# Using local search methods
# To solve two machine flow shop scheduling Problem

**Waffa Abdul-Abbas**
**Dept. of mathematics / College of Education,**
**University of Thi-Qar.**

**Keywords**
**Scheduling, Branch and bound, two machine flow shop, local search, descent method (DM), adjacent pairwise interchange method (APIM).**

## Abstract

**There are a lot of scheduling problems that have a combinatorial aspect and these problems are difficult to be solved. Therefore we can use the local search methods to find the optimal solution or near optimal solution.**

**In this paper we consider the scheduling problem on two machine flow shop to find the minimum value of the objective function (maximum completion time and maximum of tardiness).The main contribution in this work is a branch and bound(BAB) algorithm with optimal solution and some of the local search methods namely descent method (DM), adjacent pairwise interchange method (APIM). Also, prove some of special cases of which leads to optimal solution. Since the problem is NP-hard, descent method (DM)and adjacent pairwise interchange method (APIM) are proposed to solve the problem efficiency.**

**1-Introduction**

**We shall introduce a number of basic definitions which explain the structure of scheduling problems. The machine scheduling problem that forms the object of this paper can be defined as follows:**

**Sequencing number of jobs on one or more machines for a given period of time to minimize, a given objective function, which can be either single or multi objective function, subject to some constraints. A schedule specifies, for each machine i and j, one or more time intervals through which processing is performed on j by i. Sequencing, generally speaking, means to assign machines to jobs in some order to complete all these jobs under the imposed constraints. The problem is to find the optimal processing order for these jobs on each machine such that minimizes the given objective function. There are two general constraints in classical scheduling theory [2]. Each job is to be processed by; at most, one machine at a time and each machine is capable of processing at most one job at time. A schedule is feasible if it satisfies the two general constraints, and also if it satisfies the various requirements relating to the specific problem. The problem type is specified by the machine environment, the characteristics and an optimality criterion.**

**2-Two machine flow shop problem**

**The general flow shop problem, in general would be indicated by $F_m$ / / $C_{max}$ , can be stated as follows. There are n jobs numbered 1, …,n, each of which is to be processed on machines 1, …, m in some order. Each job i (i= 1, …, n) has a processing time $P_{ik}$ on machine k (k= 1, …, m). Each machine can be processed not more than one job at a time and each job can be processed by not more than one machine at a time. The order in which jobs are processed need not be the same on all machines. The objective is to find a processing sequence order on each machine which minimizes $C_{max}$ ( the maximum completion time of all the jobs).**

**For $F_m$ / / $C_{max}$ problem, Conway, Maxwell and Miller [4] observed that there exists an optimal schedule with the same processing order of jobs on the first pair of machines and the same order on the last pair of machines.**

**It is also well known that for m=2, the resulting flow shop problem (i.e. $F_2 // C_{max}$), can be solved using Johnson's algorithm [6] in which job i is sequenced before job j if**

**min $\{P_{i1}, P_{j2}\} \leq \min\{P_{i2}, P_{j1}\}$.**

**Note that using other criteria usually leads to NP- hard problems for example $F_2 // L_{max}$ (Lenstra et al, [8]).**

**3- Problem Formulation**

**To state our scheduling problem more precisely, we are given n jobs which are numbered 1, …, n. All jobs are available for processing at time zero and are to be processed on machines 1 and 2 in that order during uninterrupted times $a_i$ and $b_i$ respectively. The objective is to find a processing order of the jobs that minimizes the composition objective function (maximum of completion time and maximum of tardiness on the second machine), this objective function is denoted by ($C_{max} + T_{max}$).Using the three field classification suggested by Graham et al. [5], This problem denoted as $F_2 // C_{max} + L_{max}$. The problem under investigation is known to be NP-complete, since $F_2 // T_{max}$ is NP-complete (Lenstra et al. [8]). Given any sequence π = (π (1), …, π(n)), the minimum completion times $C^A_{\pi(1)}$ and $C_{\pi(1)}$ of the first job π(1), in the sequence on the first and second machine are equal to $a_{\pi(1)}$ and $a_{\pi(1)} + b_{\pi(1)}$ respectively. The minimum complation times of any other job π(i), (i=2, …, n) on the first and second machines are given by**

$C^A_{\pi(i)} = C^A_{\pi(i-1)} + a_{\pi(i)}$ **and** $C_{\pi(i)} = \max\{C^A_{\pi(i)}, C^A_{\pi(i-1)}\} + b_{\pi(i)}$ **respectively, hence the tardiness for each job π(i), i=1, …, n on the second machine is given by**

$T_{\pi(i)} = \max\{C_{\pi(i)} - d_{\pi(i)}, 0\}$, **where** $d_{\pi(i)}$ **is the due date of job π(i).**

**The objective is to find a schedule σ (σ(1), …, σ(n)) of the jobs that minimizes the total cost Z(σ). Our problem (p) can formally be stated as :**

$$\min_{\sigma \in S} \{Z(\sigma)\} = \text{Min } \{C_{max} + T_{max}\}$$

**s.t.**

$$C_{\sigma(1)} = C^{A}_{\sigma(1)} + b_{\sigma(1)}$$

$$C_{\sigma(i)} = \max \{C^{A}_{\sigma(i)}, C^{A}_{\sigma(i-1)}\} + b_{\sigma(i)} \qquad i = 2,\ldots, n$$

$$C_{max} = C_{\sigma(n)}$$

$$T_{\sigma(i)} \geq C_{\sigma(i)} - d_{\sigma(i)} \qquad i = 1,\ldots, n$$

$$T_{max} = \max \{T_{\sigma(i)}\}$$

**(P)**

**where σ (i) denoted the position of job i in the ordering σ and S denotes the set of all sequences.**

**3-1 Special cases( yields optimal solutions):**

**Finding a special case for scheduling problem means finding an optimal schedule directly without using BAB algorithm. A special case (if it exists) depends on satisfying some conditions in order to make the problem easily solvable. These conditions depend on the objective function as well as on the jobs. Now we shall give the following special cases:**

**Case (1): The sequence which is obtained by Johnson's rule is optimal for $F_2$ / / $C_{max}$+ $T_{max}$ problem if $T_i = 0$ for each job i (i=1, …, n).**

**Proof:**

**Since all jobs are early or completed on time this mean that there are no tardiness (i.e. $T_i = 0$ for each i) and hence $T_{max} = 0$, and our problem F2 / / $C_{max}$+ $T_{max}$ is reduced to $F_2$ / / $C_{max}$ which can be solved by Johnson's rule (J. R).**

**Case (2): The sequence which is obtained by Johnson's rule is optimal for $F_2$ / / $C_{max}$+ $T_{max}$ problem if $d_i = d$ for each i (i=1, …, n).**

**Proof:**

**Let σ = (1, …, n) be the sequence obtained by Johnson's rule ( J. R) and the minimum $C_{max}$ is given by $C_{max} = \max \{C_{n-1}, C^{A}_{n}\} + b_n$**

where $C_{n-1}$ is the completion time of job n-1 on machine B, $C_n^A$ is the completion time of job n on machine A, $b_n$ is the processing time of the last job n on machine B.

For each job i $\in$ σ, $T_i = C_i - d$ and hence the maximum tardiness $T_{max} = C_{max} - d$, which is less than or equal to the maximum tardiness given by any other sequence $\pi \neq \sigma$. Hence σ is optimal for $F_2 // C_{max} + T_{max}$.

**3-2 Derivation the Upper and Lower bounds for the problem (P):**

Consider the two machine flow-shop problem to be minimized the composition objective function, the maximum of completion time and maximum of tardiness ($C_{max} + T_{max}$). This problem is clearly NP-hard since the simpler version $F_2 // T_{max}$ is already NP-hard [7]. It is well known that computation can be reduced by using a heuristic approach to find a good solution to act as an upper bound (UB). Also a simple technique is used to obtain a lower bound (LB) for our problem (P).

**3-3 Upper bound (UB):**

We can find upper bound (UB) for our problem (P) by using Johnson's rule (J.R), since Johnson's rule gives the optimal solution to the important part $C_{max}$ of this problem.

**3-4 Lower bound (LB):**

Decomposition of the problem and derivation of lower bound (LB). Now consider again the formulation of the problem (P), we decompose the problem into two subproblems with a simpler structures. Then the lower bound (LB) of the problem (P) is the sum of the minimum value of the problem ($P_1$) and the lower bound of the sub problem ($P_2$).

$$Z_1 = \min_{\sigma \in S} \{C_{max}\}$$

s.t.

$$C_{\sigma(1)} = C^A_{\sigma(1)} + b_{\sigma(1)}$$

$$C_{\sigma(i)} = \max \{C^A_{\sigma(i)}, C^A_{\sigma(i-1)}\} + b_{\sigma(i)} \qquad i = 2,\ldots, n$$

$$C_{max} = C_{\sigma(n)}$$

($P_1$)

**And**

$$\left.\begin{aligned} &\mathbf{Z_2 = \min T_{max}} \\ &\mathbf{s.t.} \\ &T_{\sigma(i)} \geq C_{\sigma(i)} - d_{\sigma(i)} \\ &\mathbf{T_{max} = Max\ \{} T_{\sigma(i)} \mathbf{\}} \\ &T_{\sigma(i)} \geq \mathbf{0} \end{aligned}\right\} \mathbf{(P_2)}$$

**It is clear from the decomposition that ($P_1$) and ($P_2$) have simple structures then (P), and thus appear easily to solve optimality for ($P_1$) (i. e.($P_1$) is solved by Johnson's rule), and a lower bound can be obtained for ($P_2$) by using the relaxation theqniquies. The lower bound for ($P_2$) is obtained as follows. Let $a_{i=0}$ for each job i and the resulting problem is 1/ / $T_{max}$ which is solved by EDD rule (the job i is sequenced before job j if $d_i \leq d_j$ (i, j=1, …, n) let LBT= min{ $T_{max}$} for ($P_2$). Hence $LB=Z_1+LBT$.**

**4-The Branch and Bound Method (BAB)**

**The BAB method starts by applying the special cases given in section (3-1). If the data for the (P) satisfy the conditions of one of the special cases ((1) or (2)) then the (P) is considered to be solved. If not, at this stage a BAB method should be used.**

**To get an optimal solution for our (P), The (BAB) method is used. At the root node of the search tree J.R. is used to generate upper bound UB on the cost of the optimal schedule.**

**Also at the root node of the search tree an initial LB on the cost of an optimal schedule is obtained from LB given in section (3-4). For all nodes, we can use the bounding procedure to calculate LB. If LB for any node is greater than or equal to the current UB already computed, then this node is discarded, otherwise it may by selected for next branching.**

**The BAB method uses a forward sequencing branching rule for which nodes at level k of the search tree correspond to initial partial sequences in which jobs are sequenced in the first k position. An adjacent job interchange rule is applied at each node of the search tree, except those at the first level in which only one job is sequenced, in an attempt to eliminate nodes through the dominance theorem of**

**dynamic programming(DP). At the current node, the adjacent job interchange rule compares the cost of the last two jobs of the initial partial sequence with the corresponding cost when the jobs are interchanged; if the formed cost is larger, then the current node is eliminated, while if both costs are the same, some convention is used to decide whether the current node should be discarded.**

**The BAB method continues in a similar way by using forward branching procedure. Whenever a complete sequence is obtained, this sequence is evaluated and the UB is altered if the new objective value is less than the old one. The procedure is repeated until all nodes have been considered.**

**5-General approach of local search method:**

**There are numerous combinatorial optimization problems for which computing exact optimal solution is computationally intractable those known as NP-hard problems. As a consequence, much effort has been devoted to construct algorithms that can find high quality approximate solutions, in reasonable running times, such as local search called also neighborhood search methods. These methods can be viewed as tools for searching a space of legal alternatives in order to find the best solution within reasonable time limitation. This section describes the local search method to solve the two flow-shop machine scheduling problem to minimize the maximum of completion time and the maximum of tardiness. It is well know that many flow shop scheduling problems have been shown to be NP-hard, the computational requirements are enormous for large sized problem to avoid this draw back we can appeal to local search method.**

**The use of search technique presupposes definitions of the problem and a neighboring in scheduling problems, as follows:**

**Definition (5-1) [9]:**

**A neighborhood function *N* is a mapping *N* : S $\longrightarrow$ S with specifies for each ((s $\in$ S) a subset *N*(s) of S of neighbors of s.**

**We can introduce four cases of *N*(s) as follows:**

**1) Inser Neighborhood**

**$N_{ins}(S) = \{S_{i \leftarrow j} / i \neq j\}$. Here $(S_{i \leftarrow j})$ is the sequence obtained from S by moving the j-th job to the location before the i-th job.**

**2) Swap Neighborhood**

**$N_{ins}(S) = \{S_i \longleftrightarrow {}_j / i \neq j\}$. Here $(S_i \longleftrightarrow {}_j)$ is the sequence obtained by interchanging the i-th job and j-th job of s.**

- **A special case is transposing $N_{tra}$ where the two jobs are adjacent.**

**5-2 Adjacent pairwise interchange method (APIM).**

**This (APIM) depends on interchange elements (jobs) at positions (i) and (i+1) of a given sequence, (i=1, …, n-1) [3]. The following steps describe this method:**

**(1) Initialization:**

**To obtain an initial current solution jobs are ordered according to Johnson rule (J. R.), by sequencing job i with $(a_i \leq b_i)$ first in non-decreasing other of $a_i$ followed by the remaining jobs i with $(a_i > b_i)$ sequencing in non-decreasing order of $b_i$ (i=1, …, n) to obtain the current sequence σ (σ(1), …, σ(n)), with its objective function value (UB), where $UB=C_{max}+T_{max}$.**

**(2)- Neighbor Generation:**

**In order to improve the sequence σ, the position of two adjacent jobs σ(i), σ(i+1) , (1≤i≤n-1) are transposed. Hence a new sequence σ* is obtained with its objective function value $UB^*= C_{max}+T_{max}$.**

**(3)-Evaluation:**

**If the improvement is made (i. e UB*<UB) then, the two jobs are left in their new positions. On the other hand, the two jobs are replaced in their original positions. The procedure is then repeated from step (2) and other possibilities are considered in a similar way.**

**(4)- Termination Step:**

**The method is terminated when all possibilities are considered for adjacent jobs σ(i), σ(i+1) , (1≤i≤n-1), without making any improvement.**

**5-3 Descent Method (DM)**

**This method is a simple form of local search methods. It can be executed as follows:**

**(1) Initialization:**

**In this step, the feasible solution σ=σ (σ(1), …, σ(n)) obtained from Johnson rule (J. R.), is chosen to be the initial current solution for descent method, with its objective function value(IUB).**

**(2)- Neighbor Generation:**

**In this step, the feasible solution σ*=σ* (σ*(1), …, σ*(n)) of the current solution is generated by choosing randomly two jobs from σ in the first stage, (not necessarily adjacent) and transpose their positions, and a feasible neighbor is also generated by choosing randomly a block of jobs from σ at the second stage, and transpose their positions, for each case calculate the function values and the minimum value is denoted by (CUB).**

**(3)-Acceptance Test:**

**Now consider the test whither to accept the move from σ to σ* or not, as follows:**

**(a)- If CUB <IUB: then σ* replace σ as the current solution and we set IUB= CUB, and go to step(2) (Neighbor generation).**

**(b)- Otherwise (i.e. UB1≤ CUB): σ is retained as the current solution, and go to step (2)**

**(5)- Termination Test:**

**Repeat step (2) and other possibilities are considered in a similar way. The DM terminates if no neighbor provides an improved objective function value, in which case the current solution IUB is a local minimum.**

**6-Test Problems**

**In selecting test problems, one important goal is to create problem instances that are representatives of the general problem class. We generated experiments the test problems with value of n (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150). When comparing the performance of algorithms, it is important to test them on a range of problem instances. The main characteristic of an instance of our scheduling problem is its size, as measured by number of jobs n.**

**Our test problems were generated as follows: [1]. The processing times $a_i$ and $b_i$ in the test problems were randomly sampled from a uniform distribution on the**

**integers defined on [1, 10], and due dates were generated from the uniform distribution on**

**[(1-TF-RDD/2) sp, (1-TF+RDD/2) sp] such that**

**sp=** $\sum_{i=1}^{n} Ci$ **where** $C_i = (a_i + bi)/2$

***TF*** **= 0.2, 0.4,**

***RDD*** **= 0.2, 0.4, 0.6, 0.8, 1**

**Since *n* = 10 and 20 are the sizes that were solved to optimality by the BAB method given in section (4), the other sizes should also be tested by using descent method (DM) and adjacent pairwise interchange method (APIM).**

**Table (1) Comparative Computation Results**

| n | no | UB | LB | BAB |
|---|---|---|---|---|
| 5 | 1 | 41 | 30 | 35 |
| | 2 | 37 | 27 | 34 |
| | 3 | 37 | 26 | 34 |
| | 4 | 46 | 32 | 41 |
| | 5 | 62 | 43 | 49 |
| 10 | 1 | 88 | 70 | 81 |
| | 2 | 85 | 79 | 83 |
| | 3 | 95 | 78 | 90 |
| | 4 | 100 | 82 | 93 |
| | 5 | 114 | 101 | 105 |
| 15 | 1 | 118 | 99 | 101 |
| | 2 | 150 | 121 | 126 |
| | 3 | 136 | 100 | 105 |
| | 4 | 157 | 122 | 131 |
| | 5 | 122 | 87 | 97 |
| 20 | 1 | 173 | 146 | 156 |
| | 2 | 170 | 154 | 162 |
| | 3 | 187 | 156 | 176 |
| | 4 | 199 | 181 | 189 |
| | 5 | 220 | 187 | 206 |

| | | UB | LB | APIM | | DM | |
|---|---|---|---|---|---|---|---|
| | | | | Values | Times | Values | Times |
| 10 | 1 | 88 | 70 | 81 | 0.06 | 81 | 0.06 |
| | 2 | 85 | 79 | 83 | 0.01 | 84 | 0.03 |
| | 3 | 95 | 78 | 90 | 0.03 | 90 | 0.03 |
| | 4 | 100 | 82 | 93 | 0.1 | 93 | 0.2 |
| | 5 | 114 | 101 | 105 | 0.02 | 107 | 0.02 |
| 20 | 1 | 173 | 146 | 156 | 0.03 | 161 | 0.03 |
| | 2 | 170 | 154 | 162 | 0.01 | 169 | 0.01 |
| | 3 | 187 | 156 | 176 | 0.1 | 176 | 0.1 |
| | 4 | 199 | 181 | 189 | 0.1 | 193 | 0.2 |
| | 5 | 220 | 187 | 206 | 0.03 | 208 | 0.03 |
| 30 | 1 | 341 | 305 | 318 | 0.2 | 318 | 0.2 |
| | 2 | 255 | 235 | 243 | 0.1 | 246 | 0.1 |
| | 3 | 279 | 253 | 272 | 0.2 | 272 | 0.5 |
| | 4 | 288 | 278 | 282 | 0.1 | 282 | 0.2 |
| | 5 | 303 | 270 | 298 | 0.1 | 298 | 0.1 |
| 40 | 1 | 347 | 310 | 325 | 0.2 | 325 | 0.1 |
| | 2 | 397 | 365 | 389 | 0.2 | 385 | 0.2 |
| | 3 | 399 | 379 | 388 | 0.2 | 392 | 0.3 |
| | 4 | 340 | 304 | 329 | 0.1 | 329 | 0.2 |
| | 5 | 382 | 354 | 371 | 0.1 | 371 | 0.1 |
| 50 | 1 | 423 | 389 | 410 | o.2 | 415 | o.3 |
| | 2 | 425 | 383 | 412 | 0.2 | 412 | 0.6 |
| | 3 | 486 | 449 | 456 | 0.2 | 456 | 0.5 |
| | 4 | 535 | 495 | 522 | 0.1 | 518 | 0.2 |
| | 5 | 594 | 512 | 573 | 0.2 | 573 | 0.2 |
| 75 | 1 | 678 | 616 | 659 | 0.2 | 657 | 0.4 |
| | 2 | 709 | 672 | 693 | 0.1 | 693 | 0.1 |
| | 3 | 711 | 674 | 699 | 0.2 | 702 | 0.2 |
| | 4 | 728 | 673 | 683 | 0.3 | 690 | 0.2 |
| | 5 | 705 | 673 | 689 | 0.1 | 689 | 0.2 |
| 100 | 1 | 897 | 851 | 875 | 1.2 | 877 | 1.1 |
| | 2 | 995 | 912 | 981 | 1.3 | 981 | 1.3 |
| | 3 | 927 | 908 | 915 | 1.2 | 918 | 1.1 |
| | 4 | 944 | 915 | 932 | 1.2 | 935 | 1.5 |
| | 5 | 997 | 952 | 983 | 1.2 | 983 | 1.2 |
| 150 | 1 | 1381 | 1329 | 1357 | 2.5 | 1362 | 2.5 |
| | 2 | 1450 | 1387 | 1410 | 2.3 | 1419 | 2.6 |
| | 3 | 1448 | 1391 | 1408 | 2.5 | 1412 | 2.5 |
| | 4 | 1429 | 1344 | 1387 | 2.5 | 1380 | 2.7 |
| | 5 | 1411 | 1392 | 1405 | 2.3 | 1407 | 2.3 |

**Comparative Results**

**This section shows the efficiency of local search methods, comparing them with the optimal solution (obtained by BAB algorithm) for each test problem. For the methods (BAB, DM and APIM), we present table of results which show the efficiency of this method. we return to calculate the optimal solution for each test problem which is obtained by BAB algorithm, these optimal solution values are used to assess the quality of solutions generated by local search methods and, for BAB algorithm, whenever a problem was not solved within a number of nodes greater than 1000000 computation was bounded for the problem.**

**For all $n \leq 20$ job problems the optimal solution are available by using BAB algorithm, It is clear from table (1) that the value of lower bound (LB) and upper bound (UB) closed to the optimal solution.**

**The problem from $n>20$ are unsolved because the number of nodes is greater than or equal to 1000000, hence we don't use the BAB method for $n>20$.**

**Since the BAB algorithm can not generate optimal solution for test problem (with $n>20$). In this case, local search methods are used to obtain the best solution value and forms the basis for comparison. Table (2) gives the results of comparison between BAB and local search methods. Also shows the results for lower and upper bounds for our problem.**

**The results in table (2) show that the local search methods(DM and APIM) perform very well, Also it is clear from table (2) that APIM has the best value and time with respect to DM.**

## References


1. **Anderson E. J., Glass C. A. and Potts C. N.,"Application of local search in machine scheduling", Mach (1995).**
2. **Blazewicz J., Ecker K.H., P Esch E. Schmidt G., and Weglarz J.," Scheduling Computer and manufacturing processes", Spring verlay Berlin. Heidelberg (1996).**
3. **Chen, B., A better heuristic for preemptive parallel machine scheduling with batch set-up time. SIAM Jornal on computing 22-1303-1318(1993).**

4. **Conway, R. W., Maxwell W. L. and Miller L.W. "Theory of scheduling" Addison Wesley, Reading, MA, (1967).**

5. **Graham R. L., Lawler E. L., Lenstra J. K., and Rinnooy Kan A. H. G., "Optimization and approximation in deterministic sequencing and scheduling theory :a survey" , Discrete math. 5, 287-326(1979).**

6. **Johnson, S.M., "Optimal two and three stage production schedules with set-up times included", Naval Research Logistic Quart, 1, 61-68, (1954).**

7. **Karp R. M., Reducibility among combinatorial problems. In complexity of computer computations, Miller R. E and Thatcher J. W. Eds. Plenum press, New York, 95-103(1972).**

8. **Lenstra J. K., Rinnooy Kan A. H. G. , and Brucker P., "Complexity of Machine Scheduling Problems". Ann. Of discrete mathematics, 1, 343-362(1977).**

9. **Tjark Vredereld, (2002)" Combinatorial Approximation Algorithms Guaranteed Versus Experimental performance".**