Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 928312, 20 pages
doi:10.1155/2012/928312

Research Article

Variable Neighborhood Search for Parallel Machines Scheduling Problem with Step Deteriorating Jobs

Wenming Cheng,[1] Peng Guo,[1] Zeqiang Zhang,[1] Ming Zeng,[1] and Jian Liang[2]

[1] *School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China*
[2] *School of Mechanical Engineering and Automation, Xihua University, Chengdu 610039, China*

Correspondence should be addressed to Peng Guo, pengguo318@gmail.com

Received 20 February 2012; Revised 27 April 2012; Accepted 8 May 2012

Academic Editor: Jung-Fa Tsai

In many real scheduling environments, a job processed later needs longer time than the same job when it starts earlier. This phenomenon is known as scheduling with deteriorating jobs to many industrial applications. In this paper, we study a scheduling problem of minimizing the total completion time on identical parallel machines where the processing time of a job is a step function of its starting time and a deteriorating date that is individual to all jobs. Firstly, a mixed integer programming model is presented for the problem. And then, a modified weight-combination search algorithm and a variable neighborhood search are employed to yield optimal or near-optimal schedule. To evaluate the performance of the proposed algorithms, computational experiments are performed on randomly generated test instances. Finally, computational results show that the proposed approaches obtain near-optimal solutions in a reasonable computational time even for large-sized problems.

1. Introduction

Scheduling is a form of decision making that plays a crucial role in manufacturing and service systems. It began to be taken seriously in manufacturing at the beginning of 20th century, and since then has been received the attention of many researchers for years. In the traditional scheduling problems, most research in the literature was usually conducted under the assumption that the processing time of a job is known in advance and remains constant throughout the whole decision process. However, there are many practical situations where the processing times of jobs are not constant but increasing overtime such that the later a job starts, the longer it takes to process. This phenomenon is known as scheduling with deteriorating jobs to many industrial applications, such as equipment maintenance, steel production, medical emergency, firefighting, and other problems. In these cases, the corresponding scheduling problem was first studied by Browne and Yechiali [1] and J. N. D. Gupta and S. K. Gupta [2]. They assumed that the processing time of all jobs is a function of their starting point ($p_i = a_i + \alpha_i \times s_i$), where a_i is the normal processing time, α_i is a deterioration rate, and s_i is the starting time of job i. From then onwards, more papers considering scheduling jobs with deterioration effects have been published recently [3–19]. Comprehensive reviews and discussions of different models and problems with time-dependent processing times were given by [20, 21].

Most of the current researches in deteriorating jobs scheduling problems assume that the job processing times are linear functions of their starting times. However, in many practical situations, if some jobs fail to be processed prior to a prespecified deteriorating date, then the jobs will require extra time for successful completion. As a result, the original schedule may become inapplicable under the new environment. This motivates the research of the scheduling problem with piecewise-deteriorating jobs. There are often two types of scheduling problems relevant to the piecewise-deteriorating model in the literature. The first one is step-deterioration scheduling problem [22]. Another one is piecewise-linear-deteriorating scheduling problem [23]. In this paper, we study the first kind of scheduling problem. Considering a set of n jobs with deteriorating date d_i for job J_i, the processing time p_i for job J_i is a_i if it is started before the deteriorating date and $a_i + b_i$ if the starting time $s_i > d_i$, where a_i is the normal processing time, b_i is a deteriorating penalty. The objective is to schedule the n jobs on parallel machines to minimize the sum of job completion times. The scheduling problem with step-deterioration jobs is NP-complete even if in single machine environment [24]. As for such a problem over a certain size, it is handicapped to render an optimal schedule within a reasonable run time, depending on exact methods. Hence, some approximate solution techniques are employed to yield an optimal or near-optimal schedule for such problems.

In this paper, a modified weighted combination search algorithm (MWCSA) is firstly proposed to obtain near optimal solution. Subsequently, variable neighborhood search (VNS) algorithm is employed to yield the better schedule for the problem under consideration. VNS is one of the modern metaheuristic techniques that use systematic changes of the neighborhood structure within a local search to solve optimization problem [25, 26]. Owing to incorporating a lot desirable properties for a metaheuristics such as simplicity, efficiency, effectiveness, generality, and so forth, VNS has been widely used to combinatorial optimization problems in recent years. In addition, to improve the performance of VNS, the proposed heuristic MWCSA is utilized to produce initial solution for the VNS. The effectiveness of the proposed approaches is demonstrated by computational results based on a large set of randomly generated test instances.

This paper is organized as follows: Section 2 reviews the relevant literature in this area of scheduling. Section 3 gives a brief description of the problem under consideration and formulates the corresponding mixed integer programming model. A detailed description of the heuristic MWCSA and VNS algorithm are elaborated in Section 4. The performances of the VNS and the heuristic algorithms are shown in Section 5. Conclusions are given in Section 6.

2. Literature Review

We discuss related work with respect to the scheduling problem with piecewise-deteriorating jobs and with regard to VNS methods in scheduling, especially for parallel machines.

Since many different problems are encountered in production and service environments where the processing time of jobs or tasks can be modeled as the piecewise-deteriorating function, many active researchers in the field of scheduling have perceived the promising advantages of considering the piecewise-deteriorating jobs in scheduling problems. Kunnathur and Gupta [27] firstly assumed that job processing times are piecewise-linear function of their starting times with two pieces and addressed five heuristics and two optimizing algorithms based on dynamic programming and branch and bound techniques. However, Sundararaghavan and Kunnathur [22] considered that the processing time can be modeled by step function and gave some solvable cases for minimizing the sum of the weighted completion times in single machine scheduling. In addition, Mosheiov [28] studied the problem of stepwise deteriorating jobs for minimizing makespan, proved that the case of single machine step-deterioration is NP-complete, and suggested heuristic methods in single and multimachine scheduling environments.

Cheng and Ding [24] studied a piecewise model where the job processing time deteriorates as a step function if its starting time exceeds a given threshold, and presented NP-complete proofs for minimizing makespan, flow time, and weighted flow time in single machine scheduling. Jeng and Lin [29] presented a pseudopolynomial time dynamic programming algorithm for a single-machine scheduling problem. Then, they described an efficient branch and bound algorithm based on two dominance rules and a lower bound for deriving optimal solution. Subsequently, a single machine scheduling problem of minimizing the total completion time was solved by branch and bound algorithms [30]. Owing to the complexity of the problem, He et al. [31] proposed a weight combination search algorithm to yield a near-optimal solution. They also developed a branch and bound algorithm incorporated with several properties and a lower bound to obtain the optimal solution. Afterwards, Layegh et al. [32] applied a memetic algorithm based on three dominance properties to minimize the total weighted completion time on a single machine under step-deterioration.

With regards to the piecewise-linear-deterioration model, Kovalyov and Kubiak [33] investigated that the job processing time can be expressed by a piecewise-linear-increasing function with three pieces and presented a fully polynomial approximation scheme for minimizing makespan of single machine scheduling problem. Then Kubiak and van de Velde [23] proved that the problem is NP-hard, and proposed a branch and bound algorithm that solves instances with up to 100 jobs in a reasonable amount of time. Alternatively, the processing job times was expressed in a piecewise-linear-decreasing function of their start times. The single machine scheduling problem was researched by Cheng et al. [34]. Moslehi and Jafari [35] dealt with the same problem proposed by [33] with the minimization of the number of tardy jobs. Owing to the complexity of the studied problem, they proposed a heuristic algorithm with $O(n^2)$ and a branch and bound algorithm.

As far as reviewed, there are some literature focused on single machine scheduling problem with piecewise-deteriorating jobs. However, to the best of our knowledge, parallel machines scheduling problem with piecewise deteriorating jobs has not been considered in the existing literature. And it still lacks some effective approaches to solve the intractable problem, especially metaheuristics. Therefore, we intend to propose a VNS approach for parallel machines scheduling problem with step-deteriorating jobs.

VNS approaches have successfully applied to solve several scheduling problems. Gupta and Smith [36] described a new hybrid of VNS for single machine total tardiness scheduling with sequence-dependence setups. Paula et al. [37] applied a VNS algorithm to solve large instances of parallel machines scheduling problems with sequence-dependent times. Anghinolfi and Paolucci [38] contributed to the design of a hybrid meta-heuristic approach which integrates several features from tabu search (TS), simulated annealing (SA), and VNS for a generalized parallel machine total tardiness scheduling problem. Moreover, Driessel and Mönch [39] proposed several variants of VNS schemes for the same problem considered precedence constraints to minimize the total weighted tardiness of all jobs. Behnamian et al. [40] proposed a hybrid meta-heuristic method which combines some advantages of ant colony optimization, SA and VNS for the minimization of makespan in parallel machine scheduling problem with sequence-dependent times. C.-L. Chen and C.-L. Chen [41] proposed several hybrid metaheuristics for unrelated parallel machine scheduling with sequence dependent setup times by integrating VNS and TS principles. Furthermore, some job shop scheduling problems were solved by VNS [42–45]. A latest systemic survey of state-of-the-art development of VNS can be found in [46].

3. Problem Formulation

The problem under consideration is to schedule n independent jobs with step-deterioration effects, noncommon deadlines and varying processing times on m identical parallel machines. The jobs are available for processing at time zero and the machines are available in the whole process. No job preemption is permitted. For each job J_i, there is a normal processing time a_i and associated with a deteriorating date d_i. If the starting point of its process is less than or equal to its deteriorating date, then it only requires a normal processing time a_i. Otherwise, it requires an extra processing time b_i, which is called the deteriorating penalty. Thus, the actual processing time p_i of job J_i depends on its starting time s_i and deteriorating date d_i, and can be defined as a step-function: $p_i = a_i$ if $s_i \le d_i$; $p_i = a_i + b_i$, otherwise. Without loss of generality, it is assumed that parameters a_i, b_i, and d_i are all integers. Let C_i denote the completion time of job J_i. The goal of the problem is to find a schedule such that the total completion time, or the sum of completion times, of all jobs is minimized. This problem is denoted as $Pm/p_i = a_i$ or $a_i + b_i, d_i/\sum C_i$ by adopting the standard three-field notation.

For convenience, a job is called early if its starting time is before or at its deteriorating date; tardy, otherwise. For schedule S, the objective value is denoted by $Z(S)$. In addition, let M be a sufficiently large positive number. Before formulating the proposed mixed integer programming model, the following variables have to be defined.

x_{ijk}: Binary, set to 1 if job i is immediately followed by job j in sequence on machine k; 0, otherwise $(1 \le i,\ j \le n,\ 1 \le k \le m)$

y_{ik}: Binary, set to 1 if job i is assigned to machine k; 0, otherwise $(1 \le i \le n,\ 1 \le k \le m)$.

Based on the definitions and notation described above, the considered problem can now be formulated as 0-1 mixed integer programming model, as shown below.

$$\text{Minimize } Z(S) = \sum_{i=1}^{n} C_i. \tag{3.1}$$

Subject to:

$$p_i = \begin{cases} a_i, & s_i \le d_i \\ a_i + b_i, & \text{otherwise} \end{cases} \quad \forall i = 1, \ldots, n, \tag{3.2}$$

$$\sum_{i=1}^{n} x_{0ik} = 1 \quad \forall k = 1, \ldots, m, \tag{3.3}$$

$$\sum_{i=1}^{n} x_{i(n+1)k} = 1 \quad \forall k = 1, \ldots, m, \tag{3.4}$$

$$\sum_{i=0, i \ne j}^{n} x_{ijk} = y_{jk} \quad \forall j = 1, \ldots, n, \ \forall k \ = \ 1, \ldots, m, \tag{3.5}$$

$$\sum_{j=1, i \ne j}^{n+1} x_{ijk} = y_{ik} \quad \forall i = 1, \ldots, n, \ \forall k = 1, \ldots, m, \tag{3.6}$$

$$C_i \ge p_i + M(x_{0ik} - 1) \quad \forall i = 1, \ldots, n, \ \forall k = 1, \ldots, m, \tag{3.7}$$

$$C_j \ge C_i + p_j + M\left(x_{ijk} - 1\right) \quad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, n, \ \forall k = 1, \ldots, m, \tag{3.8}$$

$$s_i \ge M(x_{0ik} - 1) \quad \forall i = 1, \ldots, n, \ \forall k = 1, \ldots, m, \tag{3.9}$$

$$s_j \ge C_i + M\left(x_{ijk} - 1\right) \quad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, n, \ \forall k = 1, \ldots, m, \tag{3.10}$$

$$\sum_{k=1}^{m} y_{ik} = 1, \quad \forall i = \ 1, \ldots, n, \tag{3.11}$$

$$x_{ijk}, y_{ik} \in \{0, 1\}, \quad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, n, \ \forall k \ = \ 1, \ldots, m. \tag{3.12}$$

In the above mathematical model, the objective (3.1) minimizes the total completion time. Constraints (3.2) depict the processing time of the step-deteriorating jobs. Constraints (3.3) and (3.4) ensure that only one job can be processed at the first and the last position on each machine. Constraints (3.5) and (3.6) guarantee that each job is scheduled only once and processed by at most one machine. Constraints (3.7) define the completion time of the first job assigned to a machine. Constraints (3.8) represent that the completion time of a job in sequence on each machine will be at least equal to the sum of the completion time of the preceding job and the processing time of the present job, if the job is immediately schedule after the previous job. Constraints (3.9) define the starting time of the first job assigned to a machine. Constraints (3.10) state that the starting time of a job in sequence on each machine is greater than or equal to the completion time of the preceding job. Constraints (3.11) confirm that each job is only processed to exactly one machine. Constraints (3.12) specify that the decision variable x and y is binary over all domains.

This described problem is NP-complete because the single machine problem $1/p_i = a_i$ or $a_i + b_i$, $d_i/\sum C_i$ has proven to be NP-complete by [24]. There exists no polynomial time algorithm for the exact solution of the considered problem. Therefore, we have to look for some efficient heuristic or meta-heuristic algorithms to yield near optimal solutions of large-sized problems in a reasonable computational time.

4. Heuristic Solution Approaches

In this section, we start with presenting one property. Then we modify a heuristic proposed in a literature. Subsequently, some detailed descriptions of VNS meta-heuristic are given.

Since the identical parallel machines scheduling problem $Pm//\sum C_i$ is solvable in $O(N\log N)$ time using the generalized shortest processing time (GSPT) rule [47]. Therefore, we can also use the same method to determine the sequence of early jobs and tardy jobs assigned to a given machine for minimizing the total completion time. Based on the GSPT, we obtain the following property directly.

Property 1. On each machine of an optimal solution for the considered problem, early jobs and tardy jobs are sequenced in the nondecreasing order of a_i and $a_i + b_i$, respectively.

Proof. The proof is straightforward from the SPT rule on a_i and $a_i + b_i$, respectively. □

4.1. Modified Weight Combination Search Approach

He et al. [31] proposed weight combination search approach (WCSA) to solve single machine total completion time scheduling problem with step-deteriorating jobs. The performance of the WCSA is relatively excellent compared to the proposed branch and bound algorithm within 24 jobs. Based on the good performance of the algorithm, it is modified to adapt the parallel machine model by incorporating Property 1. But the ranges of the weights in the literature are not suitable for parallel machine scheduling. For specifying the suitable ranges of the weights we carried out the preliminary tests of 30 10-job instances and found that the linear combination of a_i, d_i, and b_i will acquire better results if $\omega_1 \in [0.4, 0.75]$, $\omega_2 \in [0.2, 0.5]$ and $\omega_3 = 1 - \omega_1 - \omega_2$. The procedure of the modified WCSA (MWCSA) is designed as show in Algorithm 1.

From Algorithm 1, all jobs are sorted in non-decreasing order of their normal processing times a_i. The first m jobs are assigned to m machines, respectively. Then the last $n-m$ jobs are arranged in a non-decreasing order of the weight combination of processing time, deteriorating date and deterioration penalty. Of course, if the completion time of the last scheduled job is greater than the maximal deteriorating date of the unscheduled jobs, then the remaining jobs are indexed in the order of sum of their normal processing times and deterioration penalties.

4.2. Variable Neighborhood Search Metaheuristic

VNS, a new local search technique, attempts to escape from local optimum by exploring more than one type of neighborhood search structure (NSS) during the course of the algorithm. In practice, VNS is very similar to iterated local search (ILS). Owing to iterating over one constant type of neighborhood structure, ILS is easy to stick in local optima: the single move required to improve the solution cannot escape from the found local optima, and even lead to a deterioration of the solution quality. The disadvantage is also existed in some other metaheuristics, such as simulated annealing, and tabu search. However, VNS makes full use of some NSSs until some stopping criterion is met. By exploring various NSSs, we expect VNS to enjoy a systematic diversification mechanism. Besides this diversification mechanism, the other reasons to high acceptability and popularity of VNS among researchers are due

```
Procedure: the modified WCSA for the parallel machine scheduling problem
Inputs: $n, m, a_i, d_i, b_i$ for $i = 1,\ldots, n$
Output: the near optimal scheme $S\_opt$ and the associated total completion time $TC$
Begin
  let $N_0 = \{J_1, J_2, \ldots, J_n\}$ be the sequence that sorts all the jobs by ascending order of their
  normal time $a_i$,
  set $TC$ = infinity, and $S\_opt_k = \Phi$ for $k = 1, \ldots, m$          % initialize the $TC$ and $S\_opt$
  set var = $\max\{2, \lceil n/m \rceil\}$                                   % initialize the ranges of $l_1$ and $l_2$
  for ($l_1 = 1$; $l_1 \le$ var; $l_1$++)
      $\omega_1 = 0.4 + (0.75 - 0.4) \times ((l_1 - 1)/(\text{var} - 1))$
      for ($l_2 = 1$; $l_2 \le$ var; $l_2$++)
        $\omega_2 = 0.2 + (0.5 - 0.2) \times ((l_2 - 1)/(\text{var} - 1))$
        $\omega_3 = 1 - \omega_1 - \omega_2$
        set $N = N_0$
        set $S = \Phi$ for $k = 1, \ldots, m$                               % initialize the scheme $S$
        set $C^{\text{mac}}(k) = 0$, for $k = 1, \ldots, m$                  % initialize the completion
                                                                            time of all machines
        set $C^{\text{job}}(i) = 0$, for $i = 1, \ldots, n$                  % initialize the completion
                                                                            time of all jobs
        for ($i = 1$; $i \le m$; $i$++)
          select the machine $h$ that has the least completion time in all machines
          select the job $J_h$ that has the least normal time from $N$
          $S_h = S_h \cup \{J_h\}$
          $C^{\text{mac}}(h) = C^{\text{mac}}(h) + a(J_h)$
          $C^{\text{job}}(J_h) = C^{\text{mac}}(h)$
          delete job $J_h$ from $N$
      end for
      for ($j = m + 1$; $j \le n$;  $j$++)
        select the machine $f$ that has the least completion time in all machines
        if($C^{\text{mac}}(f) > \max\{d(J_r), (J_r \in N)\}$)               % tardy jobs are sequenced in the
                                                                            nondecreasing order of $a_i + b_i$
          select the job $J_f$ with the smallest $a(J_f) + b(J_f)$ from $N$
          $S_f = S_f \cup \{J_f\}$
          $C^{\text{mac}}(f) = C^{\text{mac}}(f) + a(J_f) + b(J_f)$
          $C^{\text{job}}(J_f) = C^{\text{mac}}(f)$
          delete job $J_f$ from $N$
        else                                                                % arrange the jobs in a ascending
                                                                            order of the weight of combination
          set $N' = \{d(J_r) \ge C^{\text{mac}}(f), (J_r \in N)\}$
          select job $J_f$ with the smallest $\omega_1 \times a(J_f) + \omega_2 \times d(J_f) - \omega_3 \times b(J_f)$ from $N'$
          $S_f = S_f \cup \{J_f\}$
          $C^{\text{mac}}(f) = C^{\text{mac}}(f) + a(J_f)$
          $C^{\text{job}}(J_f) = C^{\text{mac}}(f)$
          delete job $J_f$ from $N$
        end if
      end for
      $TC\_\text{temp} = \text{sum}\{C^{\text{job}}(i) = 0, \text{ for } i = 1, \ldots, n\}$
      if $TC\_\text{temp} < TC$                                             % update the scheme $S\_opt$ and the
                                                                            assisted completion time $TC$
        $TC = TC\_\text{temp}$
        $S\_opt = S$
      end if
    end for
  end for
End
```

Algorithm 1: Procedure of MWCSA.

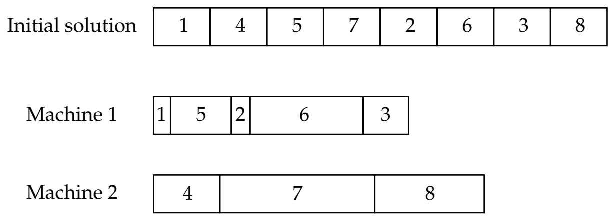


Figure 1: Sequence in one solution and its schedule.

to conceptual simplicity to understand and implement and the high flexibility and brilliant adaptability to different problems. Consequently, we intend to use it to solve our problem.

Since the VNS was proposed in 1997, some variants of VNS are developed by Hansen and Mladenović [25]. Variable neighborhood descent (VND), as one of these variants of VNS, performs change of neighborhoods in a deterministic way and adopts the first improvement rather than the best improvement. In the subsequent subsection, we are going to employ a VNS with five powerful NSSs based on insertion, swap, and inversion under the framework of VND.

4.2.1. Encoding, Decoding, and Initialization Schemes

Encoding scheme is a procedure to make a solution recognizable for an algorithm. Permutation list is a common used encoding scheme for combinatorial optimization problems. Since the parallel machines scheduling problem is a combination of machine assignment and operation sequencing decisions, we construct five neighborhood structures that expediently use insertion, swap, and inversion operations based on permutation list. A good initial solution can reduce considerably the computational time. According to observing the processing sequence of jobs, we find that a job with the smallest normal processing time should be processed at first, and a job with the largest penalty should be processed at last. Therefore, we can obtain the initial solution by arranging jobs in the non-decreasing order of the ratio of a_i/b_i. The sample sequencing method is called smallest rate first (SRF). A permutation with all jobs sequence can express the order in which the jobs are processed. To decode a permutation list into a schedule, we use LIST method. For one solution sequence, whenever a machine is freed the job that is chosen from the sequence in succession is put on the machine.

The procedures of encoding and decoding a candidate solution are illustrated according to an example. Consider a problem with 8 jobs and 2 machines. The parameters of all jobs are given in Table 1. The initial solution is obtained by using SRF, as shown in Figure 1. The job that is selected from the solution sequence one by one is processed by the machine with the smallest completion time. The schedule is also depicted in Figure 1. We calculated that the total completion processing time is 1113.

4.2.2. Neighborhood Search Structures

The main purpose of applying a neighborhood structure is to produce a neighboring solution from the current solution via making some changes in it. A variety of neighborhood structures have been applied to scheduling problems. Most of these neighborhood structures are based

Table 1: The parameters of all jobs in the example under consideration.

Parameter	Job number							
	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
a_i	10	13	28	55	63	81	90	95
d_i	14	48	21	52	55	60	36	7
b_i	14	3	5	41	35	16	47	2

```
Procedure: $NSS_1$
improvement = yes
while (improvement = yes) do
    improvement = no
    $i = 1$
  while $i \le n$ do        ($n$ is the number of jobs)
    select randomly one job $p$ without repletion from the current solution $x$
    $x'$ = swap job $i$ and job $p$ on their positions.
    if $Z(x') < Z(x)$ do
        $x = x'$
        $i = n$
        improvement = yes
    end if
    $i = i + 1$
  end while
end while
```

Algorithm 2: Procedure of neighborhood structure NSS$_1$.

on insertion or swap operations. In addition, the inversion operation has been reported to be significantly surpassing all the modified forms of crossover of genetic algorithm especially tailored to deal with combinational problems [48]. Therefore, we define five types of neighborhood search structures based on insertion, swap, and inversion.

To generate a neighboring solution, NSS$_1$ makes some slight changes in the candidate solution by swapping the positions of all jobs in the sequence with one selected job at random and without repetition. All jobs are selected one after another without repetition in a random order. If we observe the first improvement, the associated sequence is accepted and the procedure restarts. According to the NSS$_1$, some available neighboring searches are found and used to improve the candidate solution. The whole procedure repeats so long as no improvement is obtained through swap all jobs with another randomly selected job. The procedure of NSS$_1$ is described in Algorithm 2 in detail. After implementing all swap operations, we believe that there is little hope for further improvement just by swapping two jobs on their positions. Hence, it necessitates consider another neighborhood structure to escape from this local optimum of the NSS$_1$. Subsequently, we need to introduce another NSS to search potential improvement based on insertion moves.

In our NSS$_2$, the procedure is similar to the NSS$_1$. The only difference between them is their move pattern that NSS$_2$ changes the position of one job based on insertion neighborhood. A job is removed from the sequence at random and without repetition, and then relocated another random selected position. The other procedure is the same as the

```
Procedure: $NSS_3$
$i = 1$
while $i \leq n$ do
    $j = j + 1$
    while $j \leq n$ do
        select randomly two jobs $p$ and $q$ from the current solution $x$
        $x'$= swap the two selected jobs on their positions
        if $Z(x') < Z(x)$ do
            $x = x'$
            $j = n$
            $i = n$
        end if
        $j = j + 1$
    end while
    $i = i + 1$
end while
```

Algorithm 3: Procedure of neighborhood structure NSS$_3$.

NSS$_1$. To avoid duplicated description, it is unnecessary to go into these details here. After relocating all jobs, we feel that the algorithm cannot rely solely on single move to improve the quality of the current solution. Hence, we need to introduce other neighborhood search structure to generate more complex neighbors than the above two structures.

In the NSS$_3$, the number of randomly selected jobs is 2. The manner of choosing these 2 jobs is all the combinations of two-out-of-n jobs. Swap the selected jobs on their positions and observe if the solution is improved. Once observing the first improvement, the associated solution is accepted and the procedure restarts. If not, this search strategy repeats for the subsequent combinations. Algorithm 3 illustrated the whole procedure of the NSS$_3$. By the same way of switching from NSS$_1$ to NSS$_2$, the double insertions are employed in the fourth neighborhood structure NSS$_4$. In the NSS$_4$, the number of removed jobs is set equal to 2. To relocate them, the two jobs are reinserted into two new randomly selected positions. The other procedure is the same to the NSS$_3$.

To improve further the search performance of the proposed VNS, an inversion operator is embedded into the NSS$_5$ when the foregoing neighborhood search methods terminate. In the NSS$_5$, it randomly selects two positions, known as the points of inversion, and inverts the sequence between these positions. The inversion procedure is repeated φ times for producing a new job sequence. Owing to the impact of the parameter φ on neighborhood search, it must be tuned. The inversion operation is unidirectional, that is, only the inverted sequence is improved, the corresponding solution is accepted to replace the incumbent one. The procedure of is described in Algorithm 4.

The general outline of the proposed VNS is shown in Algorithm 5. According to using the above neighborhood search structures, the proposed VNS can be quick to yield the near optimal solution. If the neighborhood search structure continues to increase the number of selected jobs, the quality of the algorithm will be not improved according to our primary experiments.

```
Procedure: $NSS_5$
for $i = 1$ to $\varphi$ do
    select two jobs $p$ and $q$ at random from the current solution $x$
    $x'$ = invert partial jobs between job $p$ and job $q$
    if $Z(x') < Z(x)$ do
        $x = x'$
    end if
end for
```

Algorithm 4: Procedure of neighborhood structure NSS$_5$.

```
Procedure: the proposed VNS algorithm
Initialization: Define the set of neighborhood structures $NSS_k$, for $k = 1, \ldots, k_{\max}$;
                find a initial solution $x$ by SRF heuristic;
                choose a stopping criterion.
$k = 1$
while the stopping criterion is not met do
    perform the neighborhood search structure $NSS_k$
    if $x$ is improved do
        continue the search with $NSS_k$
    else
        $k = k \bmod k_{\max} + 1$
    end if
end while
```

Algorithm 5: Procedure of the proposed VNS algorithm.

5. Computational Analysis

To test the performance of our approaches, computational experiments were carried out for the $Pm/p_i = a_i$ or $a_i + b_i$, $d_i/\sum C_i$ problem. More specifically, we also carried out comprehensive computational and statistical tests to assess the performance of MWCSA, and VNS. Firstly, we gave a detailed description of the different instance sets that we have employed. After reporting the results, we carried out comprehensive statistical analyses in order to soundly test the significance of the reported results.

Because there is not a widely available set of benchmark instances for the problem under study, data for the test instances are generated randomly. The normal processing times (a_i) are randomly generated from an integer uniform distribution on $U[1, 100]$. The deterioration penalties (b_i) are randomly picked from an integer uniform distribution on $U[1, 100 \times \beta]$, where $\beta = 0.5$. Let $D_f = \sum_{i=1}^{f \times n} a_i/m$, $0 \leq f \leq 1$. The deteriorating dates d_i are randomly selected from three intervals $U(0, D_{0.5}]$, $U[D_{0.5}, D]$, and $U(0, D]$. In this paper, we have divided the benchmark instances into 2 different types according to the number of jobs. For small-sized instances, we test all combinations of $n = \{6, 8, 10\}$ and $m = \{2, 3\}$. For large-sized instances, we use the combinations of $n = \{20, 40, 60, 80, 100\}$ and $m = \{2, 4, 6, 8, 10\}$ to evaluate the proposed algorithms. Since the deteriorating date may be generated from three different intervals, we totally have $6 \times 3 + 25 \times 3 = 93$ instances.

In order to validate the performance of the proposed approaches, attempts are made to solve the MIP model presented in Section 3 using the software IBM-ILOG CPLEX 12.3 solvers. The software is run on a PC with Intel Pentium Dual-Core 2.60 GHz CPU and 2 GB RAM. Because the problem under consideration is NP-complete, it is impossible to obtain optimal solutions by using some polynomial time algorithms. Therefore, only small-sized instances can be solved to optimality using CPLEX. For the small-sized instances, we use relative percentage deviation (RPD) as a performance measure to compare those results of the CPLEX software, the MWCSA and the VNS. RPD is obtained by given formula below:

$$\text{RPD} = \frac{Z(\text{Alg}) - Z(\text{OPL})}{Z(\text{OPL})} \times 100, \tag{5.1}$$

where $Z(\text{Alg})$ is the total completion time of the solution obtained by a given algorithm and instance and $Z(\text{OPL})$ is the total completion time of the optimal schedule given by CPLEX. The SRF (small rate first) of Section 4 is also brought into the large-sized instances for comparison. Our purpose to utilize SRF is to use its result as upper bound for a given instance to evaluate the VNS. In addition, the MWCSA is used to obtain the initial solution of the VNS for improving the quality of solutions. The hybrid algorithm, denoted by VNS + MWCSA, is employed for the large-sized instances.

All proposed algorithms in the previous sections were coded in MATLAB 7.11 and implement on the same PC. According to the preliminary tests, the stopping criterion used when testing all instances with the proposed algorithms is set to the maximum iterations time fixed to 200. Thus, there is only one parameter (φ) is tuned in our proposed VNS. The considered candidate levels are 10, 30, 50, and 70. A set of 15 large-sized instances are randomly generated when the number of jobs is fixed to 100. All the 15 instances are solved by these VNSs with different parameter candidate levels. The results were analyzed by the one-way ANOVA test. The means plot and Fisher LSD intervals for the 4 levels of parameter φ are shown in Figure 2. As we can see, φ of 50 and 70 provide the better results among all levels. However, it needs more computational time when φ is 70. Therefore, the most reasonable value of parameter φ is 50.

CPLEX, SRF, and MWCSA are deterministic and only one run is necessary. The VNS and VNS + MWCSA are stochastic and we need to run some replicates in order to better assess the results. We run each algorithm five times. The listed objective value and computational time are the means of five reported results.

Table 2 provides a comparison of the solutions to the mixed integer programming model generated by the CPLEX and those solutions provided by execution of the proposed MWCSA and VNS algorithms. All small-sized instances were solved to optimality by CPLEX as shown in Table 1. The computational time of the CPLEX increases rapidly as the instance become larger. When the number of jobs is equal to 10, solutions to MIP models require far longer than 1 hour. From Table 1, the longest computational time of CPLEX is 144761.59 seconds (40.21 hours). It is impractical in scheduling for such a long time. In addition, we found that CPLEX is not greatly affected by the distribution of the deteriorating dates almost. While the deteriorating dates are randomly picked from the interval $U[D_{0.5}, D]$, all corresponding instances are relatively easy. This is because the number of deteriorating jobs is less. It is worthwhile to note that the MWCSA behaves well for all small-sized instances and gives an average result of 0.62% deviation from the optimal solution. Specially, eleven of these instances can be solved optimally by the proposed heuristic. Its execution time is far

Table 2: Comparison of CPLEX results with the proposed algorithms for small-sized instances.

		CPLEX		MWCSA		VNS		
n	m	Objective value (gap)	Time (sec)	Objective value	RPD	Objective value	RPD	Time (sec)
6	2	582	1.48	**582**	0.00%	**582**	0.00%	0.36
	3	360	2.47	370	2.78%	**360**	0.00%	0.30
8	2	859	103.55	**859**	0.00%	**859**	0.00%	0.52
	3	513	122.20	**513**	0.00%	**513**	0.00%	0.46
10	2	1284	24632.09	1308	1.87%	**1284**	0.00%	0.72
	3	1190	144761.59	1216	1.59%	**1190**	0.00%	0.72
*the deteriorating date with the interval $U(0, D_{0.5}]$								
6	2	430	1.50	**430**	0.00%	**430**	0.00%	0.30
	3	392	2.05	**392**	0.00%	**392**	0.00%	0.30
8	2	835	80.70	**835**	0.00%	**835**	0.00%	0.46
	3	590	233.25	**590**	0.00%	**590**	0.00%	0.47
10	2	1186	17676.28	**1186**	0.00%	**1186**	0.00%	0.71
	3	794	62828.11	799	0.63%	**794**	0.00%	0.72
*the deteriorating date with the interval $U[D_{0.5}, D]$								
6	2	446	1.63	**446**	0.00%	**446**	0.00%	0.30
	3	335	1.92	**335**	0.00%	**335**	0.00%	0.31
8	2	934	102.61	939	0.54%	**934**	0.00%	0.47
	3	486	205.08	491	1.03%	**486**	0.00%	0.47
10	2	1161	22216.36	1193	2.76%	**1161**	0.00%	0.72
	3	638	51367.00	**638**	0.00%	**638**	0.00%	0.70
*the deteriorating date with the interval $U(0, D]$								
Average					0.62%		0.00%	

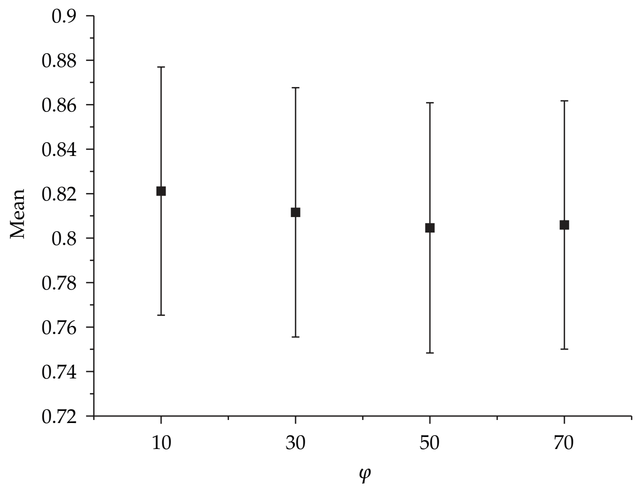


Figure 2: Means plot and Fisher LSD intervals for the different levels of parameter φ (the significance level α is 0.05).

Table 3: Test results of the algorithms on the instances with the interval $U(0, D_{0.5}]$.

Problem number	n	m	SRF Objective value (A)	MWCSA Objective value (B)	$(B)/(A)$	VNS Objective value (C)	$(C)/(A)$	Time (sec)	VNS + MWCSA Objective value (D)	$(D)/(A)$	Time (sec)
1	20	2	5859	5062	86.40%	5001	85.36%	3.71	5001	85.36%	3.90
2		4	3274	2589	79.08%	2587	79.02%	3.58	2583	78.89%	3.77
3		6	1792	1676	93.53%	1655	92.35%	3.65	1655	92.35%	3.71
4		8	1299	1247	96.00%	1199	92.30%	3.68	1199	92.30%	3.73
5		10	1324	1309	98.87%	1269	95.85%	3.71	1267	95.69%	3.76
6	40	2	17056	14013	82.16%	13361	78.34%	23.00	13307	78.02%	23.73
7		4	11530	9614	83.38%	9513	82.51%	23.57	9508	82.46%	24.38
8		6	7833	6890	87.96%	6800	86.81%	23.05	6797	86.77%	23.56
9		8	4684	4299	91.78%	4201	89.69%	23.60	4198	89.62%	23.67
10		10	3858	3465	89.81%	3434	89.01%	24.15	3428	88.85%	24.71
11	60	2	47917	41030	85.63%	39204	81.82%	73.04	39149	81.70%	71.14
12		4	28547	22651	79.35%	22234	77.89%	72.34	22184	77.71%	67.62
13		6	17378	13616	78.35%	13504	77.71%	70.08	13474	77.53%	70.38
14		8	12577	10288	81.80%	10122	80.48%	69.37	10121	80.47%	72.68
15		10	10346	8695	84.04%	8573	82.86%	71.59	8548	82.62%	68.72
16	80	2	73047	66321	90.79%	58736	80.41%	157.50	58392	79.94%	181.86
17		4	41378	33779	81.64%	32734	79.11%	160.57	32687	79.00%	132.71
18		6	27955	22144	79.21%	21679	77.55%	152.83	21671	77.52%	142.25
19		8	20569	16200	78.76%	16055	78.05%	157.20	16013	77.85%	136.14
20		10	15479	12720	82.18%	12569	81.20%	160.32	12516	80.86%	152.90
21	100	2	126101	108828	86.30%	98779	78.33%	288.47	98026	77.74%	290.95
22		4	58800	46684	79.39%	45214	76.89%	295.09	44858	76.29%	277.00
23		6	46629	35717	76.60%	34678	74.37%	307.01	34466	73.92%	240.82
24		8	35233	27602	78.34%	27275	77.41%	286.20	27143	77.04%	263.09
25		10	29084	22906	78.76%	22753	78.23%	299.92	22629	77.81%	303.41
Average					84.40%		82.14%			81.93%	

less than 1 second so that its log is not necessary. It can also be noticed that the solutions from the VNS algorithm are optimal for these small-sized instances. Meanwhile, the computational time of these instances given by VNS are not more than 1 second. This means that the VNS algorithm is effective in solving the scheduling problem under study and has significantly better optimization performance than the others.

Tables 3, 4, and 5 show the results of large-sized instances where the deteriorating dates are generated from three different intervals. In order to compare the performance of the proposed algorithms, the ratios of the results of a given algorithm to the values obtained by SRF were calculated out. For the instances with the deteriorating dates generated from interval $U(0, D_{0.5}]$, VNS behaves slightly better than MWCSA. The difference in average rate between two algorithms is only 2.26%. For other two types of instances with intervals $U[D_{0.5}, D]$ and $U(0, D]$, the VNS and VNS + MWCSA strikingly outperformed the MWCSA heuristic because of its larger search space. As a rule of thumb, a given algorithm takes possession of the better initial solution and should yield better result. However, the hybrid

Table 4: Test results of the algorithms on the instances with the interval $U[D_{0.5}, D]$.

Problem number	n	m	SRF	MWCSA		VNS			VNS + MWCSA		
			Objective value (A)	Objective value (B)	$(B)/(A)$	Objective value (C)	$(C)/(A)$	Time (sec)	Objective value (D)	$(D)/(A)$	Time (sec)
1	20	2	5065	4936	97.45%	4576	90.35%	3.68	4576	90.35%	3.81
2		4	2405	2186	90.89%	2145	89.19%	3.66	2145	89.19%	3.70
3		6	1420	1298	91.41%	1296	91.27%	3.67	1296	91.27%	3.76
4		8	1172	1146	97.78%	1146	97.78%	3.73	1146	97.78%	3.75
5		10	1567	1528	97.51%	1528	97.51%	3.74	1528	97.51%	3.73
6	40	2	17081	18299	107.13%	15326	89.73%	23.57	15316	89.67%	26.33
7		4	10157	9559	94.11%	9133	89.92%	24.56	9133	89.92%	24.69
8		6	5847	5040	86.20%	4944	84.56%	24.00	4943	84.54%	23.70
9		8	3708	3376	91.05%	3334	89.91%	23.87	3334	89.91%	24.17
10		10	4141	3772	91.09%	3733	90.15%	24.67	3733	90.15%	24.67
11	60	2	37202	40606	109.15%	32409	87.12%	74.86	32406	87.11%	85.56
12		4	18904	18275	96.67%	16831	89.03%	73.84	16830	89.03%	76.31
13		6	14321	13618	95.09%	13122	91.63%	72.86	13113	91.56%	74.14
14		8	9488	8980	94.65%	8792	92.66%	77.12	8792	92.66%	75.39
15		10	10032	9203	91.74%	8991	89.62%	77.29	8988	89.59%	78.05
16	80	2	61930	68725	110.97%	54652	88.25%	163.31	54651	88.25%	205.40
17		4	27137	28577	105.31%	24790	91.35%	167.83	24776	91.30%	169.78
18		6	29003	26914	92.80%	25022	86.27%	175.63	25021	86.27%	177.12
19		8	17291	15715	90.89%	15037	86.96%	162.31	15032	86.94%	164.65
20		10	12900	12215	94.69%	11860	91.94%	168.39	11855	91.90%	169.90
21	100	2	110757	128605	116.11%	98384	88.83%	286.03	98358	88.81%	386.87
22		4	50773	49696	97.88%	44153	86.96%	310.59	44152	86.96%	315.32
23		6	30745	31697	103.10%	27834	90.53%	319.17	27833	90.53%	319.76
24		8	26562	25254	95.08%	23576	88.76%	320.56	23573	88.75%	288.13
25		10	24330	22469	92.35%	21508	88.40%	318.57	21506	88.39%	327.75
Average					97.24%		89.95%			89.93%	

algorithm, with longer run times, does not perform noticeably better than the VNS. It is chiefly because the VNS carries out enough neighborhood searches before the stopping criterion meets. The rates of three different types of instances were plotted into smooth curves shown in Figures 3, 4, and 5. It can be seen from those figures that the performance of MWCSA heuristic is not particularly robust as regards the distribution of the deteriorating date. Alternatively, VNS and VNS + MWCSA are statistically better than MWCSA for three types of instances.

6. Conclusions

This paper considers the identical parallel machines scheduling problem with step-deteriorating jobs. The processing time of each job is a step function of its starting time and

Table 5: Test results of the algorithms on the instances with the interval $U(0, D]$.

Problem number	n	m	SRF Objective value (A)	MWCSA Objective value (B)	$(B)/(A)$	VNS Objective value (C)	$(C)/(A)$	Time (sec)	VNS + MWCSA Objective value (D)	$(D)/(A)$	Time (sec)
1	20	2	4670	4681	100.24%	4103	87.86%	3.65	4100	87.79%	3.75
2		4	3762	3034	80.65%	2981	79.24%	3.70	2980	79.21%	3.69
3		6	2179	2077	95.32%	2048	93.99%	3.69	2048	93.99%	3.77
4		8	1166	1126	96.57%	1070	91.77%	3.66	1067	91.51%	3.62
5		10	1056	1037	98.20%	1037	98.20%	3.73	1037	98.20%	3.74
6	40	2	19603	19675	100.37%	17212	87.80%	23.44	17174	87.61%	25.33
7		4	9990	9373	93.82%	8871	88.80%	23.67	8866	88.75%	23.89
8		6	7720	6942	89.92%	6786	87.90%	24.45	6786	87.90%	24.49
9		8	6087	5481	90.04%	5442	89.40%	23.81	5433	89.26%	24.45
10		10	5067	4356	85.97%	4321	85.28%	24.70	4321	85.28%	24.33
11	60	2	40513	40049	98.85%	33051	81.58%	69.64	33014	81.49%	80.71
12		4	19380	17741	91.54%	15364	79.28%	69.82	15358	79.25%	75.96
13		6	15350	12530	81.63%	11737	76.46%	75.05	11722	76.36%	72.58
14		8	11725	10139	86.47%	9889	84.34%	74.55	9874	84.21%	73.80
15		10	9949	8786	88.31%	8699	87.44%	73.75	8685	87.30%	72.12
16	80	2	74532	75705	101.57%	61269	82.20%	152.41	61263	82.20%	199.07
17		4	41890	37588	89.73%	32807	78.32%	171.50	32798	78.30%	170.17
18		6	27990	23587	84.27%	21803	77.90%	155.46	21754	77.72%	160.16
19		8	19278	16845	87.38%	16162	83.84%	161.03	16117	83.60%	158.46
20		10	16916	14285	84.45%	13804	81.60%	169.19	13802	81.59%	170.30
21	100	2	108067	107291	99.28%	81743	75.64%	281.41	81426	75.35%	373.51
22		4	59031	56349	95.46%	48136	81.54%	294.00	48135	81.54%	325.10
23		6	43204	38473	89.05%	34873	80.72%	297.22	34774	80.49%	323.49
24		8	30485	26263	86.15%	23977	78.65%	295.55	23957	78.59%	308.85
25		10	19559	17986	91.96%	16919	86.50%	305.73	16853	86.16%	304.20
Average					91.49%		84.25%			84.15%	

a deteriorating date that is individual to all jobs. The problem is to determine the allocation of jobs to machines as well as the sequence of the jobs assigned to each machine for the criteria of minimizing the total completion time. A mathematical model for this problem has been formulated. Since the problem under study is NP-complete, it is impossible to solve large-sized instances to optimality. To solve the tackled problem, a heuristic MWCSA and a VNS are proposed to obtain the near optimal solutions. In order to further improve the quality of solution, the heuristic MWCSA has been hybridized with the VNS algorithm and implemented to provide a good initial solution. Numerical experiments are conducted on small- and large-sized instances. Computational results show that MWCSA produces some good solutions compared to CPLEX, but the performance is greatly affected by the distribution of the deteriorating date. In contrast, VNS and VNS + WMCSA are robust as regards three types of instances. Therefore, fairly good solutions can be obtained by the proposed methods within reasonable amount of time.

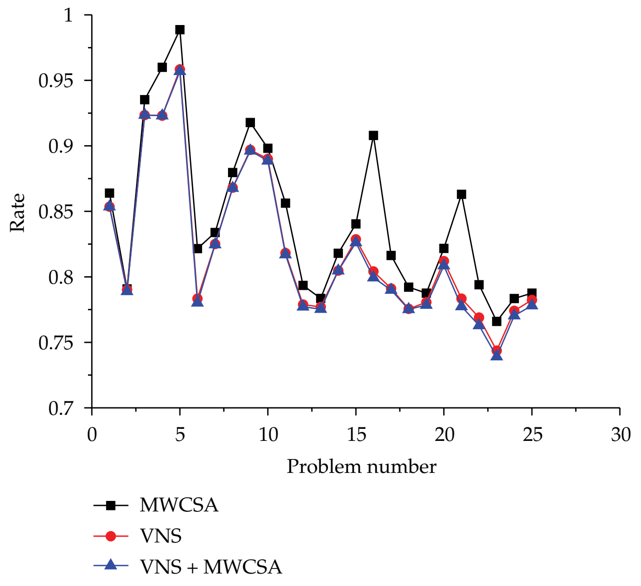


Figure 3: Comparison of rates between the three algorithms for the case with the interval $U(0, D_{0.5}]$.

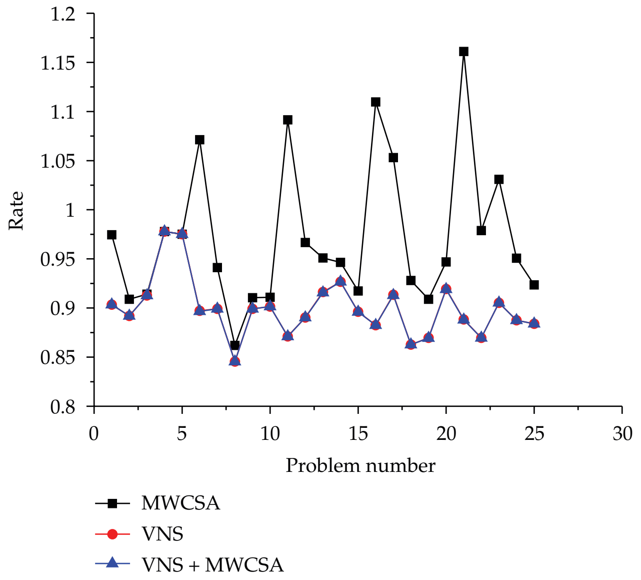


Figure 4: Comparison of rates between the three algorithms for the case with the interval $U[D_{0.5}, D]$.

For further study, it is worth considering the setup times between jobs for the problem of scheduling jobs with piecewise-deterioration on multimachine. In addition, other efficient constructive heuristics and neighborhood properties are worthwhile to investigate for improving the quality of solutions.

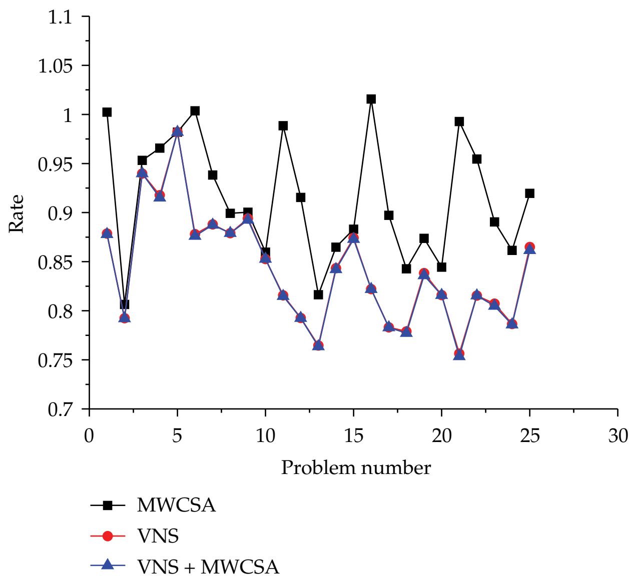


Figure 5: Comparison of rates between the three algorithms for the case with the interval $U(0, D]$.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant no. 51175442, the Youth Foundation for Humanities and Social Sciences of Ministry of Education of China under Grant no. 12YJCZH296, the Ph.D. Programs Foundation of Ministry of Education of China under Grant no. 200806131014, and the Fundamental Research Funds for the Central Universities under Grant nos. SWJTU09CX022, 2010ZT03.

References

[1] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.

[2] J. N. D. Gupta and S. K. Gupta, "Single facility scheduling with nonlinear processing times," *Computers and Industrial Engineering*, vol. 14, no. 4, pp. 387–393, 1988.

[3] A. Bachman and A. Janiak, "Minimizing maximum lateness under linear deterioration," *European Journal of Operational Research*, vol. 126, no. 3, pp. 557–566, 2000.

[4] A. Bachman, A. Janiak, and M. Y. Kovalyov, "Minimizing the total weighted completion time of deteriorating jobs," *Information Processing Letters*, vol. 81, no. 2, pp. 81–84, 2002.

[5] C.-C. Wu, W.-C. Lee, and Y.-R. Shiau, "Minimizing the total weighted completion time on a single machine under linear deterioration," *International Journal of Advanced Manufacturing Technology*, vol. 33, no. 11-12, pp. 1237–1243, 2007.

[6] J.-B. Wang, L. Lin, and F. Shan, "Single-machine group scheduling problems with deteriorating jobs," *International Journal of Advanced Manufacturing Technology*, vol. 39, no. 7-8, pp. 808–812, 2008.

[7] J.-B. Wang, C. T. Ng, and T. C. E. Cheng, "Single-machine scheduling with deteriorating jobs under a series-parallel graph constraint," *Computers & Operations Research*, vol. 35, no. 8, pp. 2684–2693, 2008.

[8] M. M. Mazdeh, F. Zaerpour, A. Zareei, and A. Hajinezhad, "Parallel machines scheduling to minimize job tardiness and machine deteriorating cost with deteriorating jobs," *Applied Mathematical Modelling*, vol. 34, no. 6, pp. 1498–1510, 2010.

[9] L. Sun, L. Sun, K. Cui, and J. B. Wang, "A note on flow shop scheduling problems with deteriorating jobs on no-idle dominant machines," *European Journal of Operational Research*, vol. 200, no. 1, pp. 309–311, 2010.

[10] D. Wang and J. B. Wang, "Single-machine scheduling with simple linear deterioration to minimize earliness penalties," *International Journal of Advanced Manufacturing Technology*, vol. 46, no. 1–4, pp. 285–290, 2010.

[11] J.-B. Wang, "Flow shop scheduling with deteriorating jobs under dominating machines to minimize makespan," *International Journal of Advanced Manufacturing Technology*, vol. 48, no. 5–8, pp. 719–723, 2010.

[12] C.-M. Wei and J.-B. Wang, "Single machine quadratic penalty function scheduling with deteriorating jobs and group technology," *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3642–3647, 2010.

[13] X. Huang and M.-Z. Wang, "Parallel identical machines scheduling with deteriorating jobs and total absolute differences penalties," *Applied Mathematical Modelling*, vol. 35, no. 3, pp. 1349–1353, 2011.

[14] J.-B. Wang and C. Wang, "Single-machine due-window assignment problem with learning effect and deteriorating jobs," *Applied Mathematical Modelling*, vol. 35, no. 8, pp. 4017–4022, 2011.

[15] J.-B. Wang, J.-J. Wang, and P. Ji, "Scheduling jobs with chain precedence constraints and deteriorating jobs," *Journal of the Operational Research Society*, vol. 62, no. 9, pp. 1765–1770, 2011.

[16] J.-B. Wang and C.-M. Wei, "Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties," *Applied Mathematics and Computation*, vol. 217, no. 20, pp. 8093–8099, 2011.

[17] S.-H. Yang and J.-B. Wang, "Minimizing total weighted completion time in a two-machine flow shop scheduling under simple linear deterioration," *Applied Mathematics and Computation*, vol. 217, no. 9, pp. 4819–4826, 2011.

[18] J.-B. Wang and M.-Z. Wang, "Single-machine scheduling with nonlinear deterioration," *Optimization Letters*, vol. 6, no. 1, pp. 87–98, 2012.

[19] J.-B. Wang, M.-Z. Wang, and P. Ji, "Single machine total completion time minimization scheduling with a time-dependent learning effect and deteriorating jobs," *International Journal of Systems Science*, vol. 43, no. 5, pp. 861–868, 2012.

[20] B. Alidaee and N. K. Womer, "Scheduling with time dependent processing times: review and extensions," *Journal of the Operational Research Society*, vol. 50, no. 7, pp. 711–720, 1999.

[21] T. C. E. Cheng, Q. Ding, and B. M. T. Lin, "A concise survey of scheduling with time-dependent processing times," *European Journal of Operational Research*, vol. 152, no. 1, pp. 1–13, 2004.

[22] P. S. Sundararaghavan and A. S. Kunnathur, "Single machine scheduling with start time dependent processing times: some solvable cases," *European Journal of Operational Research*, vol. 78, no. 3, pp. 394–403, 1994.

[23] W. Kubiak and S. van de Velde, "Scheduling deteriorating jobs to minimize makespan," *Naval Research Logistics*, vol. 45, no. 5, pp. 511–523, 1998.

[24] T. C. E. Cheng and Q. Ding, "Single machine scheduling with step-deteriorating processing times," *European Journal of Operational Research*, vol. 134, no. 3, pp. 623–630, 2001.

[25] P. Hansen and N. Mladenović, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[26] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[27] A. S. Kunnathur and S. K. Gupta, "Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 56–64, 1990.

[28] G. Mosheiov, "Scheduling jobs with step-deterioration; Minimizing makespan on a single- and multi-machine," *Computers and Industrial Engineering*, vol. 28, no. 4, pp. 869–879, 1995.

[29] A. A. K. Jeng and B. M. T. Lin, "Makespan minimization in single-machine scheduling with step-deterioration of processing times," *Journal of the Operational Research Society*, vol. 55, no. 3, pp. 247–256, 2004.

[30] A. A. K. Jeng and B. M. T. Lin, "Minimizing the total completion time in single-machine scheduling with step-deteriorating jobs," *Computers and Operations Research*, vol. 32, no. 3, pp. 521–536, 2005.

[31] C.-C. He, C.-C. Wu, and W.-C. Lee, "Branch-and-bound and weight-combination search algorithms

for the total completion time problem with step-deteriorating jobs," *Journal of the Operational Research Society*, vol. 60, no. 12, pp. 1759–1766, 2009.

[32] J. Layegh, F. Jolai, and M. S. Amalnik, "A memetic algorithm for minimizing the total weighted completion time on a single machine under step-deterioration," *Advances in Engineering Software*, vol. 40, no. 10, pp. 1074–1077, 2009.

[33] M. Y. Kovalyov and W. Kubiak, "A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs," *Journal of Heuristics*, vol. 3, no. 4, pp. 287–297, 1998.

[34] T. C. E. Cheng, Q. Ding, M. Y. Kovalyov, A. Bachman, and A. Janiak, "Scheduling jobs with piecewise linear decreasing processing times," *Naval Research Logistics*, vol. 50, no. 6, pp. 531–554, 2003.

[35] G. Moslehi and A. Jafari, "Minimizing the number of tardy jobs under piecewise-linear deterioration," *Computers and Industrial Engineering*, vol. 59, no. 4, pp. 573–584, 2010.

[36] S. R. Gupta and J. S. Smith, "Algorithms for single machine total tardiness scheduling with sequence dependent setups," *European Journal of Operational Research*, vol. 175, no. 2, pp. 722–739, 2006.

[37] M. R. D. Paula, M. G. Ravetti, G. R. Mateus, and P. M. Pardalos, "Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search," *IMA Journal of Management Mathematics*, vol. 18, no. 2, pp. 101–115, 2007.

[38] D. Anghinolfi and M. Paolucci, "Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach," *Computers & Operations Research*, vol. 34, no. 11, pp. 3471–3490, 2007.

[39] R. Driessel and L. Mönch, "Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times," *Computers and Industrial Engineering*, vol. 61, no. 2, pp. 336–345, 2011.

[40] J. Behnamian, M. Zandieh, and S. M. T. F. Ghomi, "Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm," *Expert Systems with Applications*, vol. 36, no. 6, pp. 9637–9644, 2009.

[41] C.-L. Chen and C.-L. Chen, "Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times," *International Journal of Advanced Manufacturing Technology*, vol. 43, no. 1-2, pp. 161–169, 2009.

[42] J. Gao, L. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Computers & Operations Research*, vol. 35, no. 9, pp. 2892–2907, 2008.

[43] V. Roshanaei, B. Naderi, F. Jolaib, and M. Khalili, "A variable neighborhood search for job shop scheduling with set-up times to minimize makespan," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 654–661, 2009.

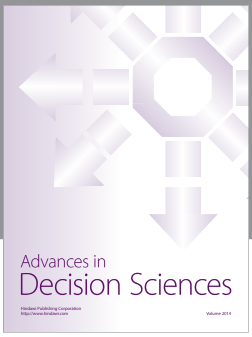
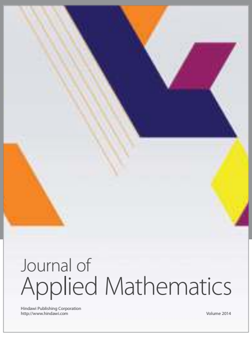
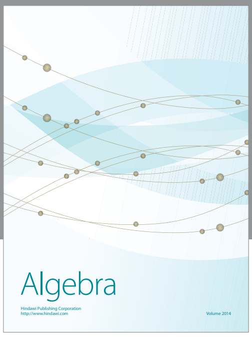
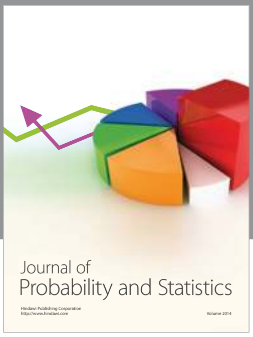
[44] M. A. Adibi, M. Zandieh, and M. Amiri, "Multi-objective scheduling of dynamic job shop using variable neighborhood search," *Expert Systems with Applications*, vol. 37, no. 1, pp. 282–287, 2010.

[45] M. Amiri, M. Zandieh, M. Yazdani, and A. Bagheri, "A variable neighbourhood search algorithm for the flexible job-shop scheduling problem," *International Journal of Production Research*, vol. 48, no. 19, pp. 5671–5689, 2010.

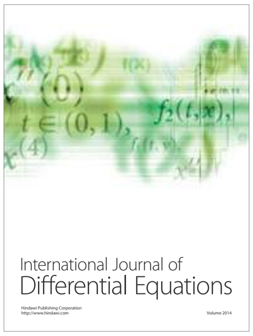
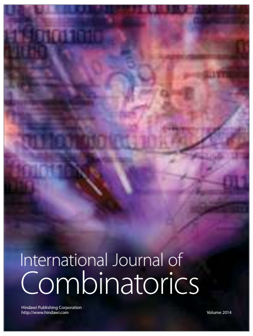
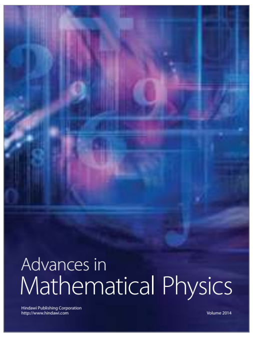
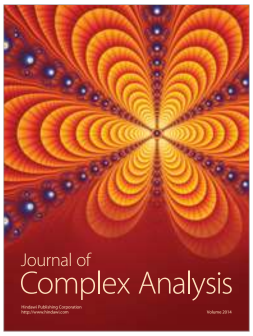
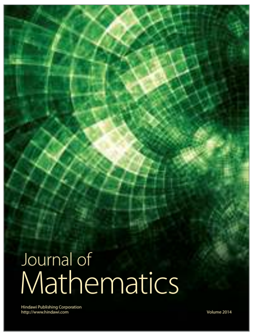
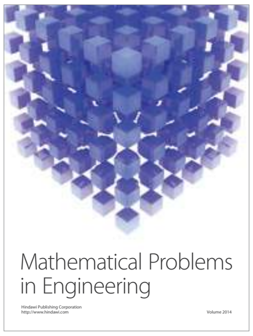
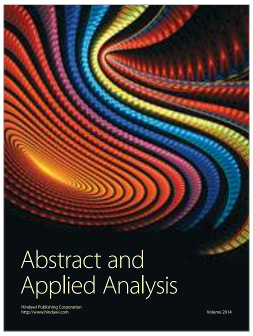
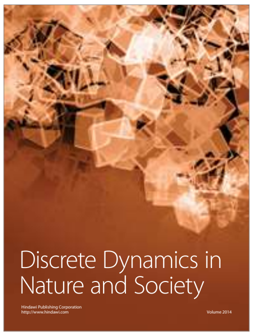
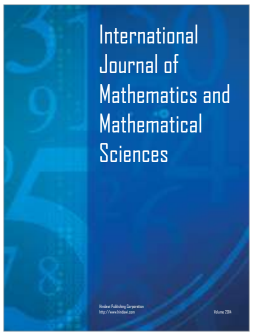
[46] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, pp. 367–407, 2010.

[47] M. L. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Springer, New York, NY, USA, 2008.

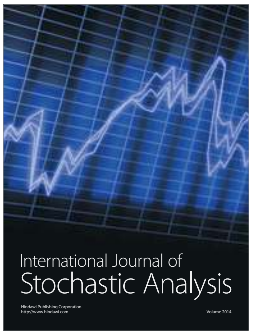
[48] C. Ferreira, "Combinatorial optimization by gene expression programming: inversion revisited," in *Proceedings of the Argentine Symposium on Artificial Intelligence*, pp. 160–174, Santa Fe, NM, USA, 2002.

Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

Algebra

Journal of
Probability and Statistics

The Scientific
World Journal


International Journal of
Differential Equations

International Journal of
Combinatorics

Advances in
Mathematical Physics

Journal of
Complex Analysis

Journal of
Mathematics

Mathematical Problems
in Engineering

Abstract and
Applied Analysis

Discrete Dynamics in
Nature and Society

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces

International Journal of
Stochastic Analysis

Journal of
Optimization