On the Two Machine Permutation Flowshop Scheduling Problems Using Variable Neighborhood Search Algorithms

T. S. Abdul – Razaq[1], H. M. Motair[2*]

[1]*Dept. of Mathematics, College of Science, University of Al-Mustansiriya*

[2] *Dep. of Mathematics, Open Educational College, Al- Qadisiya, Ministry of Education*

[*]*Corresponding Author: hafedmotair@gmail.com*

Received:7/3/2018, Accepted:23/4/2018, published: 26/04 /2018

DOI: 10.52113/2/05.01.2018/36-47

Abstract : This paper considers the two machine permutation flow shop scheduling problem PFSSP. We study some special cases of the problem $F_2||(C_{max}, \sum C_i, T_{max}, \sum T_i)$ (P). In the first one, the special cases to find the exact non dominated solutions for problem (P) are based on the relation between processing times a_i, b_i and the due dates d_i on machine A and machine B, ($i = 1,2,\dots,n$). In the second one, the special cases are based on the lexicographical order minimization to solve three problems, in each one the primary objective function is maximum completion time (C_{max}), while the secondary objective function is one of the objective functions: total completion times $(\sum C_i)$, maximum tardiness (T_{max}) and total tardiness $(\sum T_i)$, the resulting problems are (P1), (P2), (P3). We propose three algorithms: Variable neighborhood simulated annealing(VNSA), Variable neighborhood descent method(VNDM) and Variable neighborhood with modefied NEH algorithm(VNMNEH). The comparison results show the efficiency of the proposed algorithms. All algorithmes were coded in Matlab programe.

Keywords: Permutation flow shop, non-dominated solutions, lexicographical order, variable neighborhood, simulated annealing, descend method.

1. Introduction

A flow shop consists of n jobs to be processed on m machines in the same machine order. This means that the order in which jobs are processed on different machines for all n jobs is the same and is specified [1]. The machines may be numbered such that each job is processed first on machine 1, then second on machine 2, and so on the machine m last. The scheduling problem in flow shops is to find a sequence of jobs for each machine according to considered performance criterion (criteria). For special case of flow shop scheduling problem (FSSP), it is assumed that the job sequences must be the same on every machine (permutation flow shop scheduling problem (PFSSP)) . In PFSSPs, the number of feasible schedules reduces to n!, while when considered in the general case, it gives $(n!)^m$ possible schedules. Other hypotheses common in scheduling research are the simultaneous availability of all jobs and all machines, deterministic processing times, etc.

Given n jobs to be processed on m machines in the same order; the processing time of job i on machine j being p_{ij}, and d_i is the due date of the job i, $i = 1,2,\dots,n$, $j = 1,2,\dots,m$, it is required to find the order schedule in which these n jobs should be processed on each of the m machines to minimize a given objective function(s).

Various $FSSPs$ have been investigated taking a multiple objectives approach, A (BAB) algorithm [1], for a bi-objective *m*-machine FSSP with makespan and total tardiness objectives. The proposed algorithm found exact NDSs for problems up to 20 jobs. [2] proposed a BAB algorithm for 2-machine PFSSP. The objective is to find the optimal solution for a weighted sum of makespan, mean flow time, and maximum lateness ($F_2||(\lambda C_{max} + \alpha \sum C_i + \beta L_{max})$). A dominance rules and a lower bound are established, also proposed a heuristic algorithm to solve the problem and to find the upper bound for the BAB algorithm. A multi-objective hybrid genetic algorithm with local search for m-machine PFSSPs proposed in [3]. The objective is to minimize simultaneously (1) makespan and maximum tardiness (2) makespan, maximum tardiness and total flow time. The algorithm compared with other multi-objective genetic algorithm (NSGA2 and SPEA) which found has better performance. A mixed integer programming model proposed in [4] Additionally, presented a BAB algorithm and a heuristic algorithm for a *2*-machine flow shop with makespan and flow-time objectives for the lexicographical order minimization of the total completion time with the minimum makespan as the primary objective. Nine heuristics presented in [5] for the *2*-machine FSSP minimizing total flow time subject to makespan. The authors described some polynomialy solvable cases and evaluated the performance of the proposed heuristics for finding approximate solutions. The authors showed that insertion heuristic based on the BAB algorithm proposed by Nagar et al. [6] yielded the best results reasonable computational time A tabu search proposed in [7], for solving the *2*-machine FSSP with makespan as a primary objective and total flow time as the secondary objective. Using a factorial experiment, the authors evaluated the effects of the algorithm's parameters on the performance of the proposed tabu search metaheuristics. Two genetic algorithms based approaches [8] were developed for solving the *2*- machine $FSSP$ to minimize total flow time subject to optimal makespan. The first approach was based on the VEGA (Vector Evaluated Genetic Algorithm). In the second approach, a linear combination of the two objectives (Weighted Criteria Genetic Algorithm) was used. An ant colony optimization algorithm proposed in [9] to study the *2*-machine FSSP for minimize flow time subject to a makespan objective. The algorithm was compared with existing heuristics and yielded much better solutions. Local search heuristics and three metaheuristics: simulated annealing, threshold accepting and tabu search developed in [10] for the *2*-machine flow shop with makespan as the primary objective and total flow time, total weighted flow time, and total weighted tardiness as the secondary objectives. The authors also analyzed the effects of the parameters in these algorithms.

In this paper we consider the two machine PFSSP, where the primary objective is the minimization of the makespan and the secondary objective is the minimizing of the total completion time, or maximum tardiness, or total tardiness. Thus, it is required to find a schedule for which the total completion time or maximum tardiness or total tardiness is minimized, subject to the

constraint that the makespan is minimum. This approach belongs to the so called lexicographical optimization. The need to consider multiple criteria in scheduling is widely known. This optimization yields to compute non dominated solutions (NDSs), also called efficient solutions [11].

This paper is organized as follows: In section 2, the problem definition is introduced. In section 3, we characterize the NDSs for problem (P). The proposed variable neighborhood search algorithms for solving the problem $F_2||Lex(C_{max}, g)$, $g \in \{\sum C_i, T_{max}, \sum T_i\}$ are introduced in section 4. In section 5, the computational experiments are introduced. A comparison of results are discussed in section 6. Finally, the conclusions of the work is given in section 7.

2. Problem Definition

Suppose we have a set of n jobs $N = \{1,2,\dots,n\}$. All jobs are available for processing at time zero and they are to be processed on machines A and B in that order during uninterrupted processed times a_i and b_i respectively, d_i is the due date of a job i. If we consider the four objective functions C_{max}, $\sum C_i$, T_{max}, and $\sum T_i$ and minimize these objective functions simultaneously, the resulting problem (P) can be written: $F_2||(C_{max}, \sum C_i, T_{max}, \sum T_i)$. Hence, the objective is to find the NDSs that minimize simultaneously these objective functions.

For special cases, the minimization of two objective functions according to their importance, which result the lexicographical order minimization of scheduling. The objective then, is to find a schedule that minimizes a secondary objective, subject that the primary objective is minimum. In this paper, we consider the minimization in the lexicographical order of two objective functions such that the primary objective is makespan (C_{max}), and the secondary objective is one of the following objectives: $\sum C_i$, T_{max}, and $\sum T_i$. That is, the objective is to find a schedule that minimize the total completion time ($\sum C_i$), or maximum tardiness(T_{max}), or total tardiness $\sum T_i$) subject to the constraint that maximum completion time C_{max} (makespan) is minimum. This approach which yield three problems to be minimized:

- $F_2||Lex(C_{max}, \sum C_i)$ … (P1)
- $F_2||Lex(C_{max}, T_{max})$ … (P2)
- $F_2||Lex(C_{max}, \sum T_i)$ … (P3)

Johnson's 1954 [12] considered a two-machine n job FSSP. The objective was to find a schedule to minimize the makespan. Johnson proposed an optimizing algorithm and showed that the same permutation of jobs can be used on the two machines (permutation schedule). He also showed that, if there are m machines, an optimal schedule consists of same job sequences on the first two machines and same job sequences on the last two machines.

3. Characterizing the *NDSs* for problem (P).

We propose two results to find a NDSs for the problem (P).

Theorem 3.1: If for each job $i \in N$, $a_i \leq b_i$ and for all job pairs i and $j \in N$, $a_i \leq a_j$ implies $b_i \leq b_j$ and $d_i \leq d_j$, then an efficient schedule is obtained by arranging jobs in non-decreasing order of their processing time on the second machine (SPT2 rule) for the problem (P1)

Proof : Let $\sigma = (1,2,\dots,n)$ be the schedule of jobs ordered in SPT2 of b_i, then σ satisfies the optimality of C_{max} (Johnson's rule) . If for all job pairs i and $j \in N$, $a_i \leq a_j$ implies $b_i \leq b_j$, then σ satisfies the optimality condition for minimum sum of completion times on the second machine

($\sum C_i$). It is clear that if $a_i \leq a_j$ implies $b_i \leq b_j$ and $d_i \leq d_j$ for all job pairs i and j then T_{max} and $\sum T_i$ are minimized .Hence σ gives efficient solution for problem P ■.

Theorem 3.2 : If for each job $i \in N$, $a_i \leq b_i$ and for all job pairs i and $j \in N$ $b_i \leq b_j$ implies $d_i \leq d_j$ and a job l such that $a_l = min_{i\in N}\{a_i\}$, $b_l \geq max_{i\in N}\{a_i\}$. Then NDS for $F_2||(C_{max}, \sum C_i, T_{max}, \sum T_i)$ is obtained by arranging jobs in non-decreasing order of their processing times on the second machine (SPT2 rule),with selected l as the first job, where ties for the first job are broken in favors of a job with smallest processing time on the second machine.

Proof : Let $\sigma = (l, 2,3,..,n)$ be the schedule of jobs ordered in SPT2 of b_i and l satisfied the condition of the first job. It is clear that: $b_1 \geq max_i\{a_i\}$ and $max\{a_i\} \geq a_i$, then

$b_1 \geq a_i, i = 1,2,\dots,n$ (3.1)

Since $a_i \leq b_i \; \forall \; i \in N$, then $\sum_{k=2}^{i-1} a_k \leq \sum_{k=2}^{i-1} b_k$

Hence $a_1 + \sum_{k=2}^{i-1} a_k + a_i \leq a_1 + \sum_{k=2}^{i-1} b_k + b_1$, (since $b_1 \geq a_i \; \forall i$ from (3.1), i = 3, … , n

Then $\sum_{k=1}^{i} a_k = C_i^A \leq a_1 + \sum_{k=1}^{i-1} b_k = C_{i-1}^B$, $i = 2,3,\dots,n$ (3.2)

Since $a_i \leq b_i$, $\forall \; i \in N$ implies that $C_i^A \leq C_{i-1}^B$, $i = 2,\dots,n$

It is well knowing that

$C_i^B = max\{C_{i-1}^B, C_i^A\} + b_i$, $i = 2,3,\dots,n$

From (3.2), we have,

$C_i^B = a_1 + \sum_{k=1}^{i} b_k, i = 1,\dots,n$ (3.3)

Hence $C_{max}^B = C_n^B = a_1 + \sum_{i=1}^{n} b_i$ is minimum for the schedule σ (since $a_1 = min\{a_i\}$).

Now To minimize $\sum C_i^B$

From (3.3), we have $\sum_i^n C_i^B = na_1 + \sum_{i=1}^{n}(n-i+1)\, b_i$ is as small as possible since $a_1 = min_i\{a_i\}$ and $\sum_{i=1}^{n}(n-i+1)b_i$ is minimum when the series are monotonic in opposite senses. Therefore its minimum by SPT2 since the terms $(n-i+1)$ are already decreasing.

To minimize T_{max} and $\sum T_i$ for the schedule σ. It is clear that by SPT2 on second machine, $b_i \leq b_j$ implies $d_i \leq d_j$ for all job pairs i and j, then T_{max} and $\sum T_i$ are minimized. Hence σ gives NDS for $F_2||(C_{max}, \sum C_i, T_{max}, \sum T_i)$ ■.

4. Solving the problem $F_2||Lex(C_{max}, g)$, $g \in \{\sum C_i, T_{max}, \sum T_i\}$.

For the problem $F_2||Lex(C_{max}, \sum C_i)$, some heuristics and one optimal algorithm to solve this problem introduced by Gupta et al. [6]. They study two special cases such that the optimal solution found. These special cases are stated in the following theorems (4.1 and 4.2):

Theorem 4.1: If for each job $i \in N$, $a_i \leq b_i$ and for all job pairs i and $j \in N$, $a_i \leq a_j$ implies $b_i \leq b_j$, Then an optimal schedule can be obtained by sorting the jobs in increasing order of the values b_i ■.

Theorem 4.2: If for each job $i \in N$, $min_{1\leq i\leq n}(a_i) \geq max_{1\leq j\leq n}(b_j)$, then an optimal schedule can be obtained by sorting the jobs in increasing order of the values a_i and by sequencing the job with the smallest b_i in the last position ■.

A heuristic algorithm to find near optimal solution for the above problem proposed by Rajendran in [13]. The algorithm use Johnson rule (JR) to obtain an optimal initial sequence for the objective function C_{max}. Then the permutations of adjacent jobs are used to reduce the vales of the objective function $\sum C_i$. These permutations are chosen according to the two indicators D_i and D_i' which are defined as follows:

$D_{s[r]} = a_{s[r]} + b_{s[r]} - a_{s[r+1]} - b_{s[r+1]}$, where $1 \le r \le n-1$.
$D'_{s[r]} = 2a_{s[r]} + b_{s[r]} - 2a_{s[r+1]} - b_{s[r+1]}$, where $1 \le r \le n-1$.
Where s is the sequence obtained by JR, and $[r]$ is the job position and a_i, b_i are the processing times on machine A and B respectively. The heuristic uses the indicator D_i and ties between the jobs obtained by a permutation are broken using D'_i.

4.1 Solving the problems Pi , i = 1, 2, 3

We use the variable neighborhood search (VNS) combined with simulated annealing algorithm (VNSA) and with descent method (VNDM), also with modefied NEH algorithm VNMNEH to solve each of the problems P_i , $i = 1,2,3$. Four kinds of neighborhoods are used in the variable neighborhood search which are:

1. N_1 the Swap neighborhood.
2. N_2 the Insertion neighborhood.
3. N_3 the Revesion neighborhood.
4. N_4 the k_1k_2 neighborhood proposed by Gupta, et al.[11]. Where the jobs between positions k_1 and k_2 are pairwise interchanged.

We use the following initial solutions:

1. Raj is the initial solution obtained by Raj algorithm.
2. JR is the initial solution obtained by Johnson algorithm.

The first comparison between the considered four algorithms to minimize $\sum C_i$ for the problem P1 which are: Rajendran algorithm (Raj), variable neighborhood search with simulated annealing algorithm as local search and with initial solution obtained by Raj algorithm (VNSA. Raj) , variable neighborhood search with descent method as local search and with initial solution obtained by Raj algorithm (VNDM. Raj) and variable neighborhood search with modefied NEH algorithm (VNMNEH). The second comparison is between the four algorithms to minimize T_{max} for the problem P2 which are: CT. JR, the values of objective functions C_{max} and T_{max} with JR sequence, (VNSA. JR) and (VNDM. JR) with initial solution obtained by JR sequence and variable neighborhood search with modefied NEH algorithm (VNMNEH). The third comparison is between three algorithms to minimize $\sum T_i$ for the problem P3 which are CST. JR, the value of objective functions C_{max} and $\sum T_i$ with JR sequence, (VNSA. JR) and (VNDM. JR) with initial solution obtained by JR sequence and variable neighborhood search with modefied NEH algorithm (VNMNEH).

4.1.1 VNSA Algorithm

1. *Input initial sequence* $Iseq$, $Maxrun$ *(the maximum number of iterations of the whole algorithm),* Ite*(number of iterations for SA alg.*
2. *Evaluate:* $f_1 = C_{max}(Iseq)$, $f_2 = g(Iseq)$.
3. *Set* $r = 1$, $cseq = Iseq$.
4. *Repeat.*
5. *Set* $k = 1$.
6. *Repeat.*
7. $x' = N_k(cseq)$,*choose k neighborhood*
8. *If* $C_{max}(x') > f_1$ *go to 10.*
9. $(cseq, f_1, f_2) = SA(x'_1, Ite)$.
10. $k = k + 1$.
11. *If* $k \le 4$, *go to (7). Otherwise set* $r = r + 1$.
12. *If* $r > Maxrun$ *go to (13). Otherwise go to (4).*
13. *Stop, with* $sequence = cseq$, C_{max} , g.

SA algorithm

1. *Input* x' , Ite.
2. $f_1 = C_{max}(x')$, $f_2 = g(x')$, $t_0 = 10$, $\alpha = 0.95$.
3. *Set* $cseq = x'$, $cf = f_2$.
4. *Set* $i = 1$.
5. $nseq = N_1(cseq)$.
6. $nf_1 = C_{max}(nseq)$, $nf_2 = g(nseq)$.

7. *If* $nf_1 > f_1$, *go to (14)*
8. $\triangle = nf_2 - cf$.
9. $t_i = \alpha \times t_{i-1}$
10. *If* $\triangle \leq 0$, *set* $cseq = nseq$, $f_1 = nf_1$, $cf = nf_2$ *go to (14)*
11. *Else*
12. $pr = e^{(-\triangle/t_i)}$.
13. *If* $pr > rand(0,1)$. $cseq = nseq$, $f_1 = nf_1$, $cf = nf_2$
14. $i = i + 1$
15. *If* $i \leq Ite$ *go to (5)*
16. *Stop with sequence cseq with values of* f_1 , $f_2 = cf$.

4.1.2 VNDM Algorithm

1. *Input initial sequence* $Iseq$, $Maxrun$ *(the maximum number of iterations of the whole algorithm),* Ite*(number of*
2. *iterations for* DM.
3. *Evaluate:* $f_1 = C_{max}(Iseq)$,$f_2 = g(Iseq)$.
4. *Set* $r = 1$, $cseq = Iseq$.
5. *Repeat.*
6. *Set* $k = 1$.
7. *Repeat.*
8. $x' = N_k(cseq)$. *choose k neighborhood*
9. *If* $C_{max}(x') > f_1$ *go to* 10
10. $(cseq, f_1, f_2) = DM(x', Ite)$.
11. $k = k + 1$.
12. *If* $k \leq 4$, *go to (7). Otherwise set* $r = r + 1$.
13. *If* $r > Maxrun$ *go to (13). Otherwise go to (4).*
14. *Stop, with sequence cseq ,with values of* C_{max} *and* g.

DM Algorithm

1. *Input* x' , Ite.
2. $f_1 = C_{max}(x')$, $f_2 = g(x')$
3. *Set* $cseq = x'$, $cf = f_2$.
4. *Set* $i = 1$.
5. $nseq = N_1(cseq)$.
6. $nf_1 = C_{max}(nseq)$, $nf_2 = g(nseq)$.
7. *If* $nf_1 > f_1$, *go to (10)*
8. $\triangle = nf_2 - cf$.
9. *If* $\triangle \leq 0$, *set* $cseq = nseq$, $f_1 = nf_1$, $cf = nf_2$
10. $i = i + 1$
11. *If* $i \leq Ite$ *go to (5)*
12. *Stop with sequence* $cseq$, *with values of* f_1 *and* $f_2 = cf$.

Example 4.1:

The following data are used by Rajendran in [13], to solve the problem P1.

i	1	2	3	4	5
a_i	*15*	*5*	*16*	*5*	*7*
b_i	*19*	*10*	*12*	*20*	*12*

Alg.	Initial Seq.	Best Seq.	Objectives Values	
			$C_{max}(s)$	$\sum C_i(s)$
Raj	JR	(2,5,4,3,1)	78	226
VNSA	Raj	(2,5,1,3,4)	78	224

In the evaluation of the algorithm, the obtained sequence is: $s = (2,5,4,3,1)$, and the best values of objective functions are: $C_{max}(s) = 78$ and $\sum C_i(s) = 226$.
After evaluate the propose VNSA we have the above results.

4.1.3 MMOPE. VN algorithm

To solve the problem $Pi, i = 1,2,3$, we implement the MMOPE. VN algorithm [14]. This algorithm uses the variable neighborhood search and intensification procedure to find set of non dominated solutions for the considered problem. But instead of using the Pareto dominance as a criterion for choosing the required partial or complete schedule(s), we propose the $\boldsymbol{d - Lex. Constructive}$ procedure to find the partial (or complete) schedule that minimize the cost function $g \in \{\sum C_i , T_{max} , \sum T_i\}$ for the problem $F_2||Lex(C_{max}, g)$. To apply the proposed algorithm, we use three initial solutions, the first is the sequence obtained by JR which minimize the cost function C_{max} and give a feasible solution to the

search space, the other two initial solutions obtained by the dispatching rules (DRs) SPT and TLB [15]. We use the four neighborhoods given in 4.1

4.1.4 VNMNEH algorithm

1. *Set* $s^i, i = 1,2,3$ *are initial sequences obtained by* JR, SPT, *and* TLB DRs *and,* $s^i = (j_1, j_2, \dots, j_n)$.
2. $For\ l = 1{:}3\ do$
3. *Set* $s_1^l = (j_1)$, $S_1^l = \{s_1^l\}$ *of partial sequences with one job.*
4. $For\ k = 2\ to\ n\ do$
5. $For\ each\ sequence\ s_j^l \in S_{k-1}^l\ do$
6. $For\ i = 1\ to\ k\ do$
7. *Insert the* kth *job in position* i *of* s_j^l *generating a sequence* s_i^l *with* k *jobs. Evaluate the objective function.*
8. *Construct the set* S_k^l *partial schedules that minimize the cost function* g *of the problem* $P_i, i = 1,2,3$ *such that the* C_{max} *is minimum from the* $k.|S_{k-1}^l|$ *generated partial sequences.*
9. *End of phase* l. *The set* S_n^l *of complete scheduels.*
10. *Form the set* $\cup_{i=1}^{3} S_n^i$ *and construct the set* D *of* $ASNDSs$.
11. $EF = \emptyset, total\ time = 0, D1 = D.$
12. *Repeat*
13. *Choose randomly a solution* s *in* $D1$, *and delete it from* $D1$.
14. *From the neighborhoods* $N_j(s), j = 1,2,3,4$ *choose at random* $ś = N_j(s)$.
15. $S_d = d - Lex.Constructive\ (ś, d)$
16. $EF = EF \cup S_d$
17. *IF* $total\ time > q * n$, *go to step (19).* $q \in (0,1)$
18. $If\ D1 = \emptyset,$ *reset* $D1 (i.e\ D1 = D)$. *Go to (13)*
19. s *is the schedule in* EF *that minimize* $g \in \{\sum C_i, T_{max}, \sum T_i\}$ *of the problem* P_i, *i=1,2,3.*
20. *Stop with the schedule* s.

d − Lex. Constructiv Procedure

1. $Choose\ d\ jobs\ at\ random\ from\ ś, let\ ś1 = \{j_{[1]}, j_{[2]}, \dots, j_{[d]}\}$.
2. $Set\ t = 0,\ S_t = \{ś \setminus ś1\}$ *is the set contain one partial sequence.*
3. $i = 0$
4. $i = i + 1\ ; X = \emptyset.$
5. $If\ i > d,$ *go to (13)*
6. *For each partial sequence* s^* *in* S_t *do*
7. *Insert the job of the position* i *in* $ś1$ *in each position of the sequence* s^*. *The resulting* W *is the set of* $n - d + i$ *partial sequences.*
8. $X = X \cup W.$
9. *End for*
10. $t = t + 1.$
11. *Let* S_t *is the set of partial sequences in* X *that minimize* $g \in \{\sum C_i, T_{max}, \sum T_i\}$ *of the problem Pi , i=1,2,3.*
12. *Go to (4)*
13. *Return the set* S_d *of complete schedules that minimize* $g \in \{\sum C_i, T_{max}, \sum T_i\}$ *of the problem Pi,* $i = 1,2,3.$
14. *Stop.*

5. Computational Experiments:

In this work, to compare the results of the four algorithms and analyze the efficiency of the proposed VNSA, VNDM and VNMNEH algorithms evaluated for each of the study problems P1, P2, and P3. The results are compared with CEM (for $n \leq 10$) which gives the exact solution, and for $n > 10$ the comparisons are among the proposed algorithms only.

All the algorithms were coded in Matlab $R2013a$ and executed on an $intel\ (R)\ Core\ TM\ i7\ CPU$ @ 2.50 GHz, and 8.00 GB of RAM. To evaluate the proposed algorithms, instances with different sizes are considered, the size of these instances are from $n = 4$ to $n = 200$. Processing times a_i and b_i are generated uniformly between 1 and 99 and the due dates of each job is generated from uniform distribution on $[(1 - TF - RDD/2)t, (1 - TF + RDD/2)t]$, where RDD is the "relative range of due

dates", it finds the length of the interval where the due dates are taken. TF the "tardiness factor" find's the relative positions of the center of the interval between 0 and t, $TF = \{0.2\ , 0.4\ , 0.8\ \}$,$RDD = \{0.2\ , 0.6\ , 1.2\ \}$, t is the lower bound of makespan [15] and given by:

$$p = max\{max_{1\le j\le m}\{\sum_{i=1}^{n} p_{ij} + min_i \sum_{l=1}^{j-1} p_{il} + min_i \sum_{l=j+1}^{m} p_{il}\}\ , max_i \sum_{j=1}^{m} p_{ij}\}.$$

We generate nine instances for each n.

6. Comparison of Results:

Tables (1), (3), (5) contain the mean values of the considered objective functions obtained by the four algorithms Raj (or JR), VNSA, VNDM and VNMNEH. Where tables (2), (4), (6) contain the mean values of execution times (in seconds) for the considered objective functions obtained by the four algorithms Raj, VNSA, VNDM and VNMNEH. From these tables we see the following notes:

1. Table (1) contains the mean values of the objective functions, C_{max} and $\sum C_i$ obtained by the four algorithms: Raj, VNSA. Raj, VNDM. Raj and VNMNEH. After evaluate the Raj algorithm and obtained the objective functions values and the schedule that minimize the cost function $\sum C_i$, then we use this schedule as initial solution for the VNSA and VNDM algorithms. Also we evaluate the VNMNEH algorithm by generate three initial solutions which are the initial solutions obtained by JR algorithm and the other initial solutions are obtained by the DRs: SPT and TLB. These mean values show that the three algorithms VNSA , VNDM and VNMNEH reach the optimal values for each values of $n \le 10$ (except for $n = 8$ where VNSA is not) compared with optimal values obtained by (CEM). For $n \ge 12$ the results show that the algorithms VNSA and VNDM are closed together and the two algorithms are better than VNMNEH.
   Also the results show that the three algorithms are better than Raj algorithm.
2. Table (3) contain the mean values of the objective functions, C_{max} and T_{max} obtained by the four algorithms: JR, VNSA. JR, VNDA. JR. and VNMNEH. The JR algorithm is used to minimize the maximum completion time (C_{max}), then the two algorithms VNSA, VNDM are evaluated using JR as initial solution to minimize the cost function T_{max} for the problem P2. Also the VNMNEH evaluated by generate three initial solutions which are the initial solution obtained by JR algorithm and the other initial solutions are obtained by the DRs: SPT and TLB .The results are compared and the mean values show that the three algorithms VNSA, VNDM, and VNMNEH reach the optimal solution for all problem instants size $4 \le n \le 10$ compared with optimal values obtained by CEM. Also the three algorithms have the same values of the cost function T_{max} for $n = 12, 14, 16$. For other problem instants sizes ($n \ge 18$), the results show that the algorithms VNSA and VNDM are closed together with preference to VNSA. Also the two algorithms VNSA, VNDM are better than VNMNEH.
3. Table (5) contains the mean values of the objective functions, C_{max} and $\sum T_i$ obtained by the four algorithms: JR, VNSA. JR, VNDA. JR and VNMNEH. The JR algorithm is used to minimize the maximum completion time (C_{max}), then the two algorithms VNSA and VNDM are evaluated using JR sequence as initial

solution to minimize the cost function $\sum T_i$ for the problem P3. Also the VNMNEH evaluated by generate three initial solutions which are the initial solution obtained by JR algorithm and the other initial solutions are obtained by the DRs: SPT and TLB. The results are compared, the mean values show that the algorithms VNSA, VNDM reach the optimal solution for problem instants of size $n = 4,5,6,7$ while the VNMNEH reach the optimal solution for problem instants of size $n = 4,5,6,7,8$ compared with optimal values obtained by CEM and the VNMNEH algorithm is better than VNSA and VNDM algorithms for $n = 8, 9, 10, 12$ compared with optimal values obtained by CEM. For other problem instants sizes (n), and according to the values of objective function $\sum T_i$ the results show that the two algorithms VNSA and VNDA are better than VNMNEH, and the two algorithms VNSA and VNDM are closed together with preference to VNDM algorithm.

7. Conclusion

This paper considered the two problem : $F_2||(\ C_{max}\ ,\ \sum C_i\ , T_{max}\ ,\ \sum T_i\ )$ (P) and $F_2||Lex(\ C_{max}\ , g)\ ,\ \ g \in \{\sum C_i\ , T_{max}\ , \sum T_i\}$ In problem (P), we show that the NDS can be found for two special cases for problem (P) as shown in theorems (3.1), (3.2). For the second problem which include three special cases of problem (P), we have proposed three algorithms VNSA , VNDM and VNMNEH to minimize the cost function $g \in \{\sum C_i\ , T_{max}\ , \sum T_i\}$ for the problem $F_2||Lex(C_{max}, g)$. The comparison results showed that the VNSA VNDM are closed together and have better performance than VNMNEH for solving the problems P1, P2, P3. We show that for the two algorithms SA and DM, the use of VNS reduce the importance of annealing approach in SA algorithm and making the two algorithms are closed together. Also the use of VNS procedure combined with $d - Lex$ Constructive procedure in VNMNEH algorithm to find the partial (or complete) schedule that minimize the cost function $g \in \{\sum C_i\ , T_{max}\ , \sum T_i\}$ for each sub-problem rather than the non domination criterion gave resonable results compared with the other two algorithms: VNSA and VNDM .

Table (1): The mean values of $Cmax$ and $\sum C_i$ obtained by $Raj\ alg.\ VNSA.Raj, VNDM.Raj$ and $VNMNEH$ algs. for solving ($P1$), compared with CEM for $n \leq 10$.

n	CEM		Raj		$VNSA.Raj$		$VNDM.Raj$		$VNMNEH$	
	$Cmax$	$SumC$	$Cmax$	$SumC$	$Cmax$	$SumC$	$Cmax$	$SumC$	$Cmax$	$SumC$
4	252.56	685.11	252.56	685.11	252.56	685.11	252.56	685.11	252.56	685.11
5	253.22	813.67	253.22	819.44	253.22	813.67	253.22	813.67	253.22	813.67
6	387.89	1456.56	387.89	1467.22	387.89	1456.56	387.89	1456.56	387.89	1456.56
7	416.33	1557.89	416.33	1570.56	416.33	1557.89	416.33	1557.89	416.33	1557.89
8	457	2029.67	457.00	2044.44	457.00	2029.67	457.00	2029.89	457.00	2029.67
9	525.44	2504.56	525.44	2545.89	525.44	2504.56	525.44	2504.56	525.44	2504.56
10	567.89	2912.00	567.89	2951.00	567.89	2912.67	567.89	2912.78	567.89	2912.00
12			624	3748.89	624.00	3692.22	624.00	3692.22	624.00	3690.33
14			806.89	5738.56	806.89	5611.89	806.89	5611.89	806.89	5618.56
16			880.67	6901.00	880.67	6753.11	880.67	6753.33	880.67	6763.56
18			930.33	8205.33	930.33	7942.78	930.33	7938.33	930.33	7958.89
20			1073.56	10390.22	1073.56	10120.22	1073.56	10127.67	1073.56	10174.56
25			1357.67	15815.78	1357.67	15167.56	1357.67	15162.67	1357.67	15241.33
30			1636.44	21817.89	1636.44	20627.89	1636.44	20645.44	1636.44	20791.89

35		1923.56	30599.00	1923.56	28880.11	1923.56	28883.33	1923.56	29242.11
40		2108.44	36171.00	2108.44	34769.22	2108.44	34782.67	2108.44	35065.22
45		2321.78	44728.78	2321.78	42773.00	2321.78	42787.11	2321.78	43068.22
50		2688.56	57531.00	2688.56	55233.56	2688.56	55251.67	2688.56	55777.00
100		7645.89	512392.89	7645.89	443705.22	7645.89	443854.78	7645.89	465443.11
150		7645.89	512392.89	7645.89	443705.22	7645.89	443854.78	7645.89	465443.11

Table (2): The mean values of execution times for Raj, VNSA. Raj, VNSDM. Raj and VNMNEH algs and CEM for solving (P1).

n	Raj	VNSA. Raj	VNDM. Raj	VNMNEH	n	Raj	VNSA. Raj	VNDM. Raj	VNMNEH
4	0.005	28.592	28.391	5.804	18	0.00	31.41	31.17	26.77
5	0.002	28.581	28.426	7.435	20	0.01	31.61	31.46	29.50
6	0.002	28.850	28.666	9.227	25	0.01	32.56	32.27	35.99
7	0.002	29.368	29.142	10.847	30	0.01	32.86	32.67	42.88
8	0.000	29.130	29.031	12.675	35	0.03	33.61	33.67	50.56
9	0.002	30.843	30.727	14.340	40	0.03	34.33	34.16	56.96
10	0.002	30.738	30.501	15.840	45	0.05	34.11	33.89	64.35
12	0.00	31.11	30.96	18.32	50	0.07	34.41	34.18	73.21
14	0.00	31.20	31.00	20.38	100	0.40	39.24	39.01	148.18
16	0.00	31.31	31.09	23.68	150	1.58	44.17	43.76	232.51

Table (3): The mean values of Cmax , T_{max} obtained JR VNSA. JR, VNDM. JR and VNMNEH algs for solving (p2), compared with CEM for $n \leq 10$.

n	CEM		CT. JR		VNSA. JR		VNDM. JR		VNMNEH	
	$Cmax$	T_{max}	$Cmax$	T_{max}	$Cmax$	T_{max}	$Cmax$	T_{max}	$Cmax$	T_{max}
4	252.56	137.78	252.56	137.78	252.56	137.78	252.56	137.78	252.56	137.78
5	253.22	149.89	253.22	164.00	253.22	149.89	253.22	149.89	253.22	149.89
6	387.89	246.22	387.89	276.78	387.89	246.22	387.89	246.22	387.89	246.22
7	416.33	261.00	416.33	280.22	416.33	261.00	416.33	261.00	416.33	261.00
8	457.00	268.44	457.00	345.22	457.00	268.44	457.00	268.44	457.00	268.44
9	525.44	285.89	525.44	368.22	525.44	285.89	525.44	285.89	525.44	285.89
10	567.89	292.67	567.89	435.00	567.89	292.67	567.89	292.67	567.89	292.67
12			624.00	497.00	624.00	429.67	624.00	429.67	624.00	429.67
14			806.89	650.44	806.89	427.00	806.89	427.00	806.89	427.00
16			880.67	744.67	880.67	578.78	880.67	578.78	880.67	578.78
18			930.33	771.67	930.33	510.00	930.33	510.00	930.33	511.33
20			1073.56	947.11	1073.56	648.78	1073.56	648.78	1073.56	658.00
25			1357.67	1149.56	1357.67	818.00	1357.67	818.00	1357.67	849.00
30			1636.44	1397.00	1636.44	925.89	1636.44	925.89	1636.44	1000.89
35			1923.56	1699.44	1923.56	1083.89	1923.56	1083.89	1923.56	1159.67
40			2108.44	1967.11	2108.44	870.22	2108.44	870.22	2108.44	1011.44
45			2321.78	2110.44	2321.78	1143.56	2321.78	1143.56	2321.78	1263.78
50			2688.56	2409.33	2688.56	1223.67	2688.56	1223.00	2688.56	1388.56
100			5174.44	4974.22	5174.44	1907.89	5174.44	1908.33	5174.44	2813.56
150			7645.89	7146.56	7645.89	3925.22	7645.89	3925.22	7645.89	5731.89

Table (4): The mean values of execution times for the VNSA. JR, VNDM. JR and VNMNEH algs. for solving (P2).

n	T. VNSA. JR	T. VNDM. JR	VNMNEH	n	T. VNSA. JR	T. VNDM. JR	VNMNEH
4	57.66	57.71	5.19	18	64.81	64.67	18.26
5	57.91	57.96	6.77	20	64.81	64.92	20.45
6	58.27	58.23	7.19	25	66.97	66.68	24.75
7	58.93	58.63	7.87	30	69.68	69.86	28.01
8	58.92	59.01	9.22	35	71.44	71.30	32.76

9	62.15	62.12	10.23	40	69.24	69.12	38.33
10	62.42	62.31	11.32	45	70.79	70.73	42.68
12	63.47	63.51	12.54	50	81.44	81.57	48.40
14	63.98	63.97	14.76	100	82.98	82.95	90.21
16	64.43	64.36	16.29	150	101.18	100.98	141.88

Table (5) The mean values of Cmax , $\sum T_i$, obtained JR, VNSA. JR, VNDM. JR and VNMNEH algs. for solving (P3), compared with CEM for $n \leq 10$.

n	CEM		CST. JR		VNSA. JR		VNDM. JR		VNMNEH	
	Cmax	*SumT*	*Cmax*	*SumT*	*Cmax*	*SumT*	*Cmax*	*SumT*	*Cmax*	*SumT*
4	252.56	241.67	252.56	252.56	252.56	241.67	252.56	241.67	252.56	241.67
5	253.22	317.89	253.22	356.44	253.22	317.89	253.22	317.89	253.22	317.89
6	387.89	612.33	387.89	707.89	387.89	612.33	387.89	612.33	387.89	612.33
7	416.33	581.78	416.33	753.78	416.33	581.78	416.33	581.78	416.33	581.78
8	457.00	829.56	457.00	1122.89	457.00	831.33	457.00	829.56	457.00	829.56
9	525.44	969.67	525.44	1412.78	525.44	976.11	525.44	969.67	525.44	969.67
10	567.89	980.89	567.89	1589.33	567.89	985.67	567.89	982.56	567.89	983.78
12			624.00	2414.56	624.00	1560.22	624.00	1553.67	624.00	1565.22
14			806.89	3332.11	806.89	2135.89	806.89	2134.67	806.89	2152.44
16			880.67	4236.44	880.67	2510.11	880.67	2510.22	880.67	2584.78
18			930.33	5120.78	930.33	2849.00	930.33	2851.78	930.33	2998.78
20			1073.56	6817.22	1073.56	3797.56	1073.56	3825.00	1073.56	4105.89
25			1215.67	4836.22	1215.67	813.56	1215.67	800.33	1215.67	1238.89
30			1536.67	8353.00	1536.67	617.00	1536.67	610.78	1536.67	1323.11
35			1835.00	11536.67	1835.00	1386.89	1835.00	1365.56	1835.00	3449.78
40			2179.22	15640.00	2179.22	1413.11	2179.22	1387.00	2179.22	3880.22
45			2304.67	19951.11	2304.67	1313.56	2304.67	1301.33	2304.67	3886.33
50			2676.67	22867.33	2676.67	2178.11	2676.67	2187.00	2676.67	5608.56
100			5174.44	165217.44	5174.44	59154.78	5174.44	59224.56	5174.44	82792.33
150			7645.89	358107.22	7645.89	127768.11	7645.89	127812.22	7645.89	191343.67

Table (6): The mean values execution times for the VNSA. JR, VNDM. JR and VNMNEH algs. for solving problem (P3).

n	VNSA. JR	VNDM. JR	VNMNEH	n	VNSA. JR	VNDM. JR	VNMNEH
4	58.37	58.15	5.73	18	71.84	71.26	26.41
5	58.48	58.46	7.35	20	72.66	72.01	29.14
6	58.92	58.98	9.24	25	73.88	73.54	33.25
7	59.32	59.38	10.86	30	73.45	73.37	39.61
8	59.59	59.56	12.58	35	78.46	77.62	47.40
9	62.95	62.89	14.34	40	78.15	77.58	52.78
10	63.13	63.06	15.99	45	78.17	77.64	61.29
12	70.47	69.79	18.47	50	90.34	89.47	66.62
14	70.65	70.23	21.03	100	94.10	94.09	147.36
16	71.06	70.48	23.47	150	105.36	104.87	231.42

8. Reference

[1] Lemesre J., Dhaenens C., Talbi E. G., (2007),"An exact parallel method for a bi-objective permutation flow shop problem". Eur J Oper Res, 177. 1641 –1655.

[2] Allahverdi, Ali. (2001),"The tri-criteria two-machine flow shop

scheduling problem" International Transactions in Operational Research, vol. 8, no 1, 403-425.

[3] Ishibuchi H., T. Yoshida, T. Murata, (2003)," Balance between genetic search and local search in memetic algorithms for multi-objective permutation flow shop scheduling", Evol Comput, 7 (2), 204–223.

[4] T'kindt V., Gupta J.N.D., Billaut J.-C., (2003), "Two-machine flow shop scheduling with a secondary criterion", Comput Oper Res, 30 (4) 505–526.

[5] Gupta J N.D, Neppalli J, R, Werner V, F., (2001), "Minimizing total flow time in a two machine flow shop problem with minimum makespan". International Journal of Production Economics Volume 69, Issue 3, pp. 323-338.

[6] Nagar A, Heragu S.S., Haddock J., (1995), "A branch and bound approach for two-machine flow shop scheduling problem", J Oper Res Soc, 46. 721–734.

[7] Gupta J.N.D., Palanimuthy N., Chen C.L., (1999), "Designing a tabu search algorithm for the two-stage flow shop problem with secondary criterion". Prod Plan Control, 10, 251– 265.

[8] Neppalli V.R., Chen C.L., Gupta J.N.D., (1996), "Genetic algorithms for the two stage bi-criteria flow shop problem", Eur J Oper Res, 95 (2), 356–373.

[9] T'kindt V., Monmarche N., Tercinet F., Laugt D., (2002), "An ant colony optimization algorithm to solve a 2-machine bi-criteria flow shop scheduling problem", Eur J Oper Res, 142,250–257.

[10] Gupta J.N.D., Hennig K., Werner F., (2002), "Local search heuristics for two- stage flow shop problems with secondary criterion", Comput Oper Res, 29 (2),123–149.

[11] Framinan J.M., Leisten R., (2006), "A heuristic for scheduling a Permutation flow shop with makespan Objective subject to maximum tardiness", Int J Prod Econ, 99 .28–40.

[12] Johnson S.M., (1954), "Optimal Two and three-stage production Schedules with setup times included". Nav Res Logist Q, 1 (1), 61–68.

[13] Rajendran.,C.,(1992),"Two-stage flow shop scheduling problem with bi-criteria ". J Oper Res Soc, 43, (9) 871–884.

[14] Abdul-Razaq, T.S., Motair, M. H, (2017)," Multi-objective scheduling problems on the two machine permutation flow shop using metaheuristics". Unpublished work to appear in J. of AL-Nahrain.

[15] Arroyo J.C., Armentano V.A., (2004),"A partial enumeration heuristic for multi-objective flow shop scheduling problems", J Oper Res Soc, 55, 1000 –1007.