real world problems. In the next section, the results of all investigated problems are reported and compared.

4.1. The 10-Job Instance Problem

The values for the job specifications and the asymmetric sequence-dependent setup times of the 10-job instance problem are shown in Table 1 and Table 2, respectively. The initial parameters were adjusted with respect to the size of the problem: The initial temperature and cooling rate of the SA algorithm were set to 1200 and 0.5, respectively while the reference set size of the SS algorithm was adjusted to 5. The termination criterion for both algorithms was reaching 150 seconds of running time.

Table 1
Job Specifications for the 10-job Instance Problem

	Jobs									
Parameters	1	2	3	4	5	6	7	8	9	10
Processing time (unit of time)	5	4	8	7	3	2	10	10	7	8
Due date (unit of time)	15	25	10	10	3	20	40	60	30	18
Tardiness penalty (per unit of time in $)	0.5	1	2	4	0.2	2.5	1	1.5	2	1
Earliness penalty (per unit of time in $)	0.3	2	1	0.5	1	4	0.2	0.4	1.5	0.2
Processing cost (per unit of time in $)	4	3	6	6	2	1	7	8	5	7
Completion revenue ($)	350	100	50	250	300	150	20	450	120	80

Table 2
The Sequence-dependent Asymmetric Setup Times for the 10-job Instance Problem

	Next Job									
First Job	1	2	3	4	5	6	7	8	9	10
1	-	0.1	0.8	0.7	0.4	1.2	1.1	0.5	0.3	0.8
2	1.4	-	0.3	0.7	2	0.2	0.4	1.5	1.6	0.1
3	0.7	1.1	-	1.6	0.5	0.5	0.1	0.2	1.6	0.2
4	1	0.3	1.7	-	0.2	2	0.2	0.3	0.7	0.4
5	2	0.6	2	0.6	-	0.4	0.3	0.2	0.3	1.5
6	0.8	0.4	0.1	0.6	1.6	-	0.8	0.2	0.6	0.2
7	1.1	2	0.7	1.8	0.3	1.8	-	0.1	0.8	0.4
8	0.8	0.3	2.9	0.1	0.1	0.2	0.2	-	0.3	0.6
9	1.6	0.4	0.9	0.3	2	0.6	1.4	1.2	-	0.3
10	0.7	1.8	0.3	1.8	0.2	0.1	0.8	0.4	1.7	-

After running the algorithms, they both produced the same results. The sequence of {5 - 4 - 1 - 6 - 8} was selected and scheduled resulting in the net profit of $1334. The complete enumeration of all possible sequences also resulted in the same solution. This proves that both algorithms are capable of reaching optimal solutions for small size problems.

4.2. The 80-job Instance Problem

In order to evaluate the performance of the algorithms in solving real life problems, a problem with 80 jobs was solved. Such large size problems are very likely to be confronted in daily manufacturing practices. The ranges of input parameters within which the job specifications vary are shown in Table 3.

Table 3
The Ranges of Job Specifications for the 80-job Instance Problem

Processing time (unit of time)	Due date (unit of time)	Job completion revenue ($)	Processing cost (per unit of time in $)	Lateness penalties (per unit of time in $)		Setup time (unit of time)	Machine availability time (unit of time)
				Earliness	Tardiness		
4 - 35	10 - 480	100 - 1000	5 - 15	0 - 2.4	0.6 - 3	0.1 - 3	960

The parameters of the algorithms need to be tuned prior to their run. Parameter tuning (e.g. tuning of cooling rate for the SA algorithm) has been found to be drastically influencing on both the efficiency and

effectiveness of heuristics (Lessmann et al., 2011). Appropriate calibration of parameters boosts the capability of heuristic algorithms to find optimal or sub-optimal solutions in a rational amount of time (Akbaripour & Masehian, 2013) especially for problems with larger size. For the SS algorithm, since the reference set size was the only parameter to be tuned, different values were tested and the pertaining changes in the objective function were observed. Eventually, the optimum value of reference set size was found to be 19. When there is more than one parameter involved, such as the case of our SA algorithm with two parameters: initial temperature and cooling rate, different parameter tuning methods can be used. Design of experiment (DOE) is one of the best and most frequently used approaches for parameter tuning. Each experiment within DOE collects information resulting from purposeful changes made to parameters of a process so that the reasons for changes in the objective function values are identified (Montgomery, 2012). Through these experiments, the largest information possible is collected with the least number of experiments and hence the optimal settings for the parameters are found.

In this paper, DOE was adopted to investigate the effect of SA parameters on the response obtained from the objective function. Each parameter had two levels of high and low denoted by (+1) and (-1), respectively and a 2^2 full factorial design was developed. The levels and values of parameters are shown in Table 4. The responses of the algorithm to different combinations of parameters' levels are shown in Table 5. According to Table 5, the best experimental result was achieved from the combination of +1 and +1 for initial temperature and cooling rate, respectively. Hence, these parameters were respectively tuned to 8000 and 0.9999 for the initiation of the algorithm.

Table 4
Levels of Parameters for the SA Algorithm

	Level	
Parameter	Low (-1)	High (+1)
Initial temperature	6000	8000
Cooling rate	0.6666	0.9999

Table 5
Experimental Results of the SA Algorithm for Parameter Tuning

Run order	Initial temperature	Cooling rate	Objective function value (response)				
			Revenue ($)	Processing cost ($)	Tardiness penalty ($)	Earliness penalty ($)	Net profit ($)
1	+1	+1	24182	9217	469	503	13992
2	+1	-1	24131	9596	424	840	13271
3	-1	+1	24095	9513	394	806	13382
4	-1	-1	23298	9633	452	777	12436

Considering the size of the problem, reaching 2500 seconds was set as the termination criterion and both algorithms were run 10 times with the same starting schedule in each run.

Table 6
Summary of Results for the 80-job Instance Problem

	SA			SS		
Cost components	Initial scheduling plan	Final scheduling plan	Improvement (%)	Initial scheduling plan	Final scheduling plan	Improvement (%)
Revenue ($)	15992	24182	51.2	15992	24434	52.7
Processing cost ($)	9372	9217	1.6	9372	9495	-1.3
Tardiness penalty ($)	13286	469	96.4	13286	387	97
Earliness penalty ($)	10140	503	95	10140	449	95.5
Net profit ($)	-16806	13992	183.2	-16806	14103	183.9

For the best run, Table 6 shows the cost components of the initial and final scheduling plans. The table shows that whilst the processing cost of the final scheduling plan shows minor changes compared to the initial scheduling plan, the revenue and weighted lateness penalties, however, have improved considerably. Consequently, the net profit of the final plan shows an upsurge of more than 183% for both algorithms. For this problem, the SA algorithm could converge in 800 seconds, while it took 1500 seconds for the SS algorithm to converge. The latter, though, could present slightly better results in the long run. Hence, we present the details of the best solution found by SS in Table 7.

Table 7
Details of the Best Solution Found by SS for the 80-job Instance Problem

Order of the job in the sequence	Job number	Completion time (time unit)	Lateness (time unit)	Lateness penalty incurred ($)	Processing cost ($)	Revenue ($)
[1]	22	6.0	-19.0	5.7	36	374
[2]	10	23.4	-11.6	24.3	85	600
[3]	37	43.6	-5.4	9.7	300	600
[4]	15	70.6	-14.4	34.5	275	455
[5]	35	86.9	+1.9	3.4	208	259
[6]	26	104.7	-3.3	4.9	119	232
[7]	45	114.6	+0.6	0.9	56	434
[8]	77	120.1	+0.1	0.0	60	180
[9]	7	136.8	+29.8	35.7	126	277
[10]	28	157.1	+2.1	5.6	108	208
[11]	32	176.8	-3.2	2.8	285	455
[12]	18	208.3	+8.3	4.4	450	950
[13]	52	235.5	-40.5	24.3	175	335
[14]	50	244.7	-7.3	6.5	117	306
[15]	1	264.2	+0.2	0.2	190	950
[16]	40	294.7	+23.7	14.2	330	416
[17]	68	317.4	-6.6	3.9	308	434
[18]	75	327.8	+1.8	3.7	50	335
[19]	30	356.0	+6.0	12.6	392	500
[20]	69	366.0	+9.0	10.8	88	800
[21]	31	392.8	+2.8	4.2	275	800
[22]	36	416.1	+84.1	75.7	322	424
[23]	63	422.9	-2.1	3.7	60	416
[24]	59	440.3	+8.3	22.4	240	424
[25]	38	458.6	+18.6	22.3	90	251
[26]	9	471.8	-8.2	4.9	72	232
[27]	13	477.3	+1.7	3.0	75	500
[28]	4	513.7	+65.7	59.1	210	258
[29]	57	522.5	+2.5	6.0	104	1000
[30]	74	531.8	+2.8	5.8	108	200
[31]	46	560.0	-260.0	78.0	270	800
[32]	42	566.0	-4.0	1.2	40	231
[33]	51	577.3	-142.7	85.6	110	200
[34]	53	584.3	+1.3	2.3	60	162
[35]	29	590.5	-19.5	11.7	25	179
[36]	14	619.7	-30.3	63.6	336	800
[37]	66	625.6	-14.4	12.9	28	369
[38]	61	646.7	-3.3	5.9	220	251
[39]	60	670.5	+0.5	0.9	322	600
[40]	80	687.6	-12.4	14.8	255	1000
[41]	54	720.4	-8.6	20.6	160	180
[42]	49	751.0	-6.0	7.2	420	496
[43]	72	776.3	+0.3	0.7	275	496
[44]	23	796.1	+46.1	41.4	108	1000
[45]	71	812.7	+12.7	7.6	208	445
[46]	73	843.0	+6.0	7.2	210	306
[47]	27	863.5	+3.5	3.1	200	600
[48]	33	872.3	-7.7	4.6	96	256
[49]	56	890.1	+2.1	5.0	204	400
[50]	6	898.1	-1.9	4.5	60	1000
[51]	39	906.1	-3.9	9.3	84	186
[52]	48	932.8	+8.8	15.8	275	445
[53]	19	941.0	+1.0	2.4	35	150
[54]	24	956.4	+8.4	10.0	180	277
Total		956.4	-638 *(total earliness)* +359 *(total tardiness)*	449.07 *(total earliness penalty)* 387.33 *(total tardiness penalty)*	9495	24434

According to the table, 54 jobs out of the 80 candidate jobs were selected and scheduled. The first three columns show the order of job, job number, and its completion time. The fourth column gives the lateness of the scheduled jobs which is the difference between the scheduled completion time (c_j) and the job's due date (d_j). Negative values indicate that the job was processed sooner than the due date (earliness), while positive values indicate completion after the due date (tardiness). The fifth column shows the lateness penalty imposed according to the type and magnitude of the job's lateness. Note that the jobs with high lateness times (in column four) had relatively small penalties per time unit and vice versa. This shows that the algorithm levels the variation of incurred penalties (column five) and avoids fluctuated results. Calculations show that about 70% of the incurred penalties are less than $20. The sixth and the seventh columns depict the processing cost for each job and its revenue, respectively. The total processing cost of this sequence is $9495 resulting in $24434 of revenue.

5. Discussion

In this section, we provide a discussion on the findings of the study and delineate the practical implications on the application of the proposed algorithms. In our experiments, we studied four problems with 10, 20, 40, and 80 jobs. Table 8 depicts the average convergence times and the net profits for the investigated problems.

Table 8
Comparison of Convergence Time and Net Profit of Results

Number of jobs	SS		SA	
	Convergence time (second)	Net profit ($)	Convergence time (second)	Net profit ($)
10	1.8	1334	0.2	1334
20	10.9	3732	1.6	3732
40	54.9	7193	19.5	7193
80	1500	14103	800	13992

5.1. Convergence Speed and Quality of Solutions

According to the results presented in Table 8, the convergence rate of SA was higher in comparison with SS for all the examples. This would be due to the mechanism of searching and moving through the solution space towards the optimum (or sub-optimum) solution. SA is a single agent algorithm that evaluates only one neighbor at each iteration. On the other side, SS examines a certain number of solutions, improves their quality, combines them, and then repeats the improvement again. Then, the algorithm moves to the best generated solution. Obviously, this procedure of searching covers a larger portion of the solution space but at the cost of higher computational times and lower convergence rate.

Both algorithms produced solutions of the same quality for the first three examples (10, 20, and 40 jobs). However, for the example problem with 80 jobs, SS generated the results with slightly higher quality compared to SA. That is because SS explores more solutions at each iteration and thus the chance of finding a better solution is higher. As the size of the problem grows, it is expected that the quality of solutions produced by SS improves. Moreover, since our proposed SS examines neighbors of high quality solutions according to Euclidean distance, the algorithm hardly traps in local optimums.

Whilst both algorithms could solve the problems of different sizes in reasonable running times with good quality solutions, their speed and quality in solving the problems were not identical, especially for large size problems. Therefore, for the problems in which the computational times are of great importance, SA may be preferred due to its higher convergence rate. Small daily problems and short-term production planning are among the instances where the quality of solutions can be of secondary consideration. Whereas, if higher quality solutions are required (e.g. larger scheduling problems, long-term planning, and planning during the product design or development stage), application of SS is recommended.

5.2. Changing of the Cost Components

Every scheduling method should be flexible enough to meet the requirements of a changing work environment. Very often, the elements of the objective function such as costs or penalties vary during the planning horizon. The algorithms should be able to deal with the changes and adapt to the new situation. As an example, let us assume the case of a company whose managers intend to increase their market penetration and prevent lost sales by minimizing delivery tardiness, a strategy which is in line with just-in-time (JIT) philosophy. The input parameters of the algorithms can be adjusted to account for higher tardiness penalties. In turn, the final solution found by the proposed approaches should have less total tardiness. To illustrate, we have resolved the 40-job instance problem while the tardiness penalties have increased incrementally. Fig. 3 presents the effect of increasing the tardiness penalties on the total tardiness time. As shown, the total tardiness of the solution found by the SS algorithm was initially 374.8 units of time. After doubling, tripling, and quadrupling the tardiness penalty, it decreased to 348.5, 240.3, and 222.6 units of time, respectively. Therefore, by increasing the importance of tardiness, the algorithm tends to find solutions with lower tardiness.

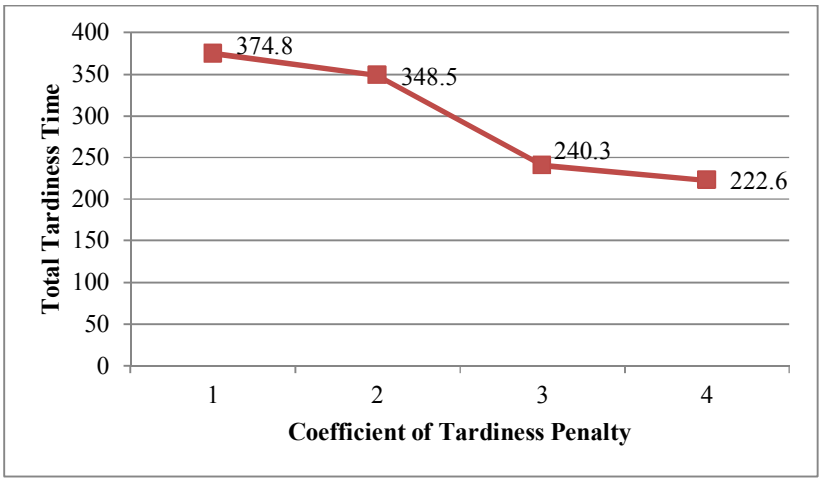

Fig. 3. The effect of increasing tardiness penalties on total tardiness time (SS)

The algorithms have dealt with changing other input parameters in the same way. For instance, when the processing costs increased, the algorithms kept the costs low by shortening the total processing time. Such purposeful changes of parameter settings help production planners to evaluate the efficiency of their scheduling methods when facing unforeseen variations after planning and ensure that these methods can properly adapt to changing environments.

6. Conclusion

The necessity of selecting from a group of available projects due to resource limitations of companies and frequent occurrence of sequence-dependent setup times pinpoint the need for studying the twofold problem of selection and scheduling. In this paper, we have addressed the problem on a single machine by proposing two efficient heuristic algorithms, SS and SA. We assumed asymmetric sequence-dependent setup times, weighted earliness and tardiness penalties, distinct processing costs and machine availability constraint for modeling the problem. Both proposed heuristics were found quite efficient in solving problems of different sizes. SS could present slightly better solutions for larger problems while SA outperformed in terms of convergence speed.

The scope of this paper can be broadened to multi-machine problems in different environments. For instance, the production system of many industries contains a bottleneck machine which determines the production rate of the entire system. Scheduling the jobs on this machine is vital to avoid delays in customer order delivery (Sioud et al., 2012). Moreover, the terms “job” and “machine” can refer not only

to their manufacturing definitions, but also to a variety of projects in different industries to be selected and scheduled. Therefore, the single machine model can be applied to a wide range of manufacturing and service industries.

Future research may include applying other heuristic algorithms such as genetic algorithm and particle swarm optimization to the problem. The problem can be modeled and optimized with different objectives such as minimizing the total weighted earliness and tardiness or minimizing the makespan with minimum number of tardy jobs.

References

Ahonen, H., de Alvarenga, A. G., & Provedel, A. (2009). Selection and scheduling in a virtual organisation environment with a service broker. *Computers& Industrial Engineering*, *57*(4), 1353-1362.

Akbaripour, H., & Masehian, E. (2013). Efficient and robust parameter tuning for heuristic algorithms. *International Journal of Industrial Engineering & Production Research, 24*(2), 143-150.

Allahverdi, A., Ng, C. T., Cheng, T. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985-1032.

Arif, S., Mohammedi, R. D., Hellal, A., & Choucha, A. (2012). A memory simulated annealing method to the unit commitment problem with ramp constraints. *Arabian Journal for Science and Engineering*, *37*(4), 1021-1031.

Baker, K.R. (1974). *Introduction to Sequencing and Scheduling*. Wiley, NY.

Behnamian, J., & Zandieh, M. (2013). Earliness and tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic. *Arabian Journal for Science and Engineering*, *38*(5), 1229-1242.

Chen, J. S., Liu, H. S., & Nien, H. Y. (2007). Minimizing makespan in single machine scheduling with job deliveries to one customer area. *International Journal of Industiral Engineering-Theory and Applilcations and Practice*, 14(2), 203-211.

Chen, S. M., & Chien, C. Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, *38*(12), 14439-14450.

Conway, R. W., Maxwell, W. L., & Miller, L. W. (2012). *Theory of scheduling*. Courier Corporation.

Du, J., & Leung, J. Y. T. (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, *15*(3), 483-495.

Emmons, H. (1975). One machine sequencing to minimize mean flow time with minimum number tardy. *Naval Research Logistics Quarterly*, *22*(3), 585-592.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, *8*(1), 156-166.

Glover, F. (1994). Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, *49*(1), 231-255.

Henriksen, A. D., & Traynor, A. J. (1999). A practical R&D project-selection scoring tool. *Engineering Management, IEEE Transactions on*, *46*(2), 158-170.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, *34*(5-6), 975-986.

Kolahan, F., & Liang, M. (1998). An adaptive TS approach to JIT sequencing with variable processing times and sequence-dependent setups. *European Journal of Operational Research*, *109*(1), 142-159.

Kyparisis, J., &Douligeris, C. (1993). Single machine scheduling and selection to minimize total flow time with minimum number tardy. *Journal of the Operational Research Society*, *44*(8), 835-838.

Lessmann, S., Caserta, M., & Arango, I. M. (2011). Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Systems with Applications, 38*(10), 12826-12838.

Low, C., Hsu, C. J., & Su, C. T. (2008). Minimizing the makespan with an availability constraint on a single machine under simple linear deterioration. *Computers& Mathematics with Applications*, *56*(1), 257-265.

Luo, X., & Chu, C. (2007). A branch-and-bound algorithm of the single machine schedule with sequence-dependent setup times for minimizing maximum tardiness. *European Journal of Operational Research*, *180*(1), 68-81.

Meade, L. M., & Presley, A. (2002). R&D project selection using the analytic network process. *Engineering Management, IEEE Transactions on*, *49*(1), 59-66.

Martí, R., Laguna, M., & Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research*, *169*(2), 359-372.

Martí, R. (2006). Scatter search—wellsprings and challenges. *European Journal of Operational Research*, *169*(2), 351-358.

Montgomery, D. C. (2012). *Design and Analysis of Experiments, 8th Edition*: John Wiley & Sons, Hoboken, NJ.

Mosavi, M. R., & Shiroie, M. (2012). Efficient evolutionary algorithms for GPS satellites classification. *Arabian Journal for Science and Engineering*, *37*(7), 2003-2015.

Sioud, A., Gravel, M., & Gagné, C. (2012). A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times. *Computers& Operations Research*, *39*(10), 2415-2424.

Shingo, S. (1989). *A study of the Toyota production system: From an Industrial Engineering Viewpoint*. Productivity Press.

Yavuz, S., & Captain, T. A. (2002). Making project selection decisions: a multi-period capital budgeting problem. *International Journal of Industrial Engineering*, *9*(3), 301-310.