On the Solution of 2-Machine Flow and Open Shop Scheduling Problems with Secondary Criteria

Jatinder N.D. Gupta
Ball State University, Department of Management,
Muncie, USA
email: jgupta@bsu.edu
Frank Werner
Otto-von-Guericke-Universität, Fakultät für Mathematik, PSF 4120,
39016 Magdeburg, Germany
email: frank.werner@mathematik.uni-magdeburg.de

Keywords: Flow Shop, Open Shop, Constructive and Iterative Heuristics, Problems with Secondary Criteria

1 Introduction

Traditional research to solve multi-stage scheduling problems has focused on single criterion. However, based on a survey of industrial scheduling practices, Panwalkar et al. [18] reported that managers develop schedules based on multicriteria. Therefore, the research for single criterion scheduling problems may not be applicable to multicriteria problems. Because of this, multicriteria scheduling problems are receiving much attention recently [16].

Multicriteria scheduling problems can be modeled in three different ways. First, when the criteria are equally important, we can generate all the efficient solutions for the problem. Then by using multiattribute decision methods, tradeoffs are made between these solutions. Second, when the criteria are weighted differently, we can define an objective function as the sum of weighted functions and transform the problem into a single criterion scheduling problem. Finally, when there is a hierarchy of priority levels for the criteria, we can first solve the problem for the first priority criterion, ignoring the other criteria and then solve the same problem for the second priority criterion under the constraint that the optimal solution of the first priority criterion does not change. Termed *scheduling with secondary criteria*, this procedure is continued until we solve the problem with last priority criterion as the objective and the optimal solutions of the other criteria as the constraints.

For a scheduling problem with two criteria of interest, if the first two approaches are required, we call the problem a *bicriteria* scheduling problem. If the second approach is required, we term the problem a *secondary* criterion problem. Using the standard three field notation [15], a bicriteria scheduling problem for finding all efficient solutions can be represented as $\alpha|\beta|F(C1, C2)$, where α denotes the machine environment, β corresponds to the deviations from standard scheduling assumptions, and $F(C1, C2)$ indicates that efficient solutions relative to criteria $C1$ and $C2$ are being sought. If the bicriteria scheduling problem involves the sum of weighted values of two objective functions, the problem is denoted as $\alpha|\beta|F_w(C1, C2)$, where $F_w(C1, C2)$ represents the weighted sum of two criteria $C1$ and $C2$. Similarly, a secondary criterion problem can be denoted as $\alpha|\beta|F_h(C2/C1)$, where $C1$ and $C2$ denote the primary criterion and secondary criterion, respectively, and the notation $F_h(C2/C1)$ represents the *hierarchical* optimization of criterion $C2$ given that criterion $C1$ is at its optimal value.

The literature on multiple and bicriteria problems for single machine problems is summarized by Dileepan and Sen [5], or Fry et al. [7]. Nagar et al. [16] provide a detailed survey of the multiple and bicriteria scheduling research involving multiple machines. Chen and Vempati [3] developed a backward branch-and-bound algorithm for problem $F2||F_h(\sum C_i/C_{\max})$. A forward branch and bound algorithm for problem $F2||F_h(\sum C_i/C_{\max})$ is developed by Rajendran [19]. However, both Rajendran's, and Chen and Vempati's algorithms cannot efficiently solve problems involving 20 or more jobs. Rajendran [19] developed two heuristics for problem $F2||F_h(\sum C_i/C_{\max})$ problem and tested their effectiveness in solving problems involving 25 or less jobs. Other constructive algorithms for this problem have been given in [12], where problems with up to 80 jobs have been considered. A genetic algorithm for this problem has been given in [17]. This algorithm has been tested on problems also with up to 80 jobs.

The problems under consideration can be described as follows. n jobs have to be processed on two machines M_1 and M_2. For each job j,

a processing time a_j on M_1 and a processing time b_j on M_2 are given. Preemptions of operations are not allowed. In this paper, we develop and compare different constructive and iterative heuristics to solve approximately the problems $F2||F_h(\sum C_i/C_{\max})$, $F2||F_h(\sum w_iC_i/C_{max})$ and $F2||F_h(\sum w_iT_i/C_{max})$ and the corresponding counterparts of the 2-machine open shop problem. Thus, the secondary criteria are the total flow time, the total weighted flow time, and the weighted total tardiness. Whereas for the flow shop problem all jobs have to be processed first on M_1 and then on M_2, in the open shop case the machine order can be chosen arbitrarily.

While problem $F2||C_{\max}$ can be solved in $O(n \log n)$ computational time [14], the problem $F2||\sum C_i$ is NP-hard in the strong sense [8]. From these results, it follows that problem $F2||F_h(\sum C_i/C_{\max})$ and also the other two problems considered are NP-hard in the strong sense [2]. Similarly, the 2-machine open shop problem with makespan minimization is polynomially solvable whereas the other traditional criteria already lead to NP-hard problems.

The rest of the paper is organized as follows. Section 2 describes the neighbourhoods that we apply in the local search algorithms, and we describe the different algorithms for the flow shop problem. Some computational results of our experiments are summarized in Section 3. In Section 4 we give some heuristics for 2-machine open shop problems with secondary criterion.

2 Constructive and iterative algorithms for flow shop

In this section we discuss some constructive and iterative heuristics suggested in [11] and [12]. Since a random solution may not satisfy the minimum makespan constraint, some heuristic algorithms that ensure a schedule with optimal makespan have been developed. Johnson's algorithm can be interpreted as a 'bad' solution since no optimization is performed with respect to the secondary criterion. A job insertion algorithm *INS* can be given as follows, where $J_{\overline{\sigma}}$ denotes Johnson's sequence for the jobs not contained in sequence σ, C stands for C_{max} and F for the secondary criterion.

1. Let $J = (\mu(1), \ldots, \mu(n))$ be Johnson's schedule of all n jobs. Let $\alpha = (\alpha(1), \ldots, \alpha(n)) = J = (\mu(1), \ldots, \mu(n))$, $C(\alpha) = C(J)$ and $F(\alpha) = F(J)$. Set $\omega = \{\emptyset\}$ and $r = 0$. Let $i = 1$, and $\sigma = \mu(1)$. Further, let $\sigma = \sigma_p\sigma_{i-p}$ for $0 \leq p \leq i$. Enter step 2.

2. For each of the $(n-i)$ jobs $\notin \sigma$, generate $(n-i) * (i+1)$ partial sequences represented by $\sigma_p k \sigma_{i-p}$, where $k \notin \sigma$ and $0 \leq p \leq i$. For each of the generated partial sequence, if $C(\sigma_p k \sigma_{i-p} J_{\overline{\sigma_p k \sigma_{i-p}}}) = C(J)$, set $\omega = \{\omega, \sigma_p k \sigma_{i-p}\}$ and $r = r + 1$. Enter step 3.

3. Let $\omega = \{\omega_1, \omega_2, \ldots, \omega_r\}$. For each $q \leq r$, if $F(\omega_q J_{\overline{\omega_q}}) < F(\alpha)$, set $F(\alpha) = F(\omega J_{\overline{\omega_q}})$ and $\alpha = (\omega_q J_{\overline{\omega_q}})$. Let $F(\omega_\nu) = \min_{1 \leq j \leq r}\{F(\omega_j)\}$. Set $\sigma = \omega_\nu$, $r = 0$, and $\omega = \{\emptyset\}$. Enter step 4.

4. If $i < n$, set $i = i+1$ and return to step 2; otherwise accept the schedule α with $C2 = F(\alpha)$ and makespan $C(\alpha)$ as the solution of the problem.

Algorithm *INS* time requirement is $O(n^4)$. Some modified algorithms (denoted as controlled insertion algorithms) are also given in [12].

Some other constructive procedure which always sequence a chosen job on the last position such that the completion to a makespan optimal sequence is always possible are given in [12]. However, it turns out that the insertion algorithm has obtained the best results among these constructive algorithms.

For the quality of a local search algorithm, the choice of a suitable neighbourhood is of significant importance. First we give some neighbourhoods for a permutation problem, where the set of feasible solutions is given by the the set of permutions of n jobs. The neighbourhoods are described by the set $N(p)$ of neighbours of a sequence p.

Shift(SH): In a permutation $p = (p(1), p(2), \ldots, p(n))$, select an arbitrary element $p(i)$, and shift it to a smaller position j, $j < i$, or to a larger position k, $k > i$. Thus, we have $|N(p)| = (n-1)^2$. In some applications this neighbourhood is used in a specialized version, where only right or left shifts of an arbitrary job are allowed for the generation of a neighbour. The shift neighbourhood is sometimes also referred to as *insert* neighbourhood.

Pairwise interchange (PI): In permutation p, select two arbitrary jobs $p(i)$ and $p(j)$ $(i \neq j)$ and interchange them. The set $N(p)$ contains $\binom{n}{2}$ elements. Sometimes this neighbourhood is referred to as *swap* neighbourhood.

Adjacent pairwise interchange (API): This is a special case of both the shift and the pairwise interchange neighbourhood. In permutation p, two adjacent jobs $p(i)$ and $p(i+1)$ $(1 \leq i \leq n-1)$ are interchanged to generate a neighbour p'. Thus, we have $|N(p)| = n - 1$.

k-Pairwise interchange (k-PI): This is a generalization of the pairwise interchange neighbourhood. A neighbour is generated from p by performing at most k consecutive pairwise interchanges.

(k_1, k_2)-Adjacent pairwise interchange neighbourhood $(k_1, k_2 - API)$: In this neighbourhood, a neighbour is generated from p by performing k successive adjacent pairwise interchanges, where $k_1 \leq k \leq k_2$.

A neighbourhood structure may be represented by a directed, an undirected or a mixed graph. The set of vertices is the set of feasible solutions M, and there is an arc from $p \in M$ to each neighbour $p' \in N(p)$. In the case of $p' \in N(p)$ and $p \in N(p')$, we replace both arcs by an undirected edge. We denote the graphs describing the above neighbourhoods as $G(SH), G(PI), G(API), G(k-PI)$ and $G(k_1, k_2 - API)$, respectively. Obviously, all these graphs are undirected, i.e. these neighbourhoods are symmetric. Note that the graphs representing the right and left shift neighbourhoods are directed.

For the problem under consideration, only a subset of the permutations of n jobs is feasible. In this case, the set of neighbours $N(p)$ of a feasible sequence in the shift neighbourhood is the set of feasible sequences that can be obtained by a shift of a job. The next theorem gives an important property for the API neighbourhood since it guarantees that a local search algorithm may reach a global optimum from an arbitrary starting solution.

Theorem 1 *Let M be the set of feasible sequences for a problem $F2//F_h(C2/C_{max})$. Then the graph $G(API)$ defined on M is strongly connected.*

To have a fair comparison of the individual local search algorithms, we used as a stopping criterion the number of generated solutions. All experiments have been performed with a bad initial solution (by applying Johnson's algorithm) and with a good starting solution (by applying algorithm *INS*). We included the different neighbourhoods mentioned above into a computational study. As far as metaheuristics are concerned, we included and tested the following types:

Simulated annealing: We applied a geometric cooling scheme and investigated the choice of a suitable initial temperature and the choice of a suitable number of iterations with a constant temperature. Moreover, we investigated whether the inclusion of dominance criteria given in [12] improved the results.

Threshold accepting: It is a deterministic variant of simulated annealing and has been originally proposed in [6]. For decreasing threshold schemes, we experimented with the initial threshold value and the number of generated solution with a constant threshold. In addition to such schemes, we also experimented with variable threshold schemes which allow both the increase and the decrease of the current threshold in dependence on the course of computations.

Tabu search: Tabu search was proposed in its present form by Glover [9]. We investigated the performance of tabu search in dependence on the tabu list size and the number of generated neighbours in one iteration. For describing neighbours that are tabu we tested both a position and an order attribute. Additionally components of long term memory are also used, e.g. search intensification.

Multi-level heuristics: Multi-level heuristics are local search procedures which apply different neighbourhoods. Typically, when generating a neighbour, a neighbourhood with 'large' changes is used (high-level neighbourhood). Then another neighbourhood with 'small' changes (low-level neighbourhood) is applied within which local optimization is performed. Some variants of such algorithms are given in [1, 4]. We apply them in the following form. In one iteration we usually generate a number of high-level neighbours, restricted by a parameter h. Then we accept a high-level neighbour satisfying a certain acceptance condition, or we choose that high-level neighbour with the best objective function value if no neighbour satisfies the acceptance condition. From the chosen high-level neighbour we perform an iterative improvement procedure within the low-level neighbourhood. As low-level neighbourhood we have chosen the API neighbourhood and additonally we include a dominance criterion into the search.

Genetic algorithms: We have included a genetic algorithm from the literature into our comparison which was originally proposed for $C2 = \sum C_i$. We included a vector evaluated approach (see [17]. We adjusted the parameters (population size and number of generations) such that the number of solutions produced are the same as for the other types of algorithms described in this section.

3 Some results for flow shop

In this section, we summarize the findings of our empirical evaluation of various heuristics as follows:

In contrast to several scheduling problems with minimizing the makespan, for the $F2||F_h(C2/C_{\max})$ problem where $C2$ is any of the three objective functions considered, the use of the *pairwise interchange* neighbourhood leads to better results than the *shift* neighbourhood.

For threshold accepting, *variable threshold scheme* outperformed decreasing threshold schemes. For simulated annealing, a cooling scheme should be used which initially allows an increase in the objective function value by a certain percentage over the starting value with a fixed probability (instead of a fixed value of the increase). However, it could also be advantageous to apply variable 'cooling' schemes in simulated annealing.

For tabu search, the *position attribute* together with a small size of the tabu list (and thus a less restrictive tabu list) performed best. Random investigation of the pairwise interchange neighbourhood when the neighbourhood size is equal to the number of jobs worked best. Complete investigation of all nontabu neigbours in the adjacent pairwise interchange neighbourhood worked poorly. The inclusion of the intensification strategy did not improve the results which is probably due to the big neighbourhood size and the rather small number of iterations in the initial parameter tests ($100n$ generated solutions).

For the multi-level algorithms, choice of the *high-level neighbourhood* significantly influences the quality of results. For the problem under consideration, the use of a (nonadjacent) pairwise interchange or the use of at most two pairwise interchanges worked best. The application of several consecutive adjacent pairwise interchanges for generating a high-level neighbours did not yield competitive results. In the low-level neighbourhood, a local optimum need not necessarily be determined. It was important to include some structural properties in the investigation of the adjacent pairwise interchange neighbourhood (in our tests we included a dominance criterion to exclude nonpromising neighbours immediately). A ratio of 3:1 between generated high-level and low-level neighbours in one iteration is recommendable.

From the comparative study of the different type of algorithms, we found that for problems with $C2 = \sum C_i$ and $C2 = \sum w_i C_i$, the *multi-level algorithms* generally produced the best results. In this case, the use of high-level neighbourhoods $PI \setminus API$ and $2 - PI$ can be recommended, where the $2 - PI$ neighbourhood is even preferable when considering how often the best value has been obtained.

For the problems with $C2 = \sum w_i T_i$, we found that *simulated annealing* obtained the best results. Due to considerably larger differences between the initial and final objective function values, even iterative improvements algorithms worked better than expected, especially for problems with a large number of jobs. Refined algorithms such as multi-level probably lose too much time at the early stages while investigating several high-level neighbours in parallel and performing only small moves in the low-level neighbourhood. However, this observation confirms the need for the development of better constructive algorithms to start with better solutions.

The *genetic algorithm* converges rather slowly, and without a good initial population, the results are usually even worse than with algorithm *INS*. Due to this, a larger number of generated solutions is required to produce competitive results.

While the problems with the three different hierarchical criteria were similar as far as the set of feasible solutions and the neighbourhood relations are concerned, the comparative performance of heuristics is different for various problems. This was mainly due to the range of the differences in the objective function values of the starting and final solutions which influenced the choice of the suitable procedure. For problems with large differences in these values (i.e. $C2 = \sum w_i T_i$) and a large number of jobs, threshold accepting and sometimes even iterative improvement could lead to better intermediate results than simulated annealing (which performed well for this problem type) since in the initial phase unnecessary increases in the objective function values are avoided.

4 Constructive and iterative algorithms for open shop

We remind that an optimal schedule for problem $O2||C_{max}$ has the following makespan value:

$$C_{max} = \max\{T_1, T_2, a_r + b_r\},$$

where $T_1 = \sum_j a_j, T_2 = \sum_j b_j$ and r is the job with the largest sum of both processing times.

First, if $C_{max} = a_r + b_r$, then an optimal schedule for $C2 \in \{\sum C_i, \sum w_i C_i\}$ can be easily determined. We have to compare two schedules, namely the first one is obtained by scheduling job r first on M_1, and the remaining jobs on this machine are ordered according to the *WSPT* rule on this machine. The second one is the schedule when job r is first sequenced on M_2, and then the remaining jobs are ordered according to the *WSPT* rule on M_2 (notice that the first operations of all jobs except r can be sequenced in arbitrary order when processing job r on the other machine). The schedule with the better $C2$ value is obviously an optimal schedule.

Thus, in the following we consider only problems with $C_{max} = \max\{T_1, T_2\}$. Without loss of generality assume in the following $C_{max} = T_1$.

A 'bad' initial solution can be obtained by using the algorithm of Gonzalez and Sahni [10] which does not perform any optimization with respect to the secondary criterion. Similarly as for the flow shop problem, we can state an insertion algorithm as follows. Based on a chosen job list $(i(1), i(2), \ldots, i(n))$, we generate a schedule with the following structure: $p^1 = (k, p^*)$ and $p^2 = (p'^*, k, p''^*)$, where $p^* = (p'^*, p''^*)$. Only job k is processed first on M_1, the other jobs are processed first on M_2. This insertion algorithm generates a makespan optimal schedule as follows.

The insertion procedure chooses the first possible job in the job list which can be taken as the first job in p^1. For a candidate job k, this can be checked by determining Johnson's sequence p^J for the jobs in $N \setminus \{k\}$. If for schedule $p^1 = (k, p^J)$ and $p^2 = p^J$ the makespan value does not exceed $C_{max} = T_1$, the first job in p^1 has been determined.

Then we determine the first job in p^* and perform afterwards the insertion procedure within the partial sequence p^*. Assume that a partial schedule $p^1 = (k, j(1), \ldots, j(l))$ and $p^2 = (j(1), \ldots, j(l))$ has already been obtained (notice that the feasibility of a partial sequence can be easily checked as described in algorithm *INS* in Section 2 by appending the Johnson sequence of the unscheduled jobs at the end of both machines). Then we choose again the first unscheduled job l in the job list and insert it at all positions within the current partial sequence p^*. Among all partial schedules which can be extended to a makespan optimal sequence, we choose the partial schedule with the best objective function value with respect to criterion $C2$. More precisely, we consider the objective function value contributions of the jobs k, l and of those contained in p^*, where job k is put first at the last position in p^2 and then shifted to the left as much as possible without violating the minimal makespan in order to reduce its contribution to the objective function value.

We also tested restricted variants, where only the last h positions of the current partial sequence p^* are considered as a candidate for inserting the chosen job. Note also that there does not necessarily exist an optimal schedule having the structure of the generated schedule.

Assume now that a feasible schedule for the open job problem is given by the job sequences p^1 and p^2 describing the job orders on the machines M_1 and M_2 and partitioning the set of jobs into two sets J_1 and J_2 where J_i contains the jobs that are first processed on machine M_i. A neighbourhood N_1 for the 2-machine open shop problem can be described as follows:
1) Select two adjacent jobs in one of the sequences $p^j (j \in \{1, 2\})$ and interchange them without changing sets J_1 and J_2.
2) Select two jobs which are adjacent in both sequences p^1 and p^2 (if any), and interchange them without changing sets J_1 and J_2.
The following neighbour generations are only considered when the current starting solution has the structure $p^1 = (p^I, p^{II}), p^2 = (p^{II}, p^I)$, where p^I is the Johnson sequence of the jobs of set J_1 and p^{II} is the Johnson sequence of the jobs of set J_2:
3) Select a job in one of the sets J_i and insert it into the other set J_j, i.e. remove it from the corresponding sequence p^I or p^{II}, and insert it at an arbitrary position in the other sequence.
4) Select two jobs in different jets J_1 and J_2 and interchange them, i.e. remove the chosen jobs from the sequences p^I and p^{II} and insert them at an arbitrary position in the other sequence.
5) Interchange both sets J_1 and J_2, and consider the reverse job sequences on both machines, where the last job becomes the first one, the second to last job becomes the second job, and so on.

Whereas a neighbour generation according to 1) or 2) performs one or two adjacent pairwise interchanges but does not change the machine order of any job, a neighbour generation according to 3) changes the machine order of one job, and a generation according to 4) changes the machine order of exactly two jobs. Then we obtain the following result:

Theorem 2 *Let M be the set of feasible schedules for a problem $O2//F_h(C2/C_{max})$. Then the neighbourhood graph $G(N_1)$ defined on M is strongly connected.*

In [13], there have been tested several neighbourhoods where the set of neighbours contains in each

case the set given above. Thus, all considered neighbourhoods are represented by a strongly connected graph. The different neighbourhoods have been included into local search heuristics combined with different metaheuristics. Detailed results with the constructive and iterative algorithms for all three criteria $C2 \in \{\sum C_i, \sum w_i C_i, \sum w_i T_i\}$ have been given in [13].

References

[1] Brucker, P., Hurink, J., and Werner, F., Improving Local Search Heuristics for some Scheduling Problems, *Discrete Applied Mathematics* 65, 1996, 87 - 107.

[2] Chen, C.L., and Bulfin, R. L., Complexity results for multi-machine multi-criteria scheduling problems, *Proceedings of the Third Industrial Engineering Research Conference*, pp. 662-665 (1994).

[3] Chen, C.L., and Vempati, V.S., *Multicriteria flow shop scheduling: minimizing total flow time and maximum completion time*, Working paper, Department of Industrial Engineering, Mississippi State University (1992).

[4] Danneberg, D., Tautenhahn, T., Werner, F., A comparison of heuristic algorithms for flow shop scheduling problems with setup times and limited batch size, *Preprint 52/97*, Otto-von-Guericke-Universität Magdeburg, 1997.

[5] Dileepan, P., and Sen, T., Bicriterion Static Scheduling Research for a Single Machine, *OMEGA*, Vol. 16, No. 1, 1988, 53 – 59.

[6] Dueck, G., and Scheuer, T., Threshold Accepting: a General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *J. Comp. Physics* 90, 1990, 161 – 175.

[7] Fry, T. D., Armstrong, R. D., and Lewis, H., A Framework for Single Machine Multiple Objective Sequencing Research, *OMEGA*, Vol. 17, No. 6, pp. 595-607 (1989).

[8] Garey, M.R., Johnson, D.S., and Sethi, R., The complexity of flow shop and job shop scheduling, *Mathematics of Operations Research*, Vol. 1, 117-129 (1976).

[9] Glover, F., Tabu search, part I, *ORSA Journal on Computing*, Vol. 1, 190-206 (1989).

[10] Gonzalez, S., and Sahni, T.: Open Shop Scheduling to Minimize Finish Time, J. Assoc. of Comput. Mach. 23, 1976, 665 – 679.

[11] Gupta, J.N.D., Hennig, K., and Werner, F.: Local Search Heuristics for Two-Stage Flow Shop Problems with Secondary Criterion, Otto-von-Guericke-Universität Magdeburg, FMA, *Preprint 1/99*, 1999.

[12] Gupta, J.N.D., Neppalli, V.R., and Werner, F.: Minimizing Total Flow Time in a 2-Machine Flowshop Problem with Minimum Makespan, Otto-von-Guericke-Universität, FMA, *Preprint 21/97*, Magdeburg, 1997.

[13] Gupta, J.N.D., Wulkenhaar, G., and Werner, F.: Heuristic Algorithms for 2-Machine Open Shop Problems with Secondary Criterion, Otto-von-Guericke-Universität Magdeburg, FMA, Working paper, 1999.

[14] Johnson S. M., Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included, *Naval Research Logistics Quarterly*, 1, 1954, 61-68.

[15] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B., Sequencing and scheduling: algorithms and complexity. In *Logistics of Production and Inventory*. (S. C. Graves, A. H. G. Rinnooy Kan and P. H. Zipkin, Eds) pp. 445-522, North Holland, Amsterdam, Netherlands (1995).

[16] Nagar, A., Haddock, J., and Heragu, S., Multiple and Bicriteria Scheduling: A Literature Survey, *European Journal of Operational Research*, Vol. 81, 1995, 88 – 104.

[17] Nepalli, V.R., Chen, C.-L., and Gupta, J.N.D., Genetic Algorithms for the Two-Stage Bicriteria Flow Shop Problem, *European Journal of Operational Research*, Vol. 95, 1996, 356 – 373.

[18] Panwalkar, S. S., Dudek, R. K., and Smith, M. L., Sequencing research and the industrial scheduling problem, in *Symposium on the Theory of Scheduling and its application*, Elmaghraby, S. E. edition, Springer, New York (1973).

[19] Rajendran C., Two-Stage Flow Shop Scheduling Problem with Bicriteria, *Journal of the Operational Research Society*, 43, No. 9, 1992, 871 – 884.