Exact and Hybrid Metaheuristic Algorithms to Solve Bi-Objective Permutation Flow Shop Scheduling Problem

Hafed M. Motair
Distinguished Secondary School, Iraq, Diwaniya, Ministry of Education.

*Author for Correspondence, E-mail: hafedmotair@gmail.com

Abstract. In this paper, we consider 2-machine permutation flow shop scheduling problem with bi-objective of minimizing the makespan \( C_{\text{max}} \) and maximum tardiness \( T_{\text{max}} \). The aim is to minimize the two objective functions lexicographically, which is minimize maximum tardiness subject to that the makespan is optimal. We introduce a Branch and bound algorithm with two dominance conditions to find the optimal solution of the problem for job sequence with up to 14 jobs. For moderate and large sized problems we apply a simulated annealing algorithm to develop a metaheuristic algorithm based on the first level branch and bound and variable neighborhood search. To compare with the original algorithm, numerical experiments provided, the results shows the efficiency of the dominance conditions also the efficiency of proposed modification of the original metaheuristic algorithm.

Key words: Permutation flow shop scheduling, simulated annealing, Branch and bound, Makespan, Maximum Tardiness, variable neighborhood search.

1. Introduction

In a flow shop scheduling, a problem consists of \( n \) jobs to be processed on \( m \) machines in the same order of machine. This means that the order in which jobs processed on different machines for every sequence is the same and specified and the number of possible sequences are\((n!)^m\). The objective in flow shop scheduling problem is to find a schedule for each machine according to the considered objective functions. For special case, if the job sequences assumed the same for all machines, then the resulting called permutation flow shop scheduling (PrFS). Hence, the number of feasible schedules reduces to \( n! \). The common hypotheses in scheduling problems research [1] are deterministic processing times, the simultaneous availability of all jobs and all machines, etc.

In multiobjective scheduling problems, there are generally three main classes. In the first class, tradeoffs between objective functions allowed, an aggregation function of the considered objective functions minimized, and the resulting is one solution. In the second class, the objective functions ordered with respect to their importance (lexicographical order) such that the first objective minimized, then the second minimized with respect to the best value of the first objective and so on. The resulting is a unique solution.
In the third class, the objectives minimized simultaneously and a set of nondominated solutions obtained. In this paper, we consider the bi-objective 2-machine PrFS problem of minimizing the makespan and maximum tardiness. The aim is to minimize the two objective functions lexicographically such that minimize maximum tardiness subject to that the makespan is optimal. It is well known that the second scheduling objective is to desires customers by regard to the due dates, while the first objective is to make the company advantage from the optimization of the machine occupation. Since minimization of makespan problem for m=2 is polynomially solvable [2], and maximum tardiness minimization is Np-hard [3]. Therefore, our problem is Np-hard also.

To solve the problem tackled in this paper, we introduce a Branch and bound (BAB) algorithm and hybrid metaheuristic algorithm. BAB method depends on efficient estimation of two bounds, the lower bound and upper bound of branches (subsets) of the search space. If these two bounds are not available, the algorithm decline to a fully search, the disadvantage of the BAB method and other exact methods is the requestments of too much computation times for large and sometimes moderate problems.

To reduce the search space and the number of visited search, dominance conditions proposed. Several BAB algorithms for bi-objective PrFS problems in literature. [4] proposed a BAB algorithm for PrFS problem to minimize makespan and maximum tardiness simultaneously. [5] proposed a BAB algorithm to minimize total flowtime subject to makespan is optimal. [6] proposed a BAB algorithm to minimize two objective functions total flowtime and total tardiness, also proposed a several dominance properties. [7], proposed BAB algorithm and a heuristic to solve a bi-objective PrFS problem to minimize total flowtime, subject to the constraint that the makespan is optimal. [7] proposed a BAB algorithm to minimize total flowtime subject to makespan is optimal.[8], proposed BAB algorithm to minimize makespan and maximum earliness simultaneously. [9] proposed a BAB algorithm to solve a weighted sum of total completion time and makespan for 2-machine PrFS problem.[10] proposed a BAB algorithm to minimize the sum of maximum tardiness and earliness for PrFS problem.

As we explained above, exact methods cannot used for large and some times for moderate sized problems, so researchers have focused their attention in recent years to the use of approximation methods such as metaheuristics (simulated annealing, Tabu search, Variable neighborhood search, etc.). Moreover, the researchers developed the so-called hybrid metaheuristics, where the original algorithm combined with other algorithm in order to get the benefit of the two algorithms. Unlike exact methods, metaheuristics allow solving large-size problems by finding reasonable solutions in a reasonable time [11].

In [12] Suresh and Mohanasundaram developed a Pareto archived simulated annealing for solving PrFS problem to minimize makespan and total flowtime simultaneously. [13] introduced a simulated annealing algorithm for 2-machine PrFS to minimize weighted sum of total flowtime and total tardiness. Khan and Govindan in [14] introduced a multiobjective simulated annealing for m machine PrFS problem to minimize makespan and maximum tardiness. Several bi-objective PrFS problems using other metaheuristics such as tabu search in [7] and [15]. Iterated Local Search in [16], [17], [18].

Hybride metaheuristics in [19], [20], [21] and [22]. Recently, Lu C et al. [23] proposed a hybrid multi-objective backtracking search algorithm to solve a bi-objective PrFS problem by considering makespan and energy consumption. Deng and Wang [24] proposed a competitive memetic algorithm to solve a bi-objective PrFS problem considering makespan and total tardiness. Li and Ma [25] introduced multiobjective discrete artificial Bee colony algorithm to solve sequence dependent setup times PrFS problem to minimize makespan and total flowtime. Fang and Billaut [26] developed evolutionary simulated annealing algorithm for mixed-model multi-robotic disassembly line balancing with interval processing time to minimize the cycle time, peak workstation energy consumption, and total energy consumption.
The remainder of this paper is completed as follows, we describe the proposed algorithms in section 2. In section 3, we explain the proposed algorithms. The experimental results are discussed in section 4. The conclusions are discussed in section 5.

2. Problem Formulation

Suppose we have a set of n jobs \( N = \{J_1, J_2, ..., J_n\} \) that are available to be processing at time zero and to be processing on machine \( A_1 \) and \( A_2 \) in that order during uninterrupted processed times \( a_i \) and \( b_i \) respectively, \( t_i \) is the due date of a job \( i \). In this paper, we consider the two objective function makespan \( (C_{max}) \) and maximum tardiness \( (T_{max}) \).

If the presented solution is \( \pi = \{J_1, J_2, ..., J_n\} \).

Then the completion time of the job \( J_k \) on machin \( j \) denoted \( C_{j_{k,j}} \) and recursively computed as follows:
\[
C_{j_{k,j}} = \max\{C_{j_{k,j-1}}, C_{j_{k-1,j}}\} + t_{j_{k,j}}, \quad k = 1, ..., n \quad j = 1, 2, \quad \text{where,} \quad C_{j_{0,j}} = 0, k = 1, ..., n.
\]

The makespan is \( C_{max}(\pi) = C_{j_{n,2}} \).

The tardiness of the job \( J_k \) is:
\[
T_{j_k} = \max\{C_{j_{k,2}} - t_{j_{k,0}}, 0\}
\]

The maximum tardiness is:
\[
T_{max}(\pi) = \max_k (T_{j_k})
\]

The objective is to minimize \( T_{max} \) subject to \( C_{max} \) is optimal, if we extend the standard notation of the scheduling problems to our bi-objective PrFS problem, then, the problem can be written as:

\[
F_2[prmu|\text{Lex}(C_{max}, T_{max})] \tag{29}
\]

denoted \( (P) \), where the notation \( \text{Lex}(f_1, f_2) \) means that minimize the objective function \( f_2 \) subject to the objective function \( f_1 \) is optimal.

The problem \( (P) \) is special case of the problem \( F_2[prmu|\#(C_{max}, T_{max})] \), which mean that the objective is to minimize the two objective functions simultaneously.

The bi-objective PrFS problem \( (P) \) is defined as:
\[
\begin{align*}
\text{Min } & \quad T_{max} \\
\text{s.t. } & \quad C_{max} = C_{max}(JOH) \\
& \quad \text{(Is optimal)}
\end{align*}
\]

where \( JOH \) is a permutation of jobs sequence (solution) obtained by Johnson algorithm [2].

Several approaches proposed in the literature [27] including the two objective functions makespan and maximum tardiness, such as lexicographical scheduling approaches, \( \varepsilon \)—constraint approaches, a priori approaches and a posteriori (Pareto) approaches.

3. Proposed Algorithms

3.1 Proposed BAB algorithm

We propose a BAB algorithm to find the optimal solution for the problem \( (P) \). Every node include a sequence \( \sigma \) of \( r \) scheduled jobs, a sequence \( \tau \) of \( n-r \) unscheduled jobs and two lower bounds: the lower bound of the objective function \( C_{max} \) and the lower bound of the objective function \( T_{max} \). The lower bound of the objective \( C_{max} (LBCmax) \) is calculated by sorting the jobs in \( \tau \) on the two machines according to Johnson algorithm (i.e \( JOH \) sequence), and then concatenate the obtained sequence at the end of \( \sigma \). The lower bound \( LBCmax \) is the value of \( C_{max} \) of the resulting sequence. To calculate the lower bound of objective \( T_{max} (LBTmax) \), we use the bound of Daniels and Chambers [4]:

\[
LBT_{max} = \max (LB1T_{max}, LB2T_{max})
\]

1. Calculate \( LB1T_{max} \):
   - Sorting the jobs in \( \tau \) on the two machines according to Johnson algorithm.
   - Concatenate the obtained sequence at the end of \( \sigma \), the resulting is sequence \( \mu \), then
   - \( LB1T_{max} = \max_{i \in \sigma} (0, C_i(\mu) - t_i) \)
2. Calculate $LB2T_{\text{max}}$:

$LB2T_{\text{max}} = \max \{ 0; \min_{j \in \tau} (a_i) + \max_{i=1,2,\ldots,|L|} (\sum_{j=1}^{i} b_{l(j)} - t_{L(j)}) \}$, where $L$ the list obtained by sorting $\tau$ in EDD rule.

The upper bound for the objective makespan ($UBC_{\text{max}}$) and for the objective maximum tardiness ($UBT_{\text{max}}$) calculated using the JOH sequence. To reduce the search space and the number of visited nodes, we use the following dominance conditions [4]:

**Theorem (1):** If $b_j \leq \min (a_j, b_j)$ and $d_i \leq d_j$, then a sequence $\pi$ where $J_j$ preced $J_i$ does not exist.

**Theorem (2):** If $a_i \leq \min (a_j, b_j)$ and $d_i \leq d_j$, then a sequence $\pi$ where $J_j$ preced $J_i$ immediately does not exist.

The resulting are two algorithms, BAB algorithm (without dominance conditions) and BAB-Dc algorithm (with dominance conditions). BAB algorithm summarized in the following steps:

**BAB algorithm:**

**Step1:** Let $N$ be the set of jobs to be schedule:

- $SEQ = JOH$
- Let $UBC_{\text{max}} = C_{\text{max}}(SEQ)$, and $UBT_{\text{max}} = T_{\max}(SEQ)$.
- Create the root node $\sigma_0 = \emptyset$, $\tau_0 = N$, $Q = \{ s_0 \}$, $S_0 = \sigma_0 / \tau_0$.

**Step2:**

- While $Q \neq \emptyset$ do
- Select in $Q$ the node $s_i$ with lowest value of $LBT_{\text{max}}$.
- $Q = Q \setminus \{ s_i \}$, $\tau = \tau_i$.
- For $M=1$ to $|\sigma_i|$ do
  - Select a job $J_j$ in $\sigma$: $\sigma = \sigma \setminus \{ J_j \}$.
  - Create a child node $s_{i+1}^{(M)} = \sigma_i \setminus \{ J_j \}$, $\tau_{i+1}^{(M)} = \tau_i \setminus \{ J_j \}$.
  - Compute $LBC_{\text{max}}(s_{i+1}^{(M)})$, $LBT_{\text{max}}(s_{i+1}^{(M)})$.
  - IF $LBC_{\text{max}}(s_{i+1}^{(M)}) \leq UBC_{\text{max}}$ and $LBT_{\text{max}}(s_{i+1}^{(M)}) < UBT_{\text{max}}$
    - Then if $\tau_{i+1}^{(M)} \neq \emptyset$, then $Q = Q \cup \{ s_{i+1}^{(M)} \}$.
    - $\sigma_i = \sigma_i^{(M)}$, $UBT_{\text{max}} = T_{\max}(SEQ)$.
  - End IF
- End For
- End While

**Step3:** Return $UBC_{\text{max}}$, $UBT_{\text{max}}$.

3.2 Proposed Hybrid Metaheuristic

In [28], we propose a hybrid metaheuristic (VNSA) based on the simulated annealing (SA) and variable neighborhood search (VNS). The VNSA described in the following steps:

**VNSA Algorithm**

1. Input IS (initial sequence), MR (the maximum number of iterations of the algorithm), IT(number of iterations for SA algorithm).
2. Calculate: $f_1 = C_{\text{max}}(IS)$, $f_2 = T_{\text{max}}(IS)$.
3. Put, $d = 1$, CS = IS.
4. Repeat
5. Put, $M = 1$.
6. Repeat.
7. $x' = N_M(CS)$, choose M neighborhood.
8. If $C_{\text{max}}(s') > f_1$ go to 10.
9. \((CS, f_1, f_2) = SA(s'_1, IT)\).
10. \(M = M + 1\).
11. If \(M \leq 4\), go to (7). Otherwise set \(d = d + 1\).
12. If \(d > MR\) go to (13). Otherwise go to (4).
13. Stop, return \(C_{\text{max}}\) and \(T_{\text{max}}\).

**SA Algorithm:**
1. Input \(s'\), IT.
2. \(f_1 = C_{\text{max}}(x'), f_2 = T_{\text{max}}(x'), T_{m0} = 10, \alpha = 0.95\).
3. Set \(CS = s', cf = f_2\).
4. Set \(z = 1\).
5. \(NS = N_1(CS)\).
6. \(nf_1 = C_{\text{max}}(NS), nf_2 = T_{\text{max}}(NS)\).
7. If \(nf_1 > f_1\), go to (14)
8. \(\Delta = nf_2 - cf\).
9. \(T_{mz} = \alpha \times T_{mz-1}\).
10. If \(\Delta \leq 0\), set \(CS = NS, f_1 = nf_1, cf = nf_2\) go to (14)
11. Else
12. \(PR = e^{(\Delta/T_m)}\).
13. If \(PR > \text{rand}(0,1)\). \(CS = NS, f_1 = nf_1, cf = nf_2\)
14. \(z = z + 1\)
15. If \(z \leq IT\) go to (5)
16. Stop with sequence \(CS\) with values of \(f_1\) and \(f_2 = cf\).

We improve the performance of VNSA algorithm by combining the BAB algorithm, the resulting algorithm denoted VNSA-BAB. The idea is to use the first level of proposed BAB algorithm to construct a set of initial solutions, the first job in each of these initial solutions come from the first level of BAB algorithm and the remaining jobs generated using Johnson's algorithm. The proposed VNSA-BAB algorithm has the same steps of the original VNSA algorithm except the first step that described as follows:

1. Generate a set of initial solutions, each of them obtained by first level of BAB algorithm and Johnson's algorithm. \(MR\) (the maximum number of iterations of the algorithm), \(IT\) (number of iterations for SA algorithm).

The rest of steps is the same of remaining VNSA algorithm steps. The advantage of using the first level of BAB algorithm to generate the initial solutions, so we reduce the search space because we don’t need to explore the job in the first position.

The neighborhoods \(N_i\) of the VNS procedure and SA algorithm are:

1. Revesion Neighborhood.
2. Swap Neighborhood.
3. Insertion Neighborhood.
4. \(k_1k_2\) Neighborhood [29].

For the proposed VNSA-BAB algorithm, these neighborhoods are implemented such that the permutation made for the jobs from \(J_2\) to \(J_n\).
3.3 Illustration Example:

Consider the following 20-job, 2-machine instance:

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |   |   |
| J_1 | a_i | b | d_t | J_15 | a_i | b | d_t |
| J_2 | 15  | 92 | 433 | J_16 | 21 | 94 | 41 |
| J_3 | 13  | 85 | 132 | J_17 | 23 | 75 | 412 |
| J_4 | 73  | 19 | 211 | J_18 | 52 | 21 | 28 |
| J_5 | 83  | 60 | 175 | J_19 | 15 | 86 | 839 |
| J_6 | 89  | 26 | 77 | J_20 | 10 | 9 | 78 |
| J_7 | 56  | 15 | 603 | J_21 | 88 | 99 | 229 |
| J_8 | 33  | 8 | 83 | J_22 | 61 | 48 | 791 |
| J_9 | 68  | 64 | 48 | J_23 | 81 | 90 | 388 |
| J_{10} | 22 | 1 | 84 | J_{20} | 68 | 5 | 229 |

To illustrate the proposed algorithms for the problem (P):

For VNSA algorithm, the initial solution obtained by Johnson algorithm is: IS=(2,1,14,11,12,7,19,16,9,14,17,18,5,13,3,6,15,8,20,10), which gives the values of objective functions: C_{max}=1040, T_{max}=956.

For VNSA-BAB algorithm, the first job obtained by first level BAB is 2, the remaining obtained by Johnson algorithm and the final initial solution is: IS=(2,1,14,11,12,7,19,16,9,14,17,18,5,13,3,6,15,8,20,10), which gives the values of objective functions: C_{max}=1040, T_{max}=956. The first job is fixed and the permutation made for the jobs from J_2 to J_n in the neighborhood generation. After run the two algorithms the values of objective functions: C_{max}=1040, T_{max}=742 for VNSA algorithm and C_{max}=1040, T_{max}=707 for VNSA-BAB.

4. Experimental Results

4.1 Parameter Setting

We compare the two algorithms, BAB algorithm and BAB-Dc algorithm, the results are in tables (1) and (2), also we compare three algorithms, pure SA (PUSA) algorithm, VNSA algorithm, and VNSA-BAB algorithm, the results are in table (3).

After several experiments, we set the number of iterations for SA algorithm (IT) for both VNSA algorithm and VNSA-BAB algorithm to 10000 iterations, while the number of runs (MR) of each algorithm implemented such that it is not less than 5 runs. For SA algorithm, the initial temperature (t_0) set to 10, The probability of accepting a nonimproving neighbor is PR = e^{-\Delta/Tm}, where \Delta is the change of the objective function and it is proportional to the temperature Tm. The generated instances consists of small problem instances with n \in \{4,5,6,7,8,9,10,11,12,13,14,15\} and m = 2, where the optimal solution can be found by BAB algorithm for most of these problems.

4.2 Generation of Instances

For the case of big sized problem instances, the generated instances with n \in \{20,30,40,50,60,70,80,90,100,120,130,140,150,175,200,250,300\} and m = 2. The processing times sampled from discrete uniform distributions on [1,99], and the due dates of each job is generated on [(1 - TF - RDD/2)P,(1 - TF + RDD/2)P] from uniform distribution where RDD and TF are hardiness factors of the problem and taken from the sets \{0.2,0.6,1.2\} and \{0.2,0.4,0.8\} respectively, P is value of makespan.
The machine is LENOVO Intel (R) Core TM i7 CPU @ 2.50 GHz, and 8 GB of RAM.

4.3 Comparison of Results

Table (1) showed the results of the BAB algorithms: BAB algorithm and BAB-Dc algorithm also the number of visited nodes of the two algorithms and the values of objective functions obtained by complete enumeration method (CEM). According to the results as shown in table (1), the dominance conditions reduce the number of visited nodes in the search space and the computational times, for example, when n=10 the mean value of the number of visited nodes obtained by BAB-Dr algorithm are 4485.44 nodes which is about 12% of that obtained by BAB algorithm (36628.00 nodes). This results explained in table (2). In table (2) we put the computational times and the number of solved and unsolved problems for 1800 seconds of CPU times for the two algorithms. In table (3), we put the mean values obtained by three algorithms, PUSA, VNSA, and VNSA-BAB. The three algorithms generate approximately the same results for small size instants (from n=4 to n=11), for moderate and large size instants,

Table 1. Computational results of proposed BAB algorithms compared with CEM Method.

| n  | CEM | BAB | BAB-Dr |
|----|-----|-----|--------|
|    | M.Cmax | M.Lmax | M.Cmax | M.Lmax | M.node | M.Cmax | M.Lmax | M.node |
| 4  | 219.56  | 154.00  | 219.56  | 154.00  | 5.33    | 219.56  | 154.00  | 3.77    |
| 5  | 311.89  | 201.78  | 311.89  | 201.78  | 23.89   | 311.89  | 201.78  | 14.11   |
| 6  | 328.00  | 231.33  | 328.00  | 231.33  | 65.56   | 328.00  | 231.33  | 31.56   |
| 7  | 425.89  | 263.11  | 425.89  | 263.11  | 350.11  | 425.89  | 263.11  | 99.33   |
| 8  | 463.67  | 223.11  | 463.67  | 223.11  | 1252.56 | 463.67  | 223.11  | 197.89  |
| 9  | 504.44  | 282.78  | 504.44  | 282.78  | 7672.00 | 504.44  | 282.78  | 1201.22 |
| 10 | 563.56  | 322.22  | 563.56  | 322.22  | 36628.00| 563.56  | 322.22  | 4485.44 |
| 11 |      |       | 648.22  | 648.22  | 648.22  | 648.22  | 648.22  | 648.22  |
| 12 |      |       | 703.44  | 394.33  | 2323515.22| 703.44 | 394.33 | 98503.67 |
| 13 |      |       | 718.67  | 341.22  | 2147825.67| 718.67 | 340.44 | 144323.22|
| 14 |      |       | 797.56  | 405.11  | 4484304.44| 797.56 | 404.22 | 2521655.33|

Table 2. Computational times of BAB algorithms and number of solved and unsolved problems.

| n  | BAB | BAB-Dr |
|----|-----|--------|
|    | Time | Solved | Unsolved | Solved | Unsolved |
| 4  | 0.0020 | 9   | 0     | 0.0012 | 9   | 0     |
| 5  | 0.0067 | 9   | 0     | 0.0031 | 9   | 0     |
| 6  | 0.0207 | 9   | 0     | 0.0074 | 9   | 0     |
| 7  | 0.0202 | 9   | 0     | 0.0903 | 9   | 0     |
| 8  | 0.0447 | 9   | 0     | 0.3414 | 9   | 0     |
| 9  | 0.3146 | 9   | 0     | 2.2607 | 9   | 0     |
| 10 | 10.3326| 9   | 0     | 1.2178 | 9   | 0     |
| 11 | 105.7862| 9  | 0     | 1.9127 | 9   | 0     |
| 12 | 471.1688| 9  | 0     | 36.7838| 9   | 0     |
| 13 | 798.1067| 8   | 1     | 56.7   | 9   | 0     |
| 14 | 1276.588 | 6  | 3     | 640.492| 7   | 2     |

Percentage | 95.95 % | 97.97 %

we saw the effect of the use of variable neighborhood search to improve the performance of the two algorithms: VNSA and VNSA-BAB compared with PUSA algorithm.
Also, we saw that the use of first level of BAB algorithm improve the performance of VNSA-BAB especially for moderate and large size problems. For small size problems, the results in table (4) show that the three algorithms obtain the same values of objective functions and most of these results are optimal. For moderate and large sized instances, the performance of VNSA-BAB is better than the other two algorithms because it restrict the search starting from the second position of the neighborhood sequence through the variable neighborhood search and through the running of SA procedure. The VNSA-BAB algorithm spend more times than other two algorithms because the number of initial solutions obtained by the first level BAB procedure is more than five (MR>5).

Table 3. The computational results of the three algorithms PUSA, VNSA, and VNSA-BAB.

| n  | M.Cmax | PUSA        | VNSA-BAB    | VNSA        |
|----|--------|-------------|-------------|-------------|
|    |        | M.Tmax     | MT          | M.Tmax     | M.T          | M.Tmax     | M.T          |
| 4  | 219.56 | 154.00     | 0.11        | 154.00     | 2.63         | 154.00     | 2.22         |
| 5  | 311.89 | 201.78     | 0.11        | 201.78     | 2.84         | 201.78     | 2.31         |
| 6  | 328.00 | 231.33     | 0.11        | 231.33     | 2.78         | 231.33     | 2.37         |
| 7  | 425.89 | 263.11     | 0.11        | 263.11     | 2.73         | 263.11     | 2.26         |
| 8  | 463.67 | 223.11     | 0.11        | 223.11     | 2.74         | 223.11     | 2.28         |
| 9  | 504.44 | 282.78     | 0.11        | 282.78     | 2.79         | 282.78     | 2.37         |
| 10 | 563.56 | 322.22     | 0.11        | 322.22     | 2.91         | 322.22     | 2.41         |
| 11 | 648.22 | 296.56     | 0.11        | 296.56     | 2.84         | 296.56     | 2.35         |
| 12 | 703.44 | 396.67     | 0.12        | 394.33     | 2.94         | 394.33     | 2.35         |
| 13 | 718.67 | 341.00     | 0.12        | 340.44     | 3.01         | 340.44     | 2.34         |
| 14 | 797.56 | 405.11     | 0.11        | 404.22     | 3.57         | 404.22     | 2.35         |
| 15 | 802.89 | 455.11     | 0.11        | 454.78     | 3.42         | 454.78     | 2.46         |
| 20 | 1073.00| 682.33     | 0.12        | 667.00     | 3.77         | 673.22     | 2.29         |
| 30 | 1542.22| 795.56     | 0.12        | 754.78     | 4.49         | 763.00     | 2.72         |
| 40 | 2132.11| 927.00     | 0.13        | 806.78     | 5.38         | 848.22     | 2.72         |
| 50 | 2598.78| 1071.56    | 0.13        | 779.00     | 8.67         | 944.11     | 2.73         |
| 60 | 3166.44| 1420.67    | 0.13        | 903.33     | 12.63        | 951.56     | 5.32         |
| 70 | 3667.78| 1920.89    | 0.13        | 1486.33    | 28.61        | 1541.33    | 5.67         |
| 80 | 4084.56| 1999.00    | 0.14        | 1057.33    | 31.67        | 1178.22    | 6.01         |
| 90 | 4735.44| 3160.89    | 0.14        | 2068.33    | 33.07        | 2273.11    | 6.36         |
| 100| 5220.78| 3934.22    | 0.14        | 2377.22    | 39.87        | 2558.89    | 6.78         |
| 120| 6391.22| 4276.67    | 0.15        | 2714.78    | 42.02        | 2886.44    | 6.37         |
| 130| 6665.78| 4960.44    | 0.15        | 3550.11    | 72.01        | 3894.78    | 6.25         |
| 140| 7324.78| 5707.33    | 0.15        | 3505.89    | 73.25        | 3565.89    | 6.37         |
| 150| 7555.89| 5729.11    | 0.16        | 3838.78    | 63.05        | 3892.67    | 8.68         |
| 175| 9091.67| 6771.89    | 0.16        | 4532.56    | 93.50        | 4665.56    | 7.50         |
| 200| 10385.22| 8446.00   | 0.17        | 5967.89    | 24.53        | 6104.33    | 7.39         |
| 250| 12951.00| 10832.89  | 0.19        | 7655.22    | 71.94        | 7862.22    | 7.89         |
| 300| 15442.33| 12954.67  | 0.21        | 10331.11   | 141.21       | 10355.89   | 10.10        |
5. Conclusions

In this paper, we consider the 2-machine PrFS problem to minimize in loxicographical order the two objective functions, makespan (primary objective function) and maximum tardiness (secondary objective function), the problem denoted $F_2 || \text{Lex}(C_{\text{max}}, T_{\text{max}})$. We propose two algorithms to solve the problem, BAB algorithm and VNSA-BAB. Also we proposed dominance conditions that improve the performance of BAB algorithm by reducing the number of visited nodes and computational times. However, the results show that the BAB algorithm can find the optimal solution for instants up to $n=14$. For the proposed VNSA-BAB, the results show the efficiency of using the variable neighborhood search and the first level of BAB algorithm to solve the considered problem. The future research can be done through the application of the proposed algorithms in practical production scheduling problems.

6. References

[1] J. M. Framiñán, R. Leisten, and R. R. García, “Manufacturing scheduling systems, an integrated view on models,” *Methods and Tools*, pp. 51–63, 2014.
[2] S. M. Johnson, “Optimal two-and three-stage production schedules with setup times included,” *Nav. Res. Logist. Q.*, vol. 1, no. 1, pp. 61–68, 1954.
[3] M. Pinedo, *Scheduling*, vol. 5. Springer, 2012.
[4] R. L. Daniels and R. J. Chambers, “Multiobjective flow-shop scheduling,” *Nav. Res. Logist.*, vol. 37, no. 6, pp. 981–995, 1990.
[5] C. Rajendran, “Two-stage flowshop scheduling problem with bicriteria,” *J. Oper. Res. Soc.*, vol. 43, no. 9, pp. 871–884, 1993.
[6] W.-C. Lee and C.-C. Wu, “Minimizing the total flow time and the tardiness in a two-machine flow shop,” *Int. J. Syst. Sci.*, vol. 32, no. 3, pp. 365–373, 2001.
[7] V. T’Kindt, J. N. D. Gupta, and J.-C. Billaut, “A branch-and-bound algorithm to solve a two-machine bicriteria flowshop scheduling problem,” 2001.
[8] B. Toktaş, M. Azizoğlu, and S. K. Kóksalan, “Two-machine flow shop scheduling with two criteria: Maximum earliness and makespan,” *Eur. J. Oper. Res.*, vol. 157, no. 2, pp. 286–295, 2004.
[9] B. M. Lin and J. M. Wu, “Bicriteria scheduling in a two-machine permutation flowshop,” *Int. J. Prod. Res.*, vol. 44, no. 12, pp. 2299–2312, 2006.
[10] G. Moslehi, M. Mirzaei, M. Vasei, M. Modarres, and A. Azaron, “Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness,” *Int. J. Prod. Econ.*, vol. 122, no. 2, pp. 763–773, 2009.
[11] E.-G. Talbi, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.
[12] R. K. Suresh and K. M. Mohanasundaram, “Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives,” in *IEEE Conference on Cybernetics and Intelligent Systems*, 2004., 2004, vol. 2, pp. 712–717.
[13] M. Mesgarpour, N. Kirkavak, and H. Ozaktas, “Bicriteria scheduling problem on the two-machine flowshop using simulated annealing,” in *European Conference on Evolutionary Computation in Combinatorial Optimization*, 2010, pp. 166–177.
[14] B. S. H. Khan and K. Govindan, “A multi-objective simulated annealing algorithm for permutation flow shop scheduling problem,” *Int. J. Adv. Oper. Manag.*, vol. 3, no. 1, pp. 88–100, 2011.
[15] V. A. Armentano and J. E. Claudio, “An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem,” *J. Heuristics*, vol. 10, no. 5, pp. 463–481, 2004.
[16] M. J. Geiger, “On operators and search space topology in multi-objective flow shop scheduling,” *Eur. J. Oper. Res.*, vol. 181, no. 1, pp. 195–206, 2007.
[17] B. Shahul Hamid Khan, G. Prabhaharan, and P. Asokan, “A grasp algorithm for multi-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness,” *Int. J. Comput. Math.*, vol. 84, no. 12, pp. 1731–1741, 2007.
[18] J. E. C. Arroyo and A. A. de Souza Pereira, “A GRASP heuristic for the multi-objective permutation flowshop scheduling problem,” Int. J. Adv. Manuf. Technol., vol. 55, no. 5–8, pp. 741–753, 2011.

[19] S. Cavalieri and P. Gaiardelli, “Hybrid genetic algorithms for a multi-objective scheduling problem,” J. Intell. Manuf., vol. 9, no. 4, pp. 361–367, 1998.

[20] H. Ishibuchi, T. Yoshida, and T. Murata, “Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,” IEEE Trans. Evol. Comput., vol. 7, no. 2, pp. 204–223, 2003.

[21] S. G. Ponnambalam, H. Jagannathan, M. Kataria, and A. Gadicherla, “A TSP-GA multi-objective algorithm for flow-shop scheduling,” Int. J. Adv. Manuf. Technol., vol. 23, no. 11–12, pp. 909–915, 2004.

[22] N. Melab, M. Mezmaz, and E.-G. Talbi, “Parallel cooperative meta-heuristics on the computational grid: A case study: the bi-objective flow-shop problem,” Parallel Comput., vol. 32, no. 9, pp. 643–659, 2006.

[23] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, “Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm,” J. Clean. Prod., vol. 144, pp. 228–238, 2017.

[24] J. Deng and L. Wang, “A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem,” Swarm Evol. Comput., vol. 32, pp. 121–131, 2017.

[25] X. Li and S. Ma, “Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times,” IEEE Trans. Eng. Manag., vol. 64, no. 2, pp. 149–165, 2017.

[26] V. T’kindt and J.-C. Billaut, “Some guidelines to solve multicriteria scheduling problems,” in IEEE SMC’99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028), 1999, vol. 6, pp. 463–468.

[27] M. M. Yenisey and B. Yagmahan, “Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends,” Omega, vol. 45, pp. 119–135, 2014.

[28] T. S. Abdul-Razaq and H. M. Motair, “On the Two Machine Permutation Flowshop Scheduling Problems Using Variable Neighborhood Search Algorithms,” Almuthanna J. Pure Sci., vol. 5, no. 1, 2018.

[29] J. N. D. Gupta, K. Hennig, and F. Werner, “Local search heuristics for two-stage flow shop problems with secondary criterion,” Comput. Oper. Res., vol. 29, no. 2, pp. 123–149, 2002.