Journal of Kerbala University , Vol. 14 No.1 Scientific . 2016

Multi-Objective Variable Neighborhood Search Algorithms
دوال متعددة لمتغيرات بحث الجوار للخوارزميات

Tariq Salih Abdul-Razaq
Department of Mathematics,
College of Science,
University of Al-Mustansirih,
Baghdad,
Iraq.
dr.tariqsalih@yahoo.com

Hussam Abid Ali Mohammed
Department of Mathematics,
College of Education for Pure Science,
University of Kerbala,
Kerbala,
Iraq.
hussammath@yahoo.com

البحث مستل

Abstract

The Multi-Objective Single Machine Scheduling (MOSMS) Problem is one of the most representative problems in the scheduling area. In this paper, we compare five multi-objective algorithms based on Variable Neighborhood Search (VNS) heuristic. The algorithms are applied to solve the MOSMS problem. In this problem, we consider minimizing the total completion times and minimizing the sum of maximum earliness/tardiness. We introduce two intensification procedures to improve a Multi-Objective Variable Neighborhood Search (MOVNS) algorithms proposed in the literature. The performance of the algorithms is tested on a set of instances of the problem. The computational results show that the proposed algorithms outperform the original MOVNS algorithms in terms of efficiency solutions.

Keywords: Scheduling; Single machine; Maximum earliness/tardiness; Completion times; Efficient solution; Variable neighborhood search algorithms.

المستخلص

مسألة جدولة الدوال المتعددة الاهداف هي واحدة من المسائل الأكثر تمثيلا في مجال الجدولة. في هذا البحث، قارنا خمس خوارزميات لدوال متعددة بالاعتماد على أساس بحث متغيرات الجوار. تم تطبيق الخوارزميات لحل مسالة جدولة الماكنة الواحدة ذات دوال هدف متعددة. وفي هذه المسألة هدفنا هو تصغير من إجمالي وقت الاتمام ومجموع أكبر التبكير/التأخر. قدمنا طريقتي تكثيف لتحسين دوال متعددة لمتغيرات بحث الجوار للخوارزميات والمقترحة من قبل الباحثين. تم اختبار أداء الخوارزميات على مجموعة من الحالات للمسألة. وأظهرت النتائج الحسابية أن الخوارزميات المقترحة تفوق على الخوارزميات الأصلية من حيث كفاءة الحلول.

1. Introduction:

This article addresses the single-machine scheduling problem with distinct due dates and no idle time inserted. Performance is measured by minimization of the total completion times $(\sum_j C_j)$ and the sum of maximum earliness and tardiness $(E_{max} + T_{max} = ET_{max})$ of jobs. Single machine scheduling environments actually occur in several practical applications. The single machine scheduling problem (SMSP) has been extensively investigated during the last decades [1][2][3][4]. Most of the contributions consider a single optimization criterion, although in practice the Decision Maker often faces several (usually conflicting) criteria. The main criteria to be considered are the minimization of the two criteria have the same important in which we have look for the set of efficient solutions.

In this work we address the SMSP with total completion times and maximum earliness/ tardiness. This scheduling problem is a very important and frequent industrial problem that is common to most Just in Time (JIT) production environments. JIT consists in delivering products and services at the right time for immediate use, having as main objective the continuous search for improvement of the production process by $(\sum_j C_j)$ and (ET_{max}), which is obtained and developed through reduced inventories [5][6]. JIT scheduling problems are very common in industry. In the JIT scheduling environment, the job should be finished as close to the due date as possible. An early job completion results in inventory carrying costs, such as storage and insurance costs. On the other hand, a tardy job completion results in penalties, such as loss of customer goodwill and damaged reputation.

The most used methods for solving multi-objective combinatorial optimization problems (COP) are metaheuristics [7][8]. Metaheuristic methods were originally used for single-objective optimization and the success achieved in their application to a very large number of problems has stimulated researchers to extend them to multi-objective COP. Applications of the VNS metaheuristic for multi-objective optimization are scarce. To the best of our knowledge, the first multi-objective VNS (MOVNS) algorithm was proposed by Geiger [9]. In his work, the MOVNS was applied to solve the permutation flow shop scheduling problem minimizing different combinations of criteria. In [10] and [11] the MOVNS of Geiger was used to solve other multi-objective problems. Arroyo et al. [12] introduced two intensification procedures to improve a MOVNS algorithm. The algorithms are applied to solve the single machine scheduling problem with sequence dependent setup times and distinct due windows for the minimizing the total weighted earliness/tardiness and the total flowtime criteria.

In this work the objective is to determine feasible job schedules (efficient solutions) for the problem of minimizing the total completion times and the sum of maximum earliness/tardiness. The goal is to provide the decision maker with a set of efficient schedules (Pareto-optimal solutions) such that he/she may choose the most suitable schedule. We propose two algorithms based on VNS metaheuristic to solve the multi-objective SMSP. VNS is a stochastic local search method that is based on the systematic change of the neighborhood during the search. The proposed algorithms are based on the algorithm developed by Geiger [9]. We introduce two an intensification procedure based on constructing non-dominated solutions according to information taken on non-dominated partial solutions rather than evaluating complete solutions generated in the neighborhood of existing solutions. Simulation results and comparisons demonstrate the effectiveness, efficiency, and robustness of the proposed algorithms.

The remainder of this paper is organized as follows. The Basic concepts of the multi-objective problem is given in Section 2. Section 3 the multi-objective problem definition is described. Characterize the set of efficient solutions in section 4. Section 5 provides description of the MOVNS algorithms. Results of computational experiments to evaluate the performance of the proposed algorithms are reported in Section 6. Finally, Section 7 provides the concluding remarks.

2. Basic concepts:

We recall the following fundamental results for the three single objective problem.

Theorem(1) [13]: The $1\|\sum_j C_j$ problem is minimized by sequencing the jobs according to the shortest processing time SPT-rule, that is, in order of nondecreasing p_j.

Theorem(2) [14]: The $1\|T_{max}$ problem is minimized by sequencing the jobs according to the earliest due date EDD-rule, that is, in order of nondecreasing d_j.

Theorem(3) [15]: The $1\|E_{max}$ problem is solved by sequencing the jobs according to the minimum slack time MST-rule, that is, in order of nondecreasing $d_j - p_j$.

The proof of each of these results proceeds by a straightforward interchange argument. Note that each of these problems is solved by arranging the jobs in a certain priority order that can be specified in terms of the parameters of the problem type.

The optimal solution values for these single machine scheduling problems will be denoted by $\sum_j C_j^*$, E_{max}^* and T_{max}^*, respectively. Furthermore, $\sum_j C_j\,(s)$, $E_{max}(s)$ and $T_{max}(s)$ are the objective values for the schedule s. In analogy, $C_j(s)$, $E_j(s)$ and $T_j(s)$ denote the respective measures for job $s(i)(i = 1, \dots, n)$. Whenever (s) is omitted, we are considering the performance measure in a generic sense, or there is no confusion possible as to the schedule we are referring to. The schedules that minimize $\sum_j C_j$, E_{max} and T_{max} are referred to as SPT, EDD, and MST rules respectively. Multi-objective scheduling refers to the scheduling problem in which the advantages of a particular schedule are evaluated using more than one performance criterion.

Definition(1) (*optimize*) [16]: The term "optimize" in a multi-objective decision making problem refers to a solution around which there is no way of improving any objective without worsening at least one other objective.

Definition(2) (*efficient*) [16]: A schedule s is said to be efficient if there does not exist another schedule s' satisfying $f_j(s') \leq f_j(s)\,, j = 1, \dots, k$ with at least one of the above holding as a strict inequality. Otherwise s is said to be dominated by s'.

Definition(3) (*neighborhood*) [17]: A neighborhood function N^* is a mapping $N^*: S \longrightarrow P(S)$ which specifies for each $s \in S$ a subset $N^*(s)$ of S neighbors of s.

Theorem(4) [16]: If the composite objective function $F(f, g)$ is non-decreasing in both argument, then there exists a Pareto optimal schedule that minimize F.

3. Multi-Objective Single Machine Scheduling (MOSMS) Problem Description

The MOSMS problem examined in this paper is stated as follows. There is a set of n jobs to be processed on a continuously available single machine. The machine can process only one job at a time. Each job j is available for processing at time zero, has a known processing time p_j, a due date d_j. The completion time C_j is cumulative total for the processing times from p_1 to p_j. The earliness and the tardiness are computed as $E_j = max\{0, d_j - C_j\}$ and $T_j = max\{0, C_j - d_j\}$, respectively.

The objective of the problem is to determine feasible schedules with minimum total flow time (f_1) and maximum earliness and maximum tardiness of the jobs (f_2). For a sequence of jobs (permutation of the n jobs) $s \in S$ where S is the set of all feasible solutions, the criteria (f_1) and (f_2) are computed in the following as:

Let $E_{max} = max_{1 \leq j \leq n}\{E_j\}$ and $T_{max} = max_{1 \leq j \leq n}\{T_j\}$

$$
Min \begin{cases} f_1(s) = \sum_{j=1}^{n} C_j & \dots (1) \\ f_2(s) = E_{max} + T_{max} = ET_{max} & \dots (2) \end{cases}
$$

$$
\left.\begin{array}{ll} s.t. & \\ C_j \geq p_j, & j = 1, 2, \dots, n \\ E_j \geq d_j - C_j, & j = 1, 2, \dots, n \\ T_j \geq C_j - d_j, & j = 1, 2, \dots, n \\ E_j \geq 0, & j = 1, 2, \dots, n \\ T_j \geq 0, & j = 1, 2, \dots, n \end{array}\right\} (P)
$$

In the addressed problem the occurrence of machine idle time is not allowed. The multi-criteria problem that we consider concerns the simultaneous minimization of the performance measures (1) and (2). The problem (P) is NP-hard since the problem $1\|ET_{max}$ is NP- hard [18].

4. Characterizing Efficient Solutions:

The problem is to characterize the set of efficient (Pareto optimal) solutions.

Proposition(1): There exists an efficient sequence in the multi-criterion problem (P) that satisfies SPT-rule.

Proof: (a) Suppose first that all processing times are different. The unique SPT-sequence (SPT^*) gives the absolute minimum of $\sum_j C_j$. Hence, there is no sequence $\pi \neq SPT^*$ such that

$$\sum_j C_j(\pi) \leq \sum_j C_j(SPT^*) \text{ and } ET_{max}(\pi) \leq ET_{max}(SPT^*) \quad \ldots(3)$$

with at least one strict inequality since (3) cannot hold.

(b) If more than one SPT-sequence (jobs with equal processing times), lets SPT^* be a sequence satisfying the SPT-rule and such that the jobs with equal processing times are ordered in EDD-sequence or MST-sequence such that $E_{max} + T_{max}$ for these jobs as minimum as possible. Note that if several jobs have identical processing times and identical slack times (see Proposition (3)), SPT^* is not unique. We prove that every SPT^*-sequence is efficient sequence, that do not satisfy the SPT-rule cannot dominate an SPT^*-sequence by (3) and if π is an SPT-sequence but not an SPT^*-sequence, it cannot dominate SPT^* since

$\sum_j C_j(SPT^*) = \sum_j C_j(\pi)$ and $ET_{max}(SPT^*) \leq ET_{max}(\pi)$

by virtue of the EDD-rule or MST-rule. ■

Lemma(1) [19]: In the problem $1\|ET_{max}$, the best common due date is assigned in each position within time space between the minimum completion time (C_{min}) and the maximum completion time (C_{max}).

Lemma(2): If $d_j = d\ \forall j$, then the SPT-rule is efficient solution for $1|d_j = d|(\sum_j C_j, ET_{max})$ problem (P) with $(\sum_j C_j, ET_{max}) = (\sum_j C_j(SPT), C_n - C_1)$.

Proof: By Proposition (1) the SPT-rule is efficient sequence. And if Lemma (1) is satisfied then we have

$ET_{max} = C_n - C_1$, where $C_n = C_{max} = \sum_{j=1}^{n} p_j$ and $C_1 = C_{min} = min\{p_j\} = p_1$ and $C_1 < d < C_n$.

Therefore, $(\sum_j C_j, ET_{max}) = (\sum_j C_j(SPT), C_n - C_1)$. ■

Proposition(2): If SPT-rule and EDD-rule have the same ordered and all jobs are tardy then this order is the only efficient solution for (P).

Proof: Suppose that s be the SPT-sequence and EDD-sequence and all jobs are tardy, which yields that $E_j = 0, j \in N$, i.e. $(\sum_j C_j, ET_{max}) = (\sum_j C_j, T_{max})$. Hence the sequence s gives $\sum_j C_j(s)$ is optimal since s is SPT-sequence and $T_{max}(s)$ is optimal since s is EDD-sequence.

Hence s is the only efficient solution for (P). ■

Proposition(3): If SPT-rule and MST-rule have the same ordered s then this order gives the only efficient solution for (P).

Proof: Suppose that s be the SPT-sequence and MST-sequence and for every adjacent jobs i and j we have:

$$p_i \leq p_j, \forall\ i, j \quad \ldots (4)$$

and

$$d_i - p_i \leq d_j - p_j, \forall\ i, j \quad \ldots (5)$$

Hence $\sum_j C_j(s)$ is minimum since s is SPT-sequence and $E_{max}(s)$ is minimum since s is MST-sequence.

For the $T_{max}(s)$, we have from (5)

$d_i - p_i + p_i \leq d_j - p_j + p_i$

$d_i \le d_j - (p_j - p_i) = d_j - p$, since $p = p_j - p_i \ge 0$ by using (4). Hence
$d_i \le d_j, \forall\, i,j.$
Hence $T_{max}(s)$ is minimum by EDD-rule.
Therefore the s-sequence is the only efficient solution for (P). ■

Lemma(3): For any adjacent jobs i and j such that $p_i \le p_j$ and $d_i - p_i \le d_j - p_j$, then job i precede job j, in any efficient schedule.
Proof: It is clear from Proposition(3). ■

Remark(1): If MST-rule and EDD-rule have the same ordered, then this order does not necessary give efficient solution for the multi-criterion problem (P) as shown by the following example.

Example(1): If we have the processing times $p_j = (1,3,2)$ and due date $d_j = (3,5,5)$, and the schedule $s = (1,2,3)$ satisfy the MST-rule and EDD-rule, which gives $\left(\sum_{j=1}^{3} C_j\,, ET_{max}\right) = (11,3)$, where the schedule $s' = (1,3,2)$ is not satisfy the MST-rule which gives $\left(\sum_{j=1}^{3} C_j\,, ET_{max}\right) = (10,3)$. It is clear that s is not efficient solution since it dominated by the sequence s'.

Theorem(5): In the problem (P) with common due date d the number of efficient solutions $|Ef|$ is

$$|Ef| = \begin{cases} 1 & if \quad d \le p_{min} = min\{p_i\} \\ k \text{ (job different proc. times)} & if \quad d \ge C_n \; or \; d \ge p_{max} = max\{p_i\} \\ i & if \quad p_{[1]} < d < p_{[i]} \;\&\; 2 \le i < k \end{cases}$$

Proof: Suppose that the sequence $s = ([1], \dots, [n])$ is obtained by SPT-rule, and the partial sequence $\mathcal{P} = ([1], \dots, [k])$ of s with k jobs different processing times such that $k \le n$, $p_{[i]} < p_{[j]}, i < j$ and $p_{[1]} = p_{min}$ and $p_{[k]} = p_{max}$, then the following cases are investigated.

Case(1) $d \le C_1$:

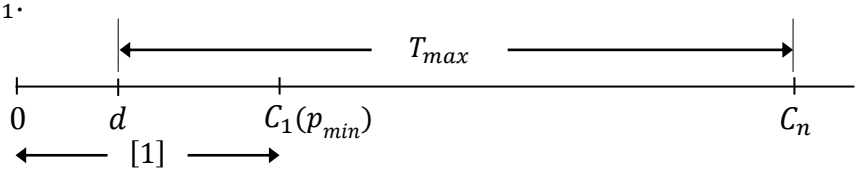

Figure1 Case(1)

If due date d is before $C_1 = C_{min} = p_{[1]}$ (Fig.1), then all jobs are tardy and maximum tardiness is $T_{max} = C_n - d$. Thus, ET_{max} is given by:
$ET_{max} = T_{max} = C_n - d.$
Therefore, we have only one efficient solution for the problem (P) with $\left(\sum_j C_j\,, ET_{max}\right) = \left(\sum_j C_j\,, C_n - d\right)$.

Case(2) $d \ge C_n$:

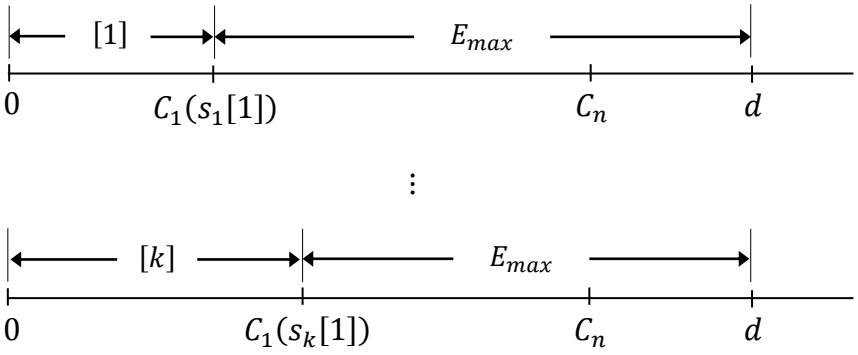

Figure2 Case(2)

If due date d is after C_n (Fig.2), then all jobs are early and maximum earliness is $E_{max} = d - C_1$. Thus, ET_{max} is given by:

$ET_{max} = E_{max} = d - C_1$.

Now, suppose that $\mathcal{P}$ is the sequence of all jobs with different processing times such that $p_{[i]} < p_{[j]}$, $\forall\, i < j$. Let $p_{[1]} \in \mathcal{P}$ and $p_{[2]} \in \mathcal{P}$ such that s_1 is the schedule with $C_1(s_1[1]) = p_{[1]}$ for the first job and the remaining jobs of s_1 ordered by SPT-rule and the same think for s_2 schedule with $C_1(s_2[1]) = p_{[2]}$ as the first job and the remaining jobs of s_2 ordered by SPT-rule, then we have for the criteria (f_1) and (f_2):

$$\sum_{j=1}^{n} C_j(s_1) = C_1(s_1[1]) + \sum_{j=2}^{n} C_j\big(s_1(SPT)\big) < \sum_{j=1}^{n} C_j(s_2) = C_1(s_2[1]) + \sum_{j=2}^{n} C_j\left(s_2(SPT)\right)$$

and

$d - C_1(s_1[1]) > d - C_1(s_2[1])$,

where ($C_n = \sum_{j=1}^{n} p_j$, $C_1(s_1[1])$ and $C_1(s_2[1])$ be the completion times of the job in the first position in s_1 and s_2 respectively and $C_1(s_2[1]) > C_1(s_1[1])$). Now using the same techniques for the schedules $s_3, \dots, s_k$ for the other jobs in $\mathcal{P}$ (see Fig.2). Then we have:

$$C_1(s_1[1]) + \sum_{j=2}^{n} C_j\big(s_1(SPT)\big) < C_1(s_2[1]) + \sum_{j=2}^{n} C_j\big(s_2(SPT)\big) < \cdots$$

$$< C_1(s_k[1]) + \sum_{j=2}^{n} C_j\big(s_k(SPT)\big)$$

Hence, in general the schedules $s_1, \dots, s_k$ and with $d \geq C_n$, we have

$$\sum_{j=1}^{n} C_j(s_1) < \sum_{j=1}^{n} C_j(s_2) < \cdots < \sum_{j=1}^{n} C_j(s_k)$$

and

$d - C_1(s_1[1]) > d - C_1(s_2[1]) > \cdots > d - C_1(s_k[1])$

Hence $ET_{max}(s_1) > ET_{max}(s_2) > \cdots > ET_{max}(s_k)$ i.e. $E_{max}(s_1) > E_{max}(s_2) > \cdots > E_{max}(s_k)$ since $T_{max}(s_i) = 0, i = 1, \dots, k$.

Therefore the schedules $s_1, s_2, \dots, s_k$ are efficient solutions for the k jobs of different processing times, i.e. $|Ef| = k$.

Case(3) $C_1 \leq d < C_n$: If common due date d is between $C_1\big(p_{[1]}\big)$ and C_n, then the first job and the last job have the maximum earliness and maximum tardiness, respectively. By considering the job position with the maximum earliness and maximum tardiness, in the first and the last position in the

sequence respectively, equations $E_{max} = d - C_1$ and $T_{max} = C_n - d$ are obtained and resulting the following equation.

$ET_{max} = E_{max} + T_{max} = C_n - C_1$; hence we have two subcases to be considered:

(a) $d \geq p_{[k]}$:

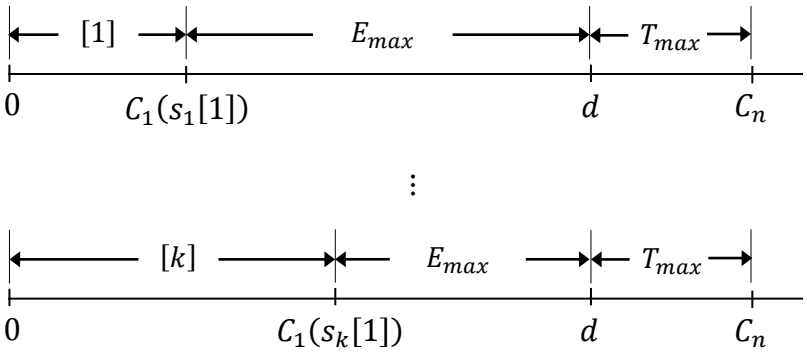

Figure3 Case(3)(a)

Suppose we have the schedules $s_1, s_2, \dots, s_k$ with $d \geq p_{max}$, where k jobs different processing times as in Case(2), then we have for the schedules s_1 and s_2:

$$\sum_{j=1}^{n} C_j(s_1) = C_1(s_1[1]) + \sum_{j=2}^{n} C_j\big(s_1(SPT)\big) < \sum_{j=1}^{n} C_j(s_2) = C_1(s_2[1]) + \sum_{j=2}^{n} C_j\left(s_2(SPT)\right)$$

and

$C_n - C_1(s_1[1]) > C_n - C_1(s_2[1])$

Then using the same techniques for the schedules $s_3, \dots, s_k$ for the other jobs in $\mathcal{P}$ (see Fig.3). Then we have for the criteria (f_1) and (f_2):

$$C_1(s_1[1]) + \sum_{j=2}^{n} C_j\big(s_1(SPT)\big) < C_1(s_2[1]) + \sum_{j=2}^{n} C_j\big(s_2(SPT)\big) < \cdots$$

$$< C_1(s_k[1]) + \sum_{j=2}^{n} C_j\big(s_k(SPT)\big)$$

Hence

$$\sum_{j=1}^{n} C_j(s_1) < \sum_{j=1}^{n} C_j(s_2) < \cdots < \sum_{j=1}^{n} C_j(s_k)$$

and

$C_n - C_1(s_1[1]) > C_n - C_1(s_2[1]) > \cdots > C_n - C_1(s_k[1])$

$ET_{max}(s_1) > ET_{max}(s_2) > \cdots > ET_{max}(s_k)$

Therefore the schedules $s_1, s_2, \dots, s_k$ are efficient solutions for the k jobs of different processing times, i.e. $|Ef| = k$.

(b) $p_{[1]} < d$ and $d = p_{[i]}$: Where $p_{[j]}, j = 2, \dots, i, i < k$ is the processing time for the first job of the schedules $s_1, s_2, \dots, s_i$

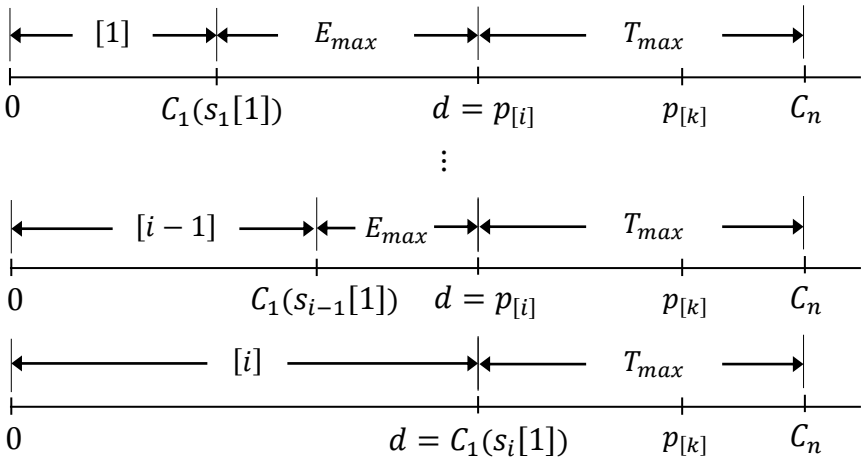

Figure3 Case(3)(b)

Let $i-1$ and i $(i < k)$ be two adjacent jobs in $\mathcal{P}$ such that $p_{[i]} < d$ and $d = p_{[i]}$, let the schedules $s_1, s_2, \dots, s_k$ as in Case(2), then we have:

$$C_1(s_1[1]) + \sum_{j=2}^{n} C_j\big(s_1(SPT)\big) < \dots < C_1(s_{i-1}[1]) + \sum_{j=2}^{n} C_j\big(s_{i-1}(SPT)\big)$$
$$< C_1(s_i[1]) + \sum_{j=2}^{n} C_j\big(s_i(SPT)\big) < C_1(s_{i+1}[1]) + \sum_{j=2}^{n} C_j\big(s_{i+1}(SPT)\big) < \cdots$$
$$< C_1(s_k[1]) + \sum_{j=2}^{n} C_j\big(s_k(SPT)\big)$$

Hence

$$\sum_{j=1}^{n} C_j(s_1) < \cdots < \sum_{j=1}^{n} C_j(s_{i-1}) < \sum_{j=1}^{n} C_j(s_i) < \sum_{j=1}^{n} C_j(s_{i+1}) < \cdots < \sum_{j=1}^{n} C_j(s_k)$$

and

$$C_n - C_1(s_1[1]) > \dots > C_n - C_1(s_{i-1}[1]) > C_n - C_1(s_i[1]) = C_n - d = \cdots = C_n - d$$
$$ET_{max}(s_1) > \cdots > ET_{max}(s_{i-1}) > ET_{max}(s_i) = T_{max}(s_i) = T_{max}(s_{i+1}) = \cdots = T_{max}(s_k)$$

It is clear that the sequence s_i is efficient solution which dominate the sequence s_{i+1}, when $d = p_{[i]}$ i.e. all jobs $i \in \mathcal{P} = ([1], \dots, [k])$ such that $2 \le i < k$ are tardy and with maximum tardiness is given by $C_n - d$. Thus, $ET_{max}(s_i)$ is given by:

$ET_{max}(s_i) = T_{max}(s_i) = C_n - d.$

The jobs $([1], \dots, [i]) \in \mathcal{P}$ have processing times less than or equal d, therefore the sequences $s_1, s_2, \dots, s_i$ are efficient solutions for the i jobs of different processing times, i.e. $|Ef| = i$.

Case(4) $p_{[i-1]} < d < p_{[i]}$: $2 \le i < k$

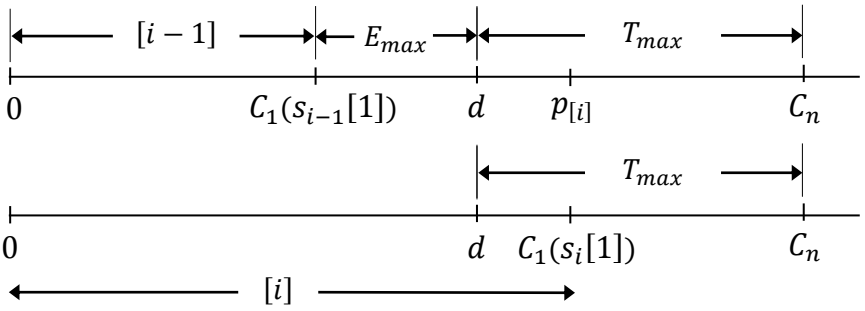

Figure4 Case(4)

From Case(3) (b) we have $|Ef| = i - 1$ since $p_{[1]} < d < p_{[i]}$ and $d \neq p_{[i]}$, and from Case(1) we have $|Ef| = 1$ since $d < p_{[i]}$, therefore we have $|Ef| = i$. ■

Remark(2): For the common due date d problem, suppose that $s = ([1], \dots, [n])$ is obtained by SPT-sequence. Let $\mathcal{P}$ be a subsequence of s of all different processing times with k jobs. We can find all efficient solutions by the following Algorithm1 if the first efficient solution s given by SPT rule $\left(\sum_j C_j , ET_{max}\right) = \left(\sum_j C_j(s) , ET_{max}(s)\right)$:

Algorithm1

1. $s_1 \leftarrow s$;
2. $h \leftarrow 1$;
3. ***for*** $l \leftarrow 2$ *to* k $(or\ i)$ % $d \le p_{[i]}$

a. ***if*** $l \leftarrow 2$

i. $\sum_j C_j(s_l) \leftarrow \sum_j C_j(s_{l-1}) + p_{[l]} - p_{[1]}$;

b. ***else***

i. $h \leftarrow h + 1$;

ii. $\sum_j C_j(s_l) \leftarrow \sum_j C_j(s_{l-1}) + h\left(s_l(p_1) - s_{l-1}(p_1)\right)$;

c. ***end***

d. $ET_{max}(s_l) \leftarrow ET_{max}(s_1) + p_{[1]} - p_{[l]}$; % $p_{[1]}, p_{[l]} \in \mathcal{P}$

4. ***end***

Example(2): The following examples explain the cases of Theorem(5) and satisfy the Remark(2) (Algorithm1) also.

	Conditions	$p_i = [4,5,7,8]$	Seq	$\sum C_j$	E_{max}	T_{max}	$\left(\sum_j C_j , ET_{max}\right)$	$\|Ef\|$
Case(1)	$d \le p_{min}$	$d = 3$	1 2 3 4	53	0	21	(53,21)	1
Case(2)	$d \ge C_n$	$d = 25$	1 2 3 4	53	21	0	(53,21)	4
			2 1 3 4	54	20	0	(54,20)	
			3 1 2 4	58	18	0	(58,18)	
			4 1 2 3	61	17	0	(61,17)	
Case(3)(a)	$C_1 \le d < C_n$ $d \ge p_{[4]}$	$d = 11$	1 2 3 4	53	7	13	(53,20)	4
			2 1 3 4	54	6	13	(54,19)	
			3 1 2 4	58	4	13	(58,17)	
			4 1 2 3	61	3	13	(61,16)	
Case(3)(b)	$C_1 \le d < C_n$ $p_{[1]} < d = p_{[3]}$	$d = 7$	1 2 3 4	53	3	17	(53,20)	3
			2 1 3 4	54	2	17	(54,19)	
			3 1 2 4	58	0	17	(58,17)	
Case(4)	$p_{[2]} < d < p_{[3]}$	$d = 6$	1 2 3 4	53	2	18	(53,20)	3
			2 1 3 4	54	1	18	(54,19)	
			3 1 2 4	58	0	18	(58,18)	

For the Remark(2), consider the Case(2), the SPT solution with the first efficient solution is $\left(\sum C_j\,(SPT), ET_{max}(SPT)\right) = (53,21)$, then the second efficient solution is $\left(\sum C_j\,(SPT) + p_{[2]} - p_{[1]}, ET_{max}(SPT) - p_{[2]} + p_{[1]}\right) = (53+5-4, 21-5+4) = (54,20)$. The third efficient solution $\left(\sum C_j\,(s_2) + h\left(p_{[3]} - p_{[2]}\right), ET_{max}(SPT) - p_{[3]} + p_{[1]}\right) = (54 + 2(7-5), 21-7+4) = (58,18)$.

Also it easily to get the remaining efficient solution by the same way which is $(61,17)$.

Lemma(4): In the problem (P), if $T_{max}(SPT) = 0$, then $E_{max} \in [E_{max}(MST), E_{max}(SPT)]$.

Proof: In the problem (P), the smallest value of E_{max} is $E_{max}(MST)$.

For the largest value of E_{max}, let s be any sequence such that $E_{max}(SPT) \leq E_{max}(s)$ that implies s is not efficient since $F\left(\sum_j C_j\,(SPT), E_{max}(SPT)\right) \leq F\left(\sum_j C_j\,(s), E_{max}(s)\right)$ where $\sum_j C_j\,(SPT) \leq \sum_j C_j\,(s)$ by the SPT rule.

Therefore $E_{max} \in [E_{max}(MST), E_{max}(SPT)]$. ■

Remark(3): Consider the $1||\sum_j C_j + ET_{max}$ problem $(P1)$ which is special case of problem (P). If there exists optimal solution s for the problem $(P1)$, then this solution s is Pareto optimal for the problem (P). And vice versa if a set S is the set of all Pareto optimal solutions to the problem (P), then there exists optimal solution $s \in S$ for the problem $(P1)$.

5. Multi-Objective Variable Neighborhood Search (MOVNS) Algorithms

Geiger [9] was developed the first application of the VNS metaheuristic for multi-objective optimization. The VNS provides approach high quality solution to NP-hard problems of realistic size in reasonable time. The VNS algorithm was randomly selection of neighborhoods and arbitrary selection of the base solution and then continually try to add better solution by searching neighborhood.

In this paper, we applied the MOVNS1 algorithm of Geiger [9] to solve the MOSMS problem (P) defined in Section 3. Also we propose the MOVNS2 and MOVNS3 are new modified algorithms and MOVNS4, MOVNS5 which were proposed by Arroyo et al. [12]. The MOVNSi $(i = 1, \dots, 5)$ use simple heuristics to generate three initial solutions s_1, s_2 and s_3 obtained by SPT, EDD and MST rules respectively, and put it in the non-dominate set S. N_1 and N_2 be two neighborhood structures which used to generate new solutions. For each iteration of the algorithm, a base non-dominated solution is randomly selected from S, this solution is removed from S. From the base solution, neighbor solutions are generated by chosen randomly neighborhood N_i $(i = 1,2)$. The set S is update with the solutions $s'' \in N_i$ by generated a set of population (pop). Solutions s'' are added to the set S if $s'' \notin S$ and there are not dominated by any solution in S, and the solutions of S dominated by s'' are removed from S.

The main work in this paper to improve a non-dominate solution and base on the previous algorithms of Geiger [9] and Arroyo et al. [12]. The pseudocode description of the proposed MOVNS2 algorithm is presented in Algorithm2 and Subalgorithm1. Also the MOVNS3 algorithm is presented in Algorithm2 and Subalgorithm2. The Subalgorithm1 and Subalgorithm2 has an input parameter c used in the intensification phase (c is the number of jobs to be cut from a sequence s). From a non-dominated neighbor solution s, new non-dominated solutions are constructed by the intensification procedure. This procedure is based on two typical approaches used in multi-objective optimization: dynamic weighted aggregation and Pareto dominance. The first approach (Subalgorithm1) is according to the objectives, f_1 and f_2 are summed to a weighted combination $f_w = min(w_1 f_1 + w_2 f_2)$, where the weights w_1 and w_2 are non-negative weights and $w_1 + w_2 = 1$. These weights can be either fixed or adapt dynamically during the optimization. The dynamic weighted aggregation [20], which we used in this paper. Alternatively, the weight can be changed gradually according to the $w_1(t) = \left|sin\left({}^{2\pi t}/_{R}\right)\right|$ and $w_2(t) = 1 - w_1(t)$, where t is the iteration's index, in our work t is a time in $[0, n]$ (n = number of jobs) and R$(= 200)$ is the weights' change frequency. The second approach (Subalgorithm2) is Pareto dominance approach, only non-dominated solutions are considered analyzed.

The algorithms were run with the same stopping criterion (StoppingCriterion) based on an amount of CPU time. This time is giving by n seconds and it depends on the size of the considered instance.

Algorithm2

1. $S \leftarrow \{s_1, s_2, s_3\}$;
2. **while** *until time is finish in sec*

a. $s \leftarrow rand(S)$;
b. $S \leftarrow S \backslash \{s\}$;
c. $N \leftarrow N_i$; % $i = 1$ *or* 2
d. $s' \leftarrow N(s)$;
e. $S_\alpha \leftarrow \emptyset$ *(empty set)*;
f. **for** $j \leftarrow 1$ *to pop*

i. $s'' \leftarrow N(s')$;
ii. **if** s'' *non − dominated solution*

I. $S_\alpha \leftarrow S_\alpha \cup \{s''\}$;

iii. **end**

g. **end**
h. $s \leftarrow rand(S_\alpha)$;
k. $S_\beta \leftarrow$ **Call**(Subalgorithm1 Intensification1 (s, c)); % *for* MOVNS2 *or* $(Subalgorithm2\ Intensification2(s, c))$ *for* MOVNS3
l. $S \leftarrow S \cup S_\alpha \cup S_\beta$;
3. **end**

The pseudocodes of the algorithms Intensification1 and Intensification2 are presented in Subalgorithm1 and Subalgorithm2, respectively.

Subalgorithm1 Intensification1 (s, c)

1. $S_\beta \leftarrow \emptyset$;
2. $w_1(t) = \left|sin\left(2\pi t/_R\right)\right|$ & $w_2(t) = 1 - w_1(t)$;
3. *Cut s to partial sequences in c*;
4. $s_p \leftarrow$ *second part from s and first part in* s_R; %s_p *is a partial sequence with* $n - c$ *jobs*;
5. **for** $i \leftarrow 1$ *to* c

a. $s_p \leftarrow s_R(end) \cup s_p$; & $s_R \leftarrow s_R \backslash s_R(end)$;
b. **for** $j \leftarrow 1$ *to* $n - c + i$

i. **if** $p_j \leq p_{j+1}$ *and* $d_j - p_j \leq d_{j+1} - p_{j+1}$

I. *Generating a set A of* $(n - c + i)$ *sequences by* $TH(N_1)$;

ii. **end**
iii. *Evaluate each sequence* $s' \in S$;
iv. **if** $c < n$

I. $s_p \leftarrow$ *best solution from A (with respect to* $f_w = min(w_1 f_1 + w_2 f_2)$);

v. **else**

I. $S_\beta \leftarrow$ *non − dominated solutions obtained of* $S_\beta \cup \{s'\}$;

vi. **end**

c. **end**
6. **end**
7. *Return* S_β

Subalgorithm2 Intensification2 (s, c)

1. *Cut s to partial sequences in c*;
2. $s_p \leftarrow$ *second part from s and first part in* s_R;
3. $S_\beta \leftarrow \{s_p\}$;
4. **for** $i \leftarrow 1$ *to* c

a. $S_\alpha \leftarrow \emptyset$;
b. **for** *each partial sequence* $s_p \in S_\beta$

i. $s_p \leftarrow s_R(end) \cup s_p$; & $s_R \leftarrow s_R \backslash s_R(end)$;
ii. **for** $j \leftarrow 1$ *to* $n - c + i$

I. **if** $p_j \leq p_{j+1}$ *and* $d_j - p_j \leq d_{j+1} - p_{j+1}$

A. *Generating a set A of* $(n - c + i)$ *sequences by* $TH(N_1)$;

II. **end**
III. *Evaluate each sequence* $s' \in A$;

IV. $S_\alpha \leftarrow non-dominated\ solutions\ obtained\ of\ S_\alpha \cup \{s'\}$;
iii. ***end***
iv. $S_\beta \leftarrow S_\alpha$;
c. ***end***
5. ***end***
6. $Return\ S_\beta$

In the next subsections, we describe in detail each phase of the MOVNS algorithms.

5.1 Initial Non-dominated Solutions

In this paper, we use simple dispatching rules to generate initial non-dominated solutions (instead of generating random solutions). In the algorithms MOVNS2 and MOVNS3, the set S of non-dominated solutions is initialized with three solutions (sequences): s_1, s_2 and s_3 obtained by SPT, EDD and MST rules respectively. The non-dominated solutions of $\{s_1, s_2, s_3\}$ are stored in the set S. This set will contain at least one solution, to start with this solution is obtained by Proposition(1).

5.2 Variable Neighborhood Structures and Local Search Algorithms

Local search algorithms usually are based on neighborhood search. These algorithms move from solution to solution in the space of candidate solutions (the search space) by applying local change (neighborhood), until a solution deemed optimal is found or a time bound is elapsed.

In this paper, the MOVNS algorithms used two neighborhood structures: transpose and exchange. For a given sequence $s = (s(1), \dots, s(n))$, the neighborhood structures are described below.

Transpose neighborhood TN(N_1): the neighbors of s are generated by interchanging jobs $s(q)$ and $s(q+1)$ in the sequence, $1 \le q \le n-1$. $N_1(s)$ neighborhood has size $(n-1)$.

Exchange neighborhood EN(N_2): the neighbors of s are generated by interchanging jobs $s(q)$ and $s(p)$ in the sequence, $1 \le q \le n$, $1 \le p \le n$, $q \ne p$. $N_2(s)$ neighborhood has size $n(n-1)/2$.

The MOVNS algorithms start with a base solution s select randomly from the current set of non-dominated solutions S. The selected solutions are removed as visited and it will be excluded from the selection of the base one.

In each iteration of the algorithms, a neighborhood structure $N_i, i = 1,2$ is selected randomly. The base solution s is perturbed by choosing randomly a solution s' from $N_i(s)$ neighborhood (shaking). Then, all the neighboring solutions of s' are analyzed, that is, the neighborhood $N_i(s')$ is explored.

In the Algorithm2, the non-dominated neighbor solutions are stored in a set S_α. From a solution selected randomly from S_α, the intensification procedures are executed.

5.3 Improving Procedures

Arroyo et al. [12] proposed two intensification procedures: scalarizing function with arbitrary fixed weighted (MOVNS4) and Pareto dominance (MOVNS5).

In this paper a new intensification procedures are used, in order to improve a non-dominated solution selected randomly from set S_α (set of non-dominated solutions). The proposed algorithms are compare with the algorithms developed by Geiger [9] and Arroyo et al. [12].

The two propose intensification procedures are composed of two stages: destruction and construction as shown in following example for $n = 5$ and $c = 2$ which are given in Fig.5 and Fig.6. In the destruction stage, c jobs (selected randomly) are cut from s (solution selected randomly from S_α) and a partial solution s_p (of size $n-c$) is obtained. The first part jobs are stored in s_R ($s_R(j), j = 1, \dots, c$, are the deducted jobs). The construction stage has c steps. In step $j = c$, inserting job $s_R(c)$ in the first possible position of s_p and using the TN(N_1) for new s_p, then we generating a set A of $(n-c+1)$ partial sequences.

In the Intensification1 algorithm, from A (set of non-dominated solution), have the $(n-c+1)$ partial solutions, the best is chosen (one that has the lowest value of f_w) and it replaces s_p. The other steps, $i = 1, \dots, c-1$, are similar. Note that, in step $i = c-1$ of the example, n complete solutions are constructed. From these n complete solutions, the set S_β (of non-dominated solutions) is determined. Fig.5 illustrate the idea of the Intensification1 procedure for the example.

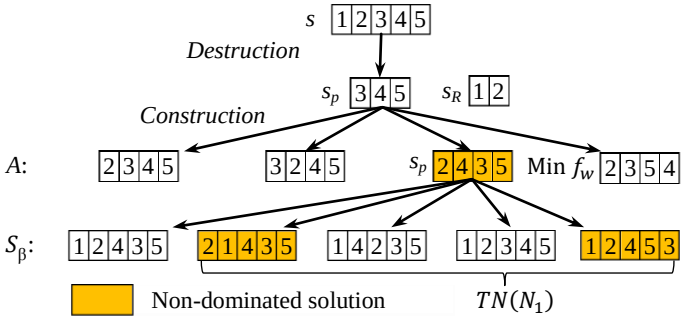

Figure5 Intensification1

Step $i = c$ of Intensification2 is the same, i.e. $(n - c + 1)$ partial solutions are constructed. From these partial solutions, the non-dominated solutions are selected. In the next step, new solutions (of size $n - c + 2$) are obtained by inserting job $s_R(c - 1)$ in first position of the partial non-dominated solutions and using TN(N_1). From the solutions constructed in each step, in the Intensification2 algorithm, the non-dominated solutions are always selected. In the last step 1, the set S_β of complete non-dominated solutions is determined. Fig.6 illustrate the idea of the Intensification2 procedure for the example.

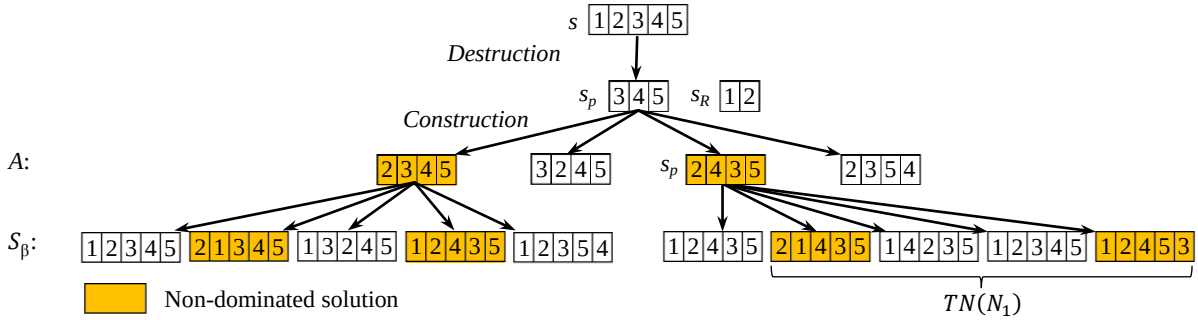

Figure6 Intensification2

6. Computational Experiments

In this paper, to compare the results of algorithms we analyze the efficiency of the proposed intensification procedures used in the MOVNS1 algorithm by Geiger [9], the new modified (MOVMS2, MOVNS3) algorithms and the MOVNS4, MOVNS5 algorithms by Arroyo et al. [12] obtained by the algorithms MOVNSi $(i = 1, \dots, 5)$.

The five algorithms were coded in Matlab R2013a and executed on an Intel Core i7 with a 2.13GHz and 4.0 of RAM. The algorithms were run with the same stopping criterion (Stopping Criterion) based on an amount of CPU time. This time is giving by n seconds (n = number of jobs) and it depends on the size of the considered instance. In this way, we assign more time to larger instances that are obviously more time consuming to solve. Stopping criteria based on CPU times are widely used for performance comparison of heuristic algorithms [21][22].

6.1 Problems Instances

The performance of the algorithms MOVNSi $(i = 1, \dots, 5)$ are compared on 12 problems instances. For the complete enumeration (CE) method and the reference set the sizes of these instances are $n = 5, 6, 7, 8$, and the sizes of more jobs instances are $n = 20, 30, 40, 50, 75$ and 100. The problems were generated randomly, similar to that of Wang and Yen [23] and Ribeiro et al. [24][25]. For each job j, the processing times p_j was uniformly generated in [1,10]. The due date d_j was uniformly generated in $[(1 - TF - RDD/2)TP, (1 - TF + RDD/2)TP]$, where TP is the total processing times of all the jobs, TF is the tardiness factor, and RDD is the relative range of the due dates. TF and RDD have values from $\{0.1, 0.2, 0.3, 0.4\}$ and $\{0.8, 1.0, 1.2\}$, respectively.

Because TF and RDD take 4 and 3 different values, respectively, there are total 12 settings for both parameters.

6.2 Characterizing Non-dominated Solutions by the Five Algorithms

For each problem instance, for the CE method we have the set of all exact efficient solutions of the each instance example of the problem (P). The exact efficient set is denoted by EE which obtained by CE method. We compare the non-dominated solutions obtained by the five algorithms. We denoted by S_1, S_2, S, S_4 and S_5 the sets of non-dominated solutions (approximated Pareto fronts) obtained by the algorithms MOVNSi $(i = 1, \dots, 5)$, respectively. Since for the problem (P) the optimal Pareto front for each instance is not known, a reference set, constituted by gathering all non-dominated solutions obtained by the five tested algorithms, is used. The reference set (the best known Pareto front) is denoted by (Ref). The performance of an algorithm is then measured in terms of the quality of the solutions obtained by this algorithm with respect to the solutions in Ref. In this paper the cardinal measure is used.

Cardinal measure: for each algorithm we compute the number of obtained non-dominated solutions that belong to the EE set, i.e. $|EE \cap S_1|$, $|EE \cap S_2|$, $|EE \cap S_3|$, $|EE \cap S_4|$ and $|EE \cap S_5|$ and that belong to the reference set, i.e. $|Ref \cap S_1|$, $|Ref \cap S_2|$, $|Ref \cap S_3|$, $|Ref \cap S_4|$ and $|Ref \cap S_5|$.

6.3 Comparison of Results

The five algorithms MOVNSi $(i = 1, \dots, 5)$ were run for all the 12 instances for each n of the problem. The sets S_i $(i = 1, .. 5)$, contain the non-dominated solutions found by all the runs. The cardinal measure is calculated for these sets.

Table1 Performance of the algorithms with the complete enumeration (CE) for $5 \leq n \leq 8$

		MOVNS1		MOVNS2		MOVNS3		MOVNS4		MOVNS5	
n	$\|EE\|$	$\|S_1\|$	$\|EE \cap S_1\|$	$\|S_2\|$	$\|EE \cap S_2\|$	$\|S_3\|$	$\|EE \cap S_3\|$	$\|S_4\|$	$\|EE \cap S_4\|$	$\|S_5\|$	$\|EE \cap S_5\|$
5	85	66	51	85	85	85	85	52	48	52	46
6	103	60	45	103	103	103	103	67	60	65	61
7	152	96	87	152	152	152	152	107	101	108	83
8	145	71	56	145	145	144	144	103	93	125	110
Total	485	293	239	485	485	484	484	329	302	350	300
Percentage			49.28%		100%		99.79%		62.27%		61.86%

Table1 presents the comparison among MOVNSi $(i = 1, \dots, 5)$ with the EE (obtained by CE method) regarding the cardinal measure. For each group of 12 instances of size n, Table1 shows the total number of exact efficient solutions $|EE|$, the number of solutions obtained by each algorithms ($|S_1|$,$|S_2|$,$|S_3|$,$|S_4|$ and $|S_5|$) and the total number of exact efficient solutions provided by each algorithm $|EE \cap S_i|, (i = 1, \dots, 5)$. Note that, the algorithms MOVNSi $(i = 1, \dots, 5)$ generate their own set of non-dominated solutions S_i $(i = 1, .. 5)$, which do not necessarily belong to EE. We note that for all groups of instances, the algorithm MOVNS2 determines a greater number of solutions in both sets $|S_2|$ and $|EE \cap S_2|$. For all the 12 instances tested, a total of 485 exact efficient solutions were obtained, from which 239 (49.28%), 485 (100%), 484 (99.79%), 302 (62.27%) and 300 (61.86%) exact efficient solutions were obtained, respectively, by MOVNSi $(i = 1, \dots, 5)$. Based on the cardinal measure, the algorithm MOVNS2 is superior to the algorithms MOVNS1, MOVNS3, MOVNS4 and MOVNS5, $(|EF \cap S_2| > |EF \cap S_3| > |EF \cap S_4| > |EF \cap S_5| > |EF \cap S_1|)$.

Table2 Performance of the algorithms with the reference solutions for $5 \leq n \leq 8$

		MOVNS1		MOVNS2		MOVNS3		MOVNS4		MOVNS5	
n	$\|Ref\|$	$\|S_1\|$	$\|Ref \cap S_1\|$	$\|S_2\|$	$\|Ref \cap S_2\|$	$\|S_3\|$	$\|Ref \cap S_3\|$	$\|S_4\|$	$\|Ref \cap S_4\|$	$\|S_5\|$	$\|Ref \cap S_5\|$
5	85	66	51	85	85	85	85	52	48	52	46
6	103	60	45	103	103	103	103	67	60	65	61
7	152	96	87	152	152	152	152	107	101	108	83
8	145	71	56	145	145	144	144	103	93	125	110
Total	485	293	239	485	485	484	484	329	302	350	300
Percentage			49.28%		100%		99.79%		62.27%		61.86%

It is clear from Table1, the MOVNS2 (S_2) gives the exact number of efficient solution $|EE|$, hence Table2 have the same results as in Table1 since $|S_2| = |EE|$.

Table3 Performance of the algorithms with the reference solutions for $20 \leq n \leq 100$

		MOVNS1		MOVNS2		MOVNS3		MOVNS4		MOVNS5	
n	$\|Ref\|$	$\|S_1\|$	$\|Ref \cap S_1\|$	$\|S_2\|$	$\|Ref \cap S_2\|$	$\|S_3\|$	$\|Ref \cap S_3\|$	$\|S_4\|$	$\|Ref \cap S_4\|$	$\|S_5\|$	$\|Ref \cap S_5\|$
20	835	473	334	737	550	788	532	399	317	485	305
30	1286	640	342	1077	533	1286	394	596	395	772	249
40	1563	845	407	1196	640	1686	576	673	403	976	134
50	1814	1243	356	1359	651	2194	572	981	639	1874	143
75	2000	1276	175	1637	832	3377	574	769	506	2928	142
100	2233	1448	147	1815	973	4128	428	666	487	4166	428
Total	9731	5925	1761	7821	4179	13459	3076	4084	2747	11201	1401
Percentage			18.10%		42.95%		31.61%		28.23%		14.40%

In Table3 with $|Ref|$, which is similar to Table2 with a larger number of jobs n and the same number of instances test, a total of 9731 reference solutions were obtained, from which 1761 (18.10%), 4179 (42.95%), 3076 (31.61%), 2747 (28.23%) and 1401 (14.40%) reference solutions were obtained, respectively, by MOVNSi $(i = 1, \dots, 5)$. Based on the cardinal measure, the algorithm MOVNS2 is superior to the algorithms MOVNS1, MOVNS3, MOVNS4 and MOVNS5, $(|Ref \cap S_2| > |Ref \cap S_3| > |Ref \cap S_4| > |Ref \cap S_1| > |Ref \cap S_5|)$.

Note, from the paper [12] the results for S_4 and S_5 were better than S_1, while the results we have obtained through our research, S_1 was better than S_5 and the reason for this is due to the difference in programming algorithms where we used in our search Stopping Criterion depended on times which equal to number of jobs instead of the number of iterations.

7. Conclusion

In this paper, we considered the MOSMS problem with the minimization the total completion times and the minimization of sum of maximum earliness and tardiness of jobs. Five algorithms based on the MOVNSi $(i = 1, \dots, 5)$ approaches were compared. The results show the MOVNS2 algorithm by using intensification2 procedure improved the efficient solutions for the problem (P) and gives superior results with respect to the others algorithms.

References

[1] J.K., Lenstra, A.H.G.R. Kan and P. Brucker, (1977), *Complexity of Machine Scheduling Problems*, Annals of Discrete Mathematics, 343–362.

[2] K.C. Tan, R. Narasimhan, P.A. Rubin and G.L. Ragatz, (2000), *A Comparison of Four Methods for Minimizing Total Tardiness on A Single Processor with Sequence Dependent Setup Times*, OMEGA, 28, 313–326.

[3] W.J. Chen, (2005), *Minimizing Total Flow Time in the Single-Machine Scheduling Problem with Periodic Maintenance*, Jour. of the Operational Research Society, 57, 410–415.

[4] W-H. Kuo, and D-L. Yang, (2006), Mi*nimizing the Makespan in a Single Machine Scheduling Problem with a Time-Based Learning Effect*, Information Processing Letters, 97, 64–67.

[5] C-J. Liao, and C-Cg. Cheng, (2007), *A Variable Neighborhood Search for Minimizing Single Machine Weighted Earliness and Tardiness with Common Due date*, Comp Ind Eng., 52, 404–413.

[6] G. Wang and B.P. Yen, (2002), *Tabu Search for Single Machine Scheduling with Distinct Due windows and Weighted Earliness/Tardiness Penalties*, Eur. Jour. of Operational Research, 142, 129–146.

[7] D.F. Jones, S.K. Mirrazavi and M. Tamiz, (2002), *Multi-Objective Metaheuristics: An Overview of the Current State-of-Art*, Eur. Jour. of Operational Research, 137, 1–19.

[8] X. Gandibleux and M. Ehrgott, (2005), 1984–2004, 20 *Years of Multiobjective Metaheuristics*, Lecture Notes in Computer Science, 3410, 33–46.

[9] M.J. Geiger, (2004), *Randomized Variable Neighborhood Search for Multi-Objective Optimization*, 4th EU/ME: Design and Evaluation of Advanced Hybrid Meta-Heuristics, 34–42.

[10] Y-C. Liang, H-L.A. Chen, and C-Y. Tien, (2009), *Variable Neighborhood Search for Multi-Objective Parallel Machine Scheduling Problems*, in: Proc. of the 8th International Conference on Information and Management Sciences (IMS 2009), China, 519–522.

[11] Liang, Y-C and M-H. Lo, *Multi-Objective Redundancy Allocation Optimization Using a Variable Neighborhood Search Algorithm*, Jour. of Heuristics, 16, 511–535, (2010).

[12] J.E.C. Arroyo, R.S. Ottoni and A.P. Oliveira, (2011), *Multi-Objective Variable Neighborhood Search Algorithms for a Single Machine Scheduling Problem with Distinct due Windows*, Electronic Notes in Theoretical Computer Science, 281,5–19.

[13] W.E. Smith, (1956), *Various Optimizers for Single-Stage Production*, Navel Research Logistics Quarterly,3, 59–66.

[14] J.R. Jackson, (1955), *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, University of California, Los Angeles.

[15] J.A. Hoogeveen and S.L. van de Velde, (1992), *A New Lower Bound Approach for Single Machine Multicriteria Scheduling*, Operations Research Letters 11,39–44.

[16] J.A. Hoogeveen, (1996), *Single Machine Scheduling to Minimize a Function of Two or Three Maximum Cost Criteria*, Journal of Algorithms, 21,415–433.

[17] P. Hansen and N. Mladenović, (2003), *Variable Neighborhood Search*, Handbook of Metaheuristics, F. Glover and G. A. Kochenberger (Ed.), International Series in Operation Research and Management Science, 57, 145–184.

[18] S.P. Ali and M. Bijari, (2012), *Minimizing Maximum Earliness and Tardiness on a Single Machine Using a Novel Heuristic Approach*, International Conference on Industrial Engineering and Operations Management Istanbul, Turkey, July 3 – 6.

[19] R. Tavakkoli-Moghaddam, G. Moslehi, M. Vasei and A. Azaron, *Optimal scheduling for a single machine to minimize the sum of maximum earliness and tardiness considering idle insert*, Applied Mathematics and Computation 167,1430–1450,(2005).

[20] K.E. Parsopoulos and M.N. Vrahatis, (2002), *Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization*, Natural Computing 1,235–306.

[21] R. Ruiz and T. Stützle, (2007), *A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem*, Eur. Jour. of Operational Research, 177,2033–2049.

[22] R. Ruiz and T. Stützle, (2008), *An Iterated Greedy Heuristic for the Sequence Dependent Setup Times Flowshop Problem with Makespan and Weighted Tardiness Objectives*, Eur. Jour. of Operational Research, 187,1143–1159.

[23] G. Wang and B.P. Yen, (2002), *Tabu Search for Single Machine Scheduling with Distinct Due Windows and Weighted Earliness/Tardiness Penalties*, Eur. Jour. of Operational Research, 142,129–146.

[24] F.F. Ribeiro, S.R. Souza and M.J.F. Souza, (2009), *An Adaptive Genetic Algorithm for Solving the Single Machine Scheduling Problem with Earliness and Tardiness Penalties*, in: Proc. of the IEEE International Conference on Systems, Man, and Cybernetics. San Antonio, USA, 698–703.

[25] F.F. Ribeiro, S.R. Souza, M.J.F. Souza, and R.M. Gomes, (2010), *An Adaptive Genetic Algorithm to Solve the Single Machine Scheduling Problem with Earliness and Tardiness*, in: Proc. of the IEEE Congress on Evolutionary Computation, 1–8.