New Lower Bounds for Flow Shop Scheduling

A. Gharbi, A. Mahjoubi

***Abstract*—**The flow shop scheduling problem has become one of the most intensively investigated topics in scheduling theory. In this paper, we propose improved branch-and-bound-based lower bounds by taking advantage of the interdependence of the relaxed sub-problems. The proposed approach is assessed on two flow shop variants, namely the makespan minimization problem and the total completion time minimization problem. Preliminary computational experiments provide promising results of our improved bounds with respect to their basic versions.

***Keywords*—**Branch-and-Bound, completion time, flow shop, lower bounds, makespan, scheduling.

I. Introduction

THE flow shop scheduling problem is one of the hardest combinational optimization problems.It can be described as follows: A set J of n jobs, available at time zero, has to be processed in a shop with m machines. Each job is processed on machines M_1, M_2, …, M_m in that order. No machine can process more than one job at a time and no job preemptions are allowed. The schedule with the same job ordering on every machine is called a *permutation* schedule. The goal is to find a sequence of jobs that minimizes some criterion. Most attention has been devoted to makespan and/or total completion time minimization. The practical implications of both criteria are obvious: minimization of the makespan leads to the minimization of the total production run, and minimization of the total completion time leads to the rapid turn-around of jobs [16].

The objective of this paper is to propose new lower bounds for two variants of the permutation flow shop problem, namely the makespan minimization problem ($F|prmu|C_{max}$), and the total completion time minimization problem ($F|prmu|\sum C_j$). Both variants have been proved NP-hard if the number of machines is greater than two [5]. Many efforts have been developed to obtain optimal or near-optimal sequences, including dynamic programming [9], integer linear programming [2,24], branch-and-bound algorithms [1,3,8,11,12,15,17,21], and heuristics [7,18,22,23,27].

The remainder of the paper is structured as follows. Section II describes the framework of our proposed lower bound. Section III provides the results of our computational study.

A. Gharbi is with the Industrial Engineering Department, College of Engineering, King Saud University, PO Box 800, 11421 Riyadh, Saudi Arabia. He is also a research fellow with BEM Management School Bordeaux, France (phone:966-1-4676829; fax: 966-1-4678657; e-mail: a.gharbi@ksu.edu.sa).

A. Mahjoubi is with High Institute of Management of Tunis, Bardo, Tunisia (email: mahjoubi.amira94@yahoo.fr).

II. Framework of the Proposed Lower Bound

The quality of the lower bound is one of the most critical components of any branch and bound algorithm. Several lower bounds have been proposed in the literature for the m-machine permutation flow shop problem. The basic idea is to compute lower bounds by relaxing in different ways the constraints of this problem.

Firstly, our bounding scheme is based on the fact that if infeasibility can be stated for a relaxed problem, then it can be obviously deduced for the original one. The main steps of our procedure can be therefore described as follows:

Step 1:

Transform the m-machine permutation flow shop problem into K sub-problems. Let T denote a valid starting lower bound. Set $k = 1$.

Step 2:

Solve the feasibility sub-problem P_k using a branch-and-bound algorithm.

If there is no feasible solution with objective value less than or equal to T

Then Set $T = T + 1$and Go to Step 2.

Else

If $k < K$, then Set $k = k + 1$ and Go to Step 2.

Else Stop and Return T.

Actually, the best performing lower bounds of the literature are those based on solving the relaxed sub-problems using branch-and-bound procedures. However, a major weakness of these lower bounds consists in assuming the sub-problems as totally independent from each others. Our main contribution is to take advantage of the *inter-dependence* of the sub-problems. Indeed, if a sub-sequence of the search tree is proven not to yield feasibility for sub-problem P_k, then it will not yield any feasibility for the original problem. Therefore, there is no need to explore this sub-sequence for any other sub-problem. For that purpose, in case of feasibility, the search tree that has been developed for P_k is stored and transferred to P_{k+1} to start with. In this way, all efforts that have been made in pruning unpromising nodes will be saved. An immediate result is that some potentially feasible solutions for P_{k+1} would not be feasible any more since they have been eliminated during the tree search of previous sub-problems. Consequently, the value of the lower bound will be increased. A new search tree is only created if the addressed sub-problem is proven to be infeasible.

In the following, we provide the details of the branch-and-bound algorithm that is implemented in Step2.

A. Notation

For each node *N* of the search tree, we associate the following data:

Notation		
$\bar{J}$	=	the set of unscheduled jobs.
*ub**	=	the current best upper bound of the optimal makespan.
σ_1	=	a partial sequence of n_1 jobs scheduled on the top of the global sequence.
σ_2	=	a partial sequence of n_2 ($n_2 \leq n - n_1$) jobs scheduled on the bottom of the global sequence.
lev (*N*)	=	level of the node *N* (lev (*N*) = $n_1 + n_2$).
LB (*N*)	=	a lower bound of the optimal objective value of a sequence of the form ($\sigma_1 J \sigma_2$).
UB (*N*)	=	an upper bound of the optimal objective value of a sequence of the form ($\sigma_1 J \sigma_2$).
N_p	=	the parental node of *N*.
Ω (N_p)	=	the set of candidate nodes descendant of N_p.
N_c	=	the closest unbranched node to *N*, such that $LB(N_c) < ub^*$.

B. Search strategy and branching scheme

To select the node to be branched, we implemented the Depth First Search Strategy. This strategy consists in branching on the node which has the largest level among the created nodes.

The branching rule adapted in our branch and bound algorithm has been introduced by Potts [20], and has been used by Carlier and Rebaï [3] and Haouari and Ladhari [12,13]. This branching will *alternatively* append an unscheduled job to σ_1 and to σ_2. The first branching will sequence a job in the first available position of σ_1, while the second branching sequences a job in the last available position of σ_2.

C. Synthesis of the branch and bound algorithm

Step 1: Initialization (creation of the root node N_0)

- $\bar{J} = J$; $\sigma_1 = \emptyset$; $\sigma_2 = \emptyset$; lev (N_0) = 0.
- Compute LB (N_0) and UB (N_0)=T+1.
- if(LB (N_0) = UB (N_0) then STOP (Optimal schedule).
  Otherwise, set $N_p = N_0$.
- ub* = UB (N_0).

Step 2: Branching of node N_p

- For all $j \in \bar{J}$ Do
  Begin (Creation of node N descendant of N_p)
  lev (N) = lev (N_p) + 1
  if lev (N) = n – 1, then N corresponds to a feasible sequence π.
  if C_{max}< ub*, then update ub* and go to Step 4.
  if lev (N) is odd, then
  $\sigma_1 = (\sigma_1 j)$ and update $C^i_{\sigma 1}$(i = 1,2)
  otherwise
  $\sigma_2 = (j\ \sigma_2)$ and update $C^i_{\sigma 2}$(i = 1,2)
  Compute LB (N) and UB (N)
  if UB (N)<ub*, then ub*=UB (N) and update Ω (N_p).
  ifUB (N) = LB (N_p), then go to Step 3.
  ifLB (N) < ub*, then $\Omega(N_p) = \Omega(N_p) \cup \{N\}$
  set Ω (N) = Ø.

Step 3: Node selection

- if $\Omega(N_p) = \emptyset$,

then go to Step 4.
otherwise, select a node N $\in \Omega(N_p)$ which has the biggest level and go to Step 2.

Step 4: Backtracking

- if N_c exists,

then set Np = Nc and go to Step 2.
otherwise, the sequence with makespan equal to ub*is optimal.

D. Implementation for $F|prmu|C_{max}$

For the $F|prmu|C_{max}$, the relaxed sub-problem is a two-machine flow shop with release dates, time lags, and delivery times (denoted by $F2|r_j, l_j, q_j, prmu|C_{max}$). In order to obtain such sub-problem, the capacity of all machines is relaxed except for a pair of selected machines. The total number of obtained sub-problems is $K = \frac{m(m-1)}{2}$.

The initial lower bound that is used in our algorithm is that of Haouari and Ladhari [12]. It can be described as follows: Let $C^{k\,l}_{max}(J)$ denote the optimal makespan of the $F2|rj, lj, qj, prmu|C_{max}$ defined on the job set *J* and machine pair (M_k, M_l); $1 \leq k < l \leq m$. Therefore, $LB_0 = \max_{1 \leq k < l \leq m} C^{k\,l}_{max}(J)$.

E. Implementation for $F|prmu|\sum C_j$

For the $F|prmu|\sum C_j$, the relaxed sub-problem is a single-machine flow shop with release dates (denoted by $1|r_j|\sum C_j$). In order to obtain such sub-problem, the capacity of all machines is relaxed except for one selected machine. The release date is defined by $r_j = \sum_{i=1}^{k-1} p_{ij}$ for k=2,...,m and $r_j = 0$ for $k = 1$. Actually, a delivery time $q_j = \sum_{i=k+1}^{m} p_{ij}$ is obtained from this relaxation but is ignored since it is a constant value that does not affect the objective function. The total number of obtained sub-problems is $K = m$.

The initial lower bound is obtained by the branch and bound algorithm proposed by T'Kindt et al. [26] together with the lower bound of Della Corce and T'Kindt [4].

III. Preliminary Computational Results

All our algorithms have been coded in C and compiled with Microsoft Visual C++ (2010). All the computational experiments were carried out on a Pentium Dual-Core with 2.00 Ghz PC and 4.00 GB RAM.

In order to assess the empirical performance of our lower bound on $F|prmu|C_{max}$, we considered the benchmark problem sets of Taillard [25]. A total number of 80 instances has been selected with sizes ranging from 20 jobs-5 machines to 100 jobs-10 machines.

The obtained results are reported in Table I. For each instance we provide:

- ✓ LB_0 = the first valid lower bound value.
- ✓ LB = the new lower bound value.
- ✓ Gap = the gap between LB and LB_0.

- ✓ UB = the best upper bound in the literature.
- ✓ Time = CPU time of LB (in seconds).

From Table I, we observe that the new lower bound LB improves LB_0 in 75% of the instances while requiring a relatively moderate CPU time.

TABLE I
PRELIMINARY COMPUTATIONAL RESULTS FOR $F|prmu|C_{max}$

n x m	Instance	LB_0	LB	Gap	UB	Time
20x5	Tail001	1278	1278	0	1278	~ 0
	Tail002	1355	1355	0	1359	1
	Tail003	1073	1074	1	1081	~ 0
	Tail004	1268	1270	2	1293	~ 0
	Tail005	1217	1217	0	1235	3
	Tail006	1193	1193	0	1195	23
	Tail007	1234	1234	0	1195	~ 0
	Tail008	1190	1191	1	1206	~ 0
	Tail009	1208	1209	1	1230	~ 0
	Tail010	1085	1088	3	1108	~ 0
20x10	Tail001	1494	1501	7	1582	142
	Tail002	1566	1568	2	1659	1
	Tail003	1451	1453	2	1496	~ 0
	Tail004	1337	1337	0	1377	~ 0
	Tail005	1344	1347	3	1419	1
	Tail006	1332	1334	2	1397	~ 0
	Tail007	1427	1427	0	1484	~ 0
	Tail008	1380	1384	4	1538	122
	Tail009	1534	1534	0	1593	~ 0
	Tail010	1496	1496	0	1591	~ 0
20x20	Tail001	1996	2006	10	2297	42
	Tail002	1815	1824	9	2099	2
	Tail003	2008	2011	3	2326	51
	Tail004	1909	1929	20	2223	6
	Tail005	2048	2066	18	2291	26
	Tail006	1886	1875	9	2226	2
	Tail007	2067	2080	13	2273	205
	Tail008	1880	1889	9	2200	1
	Tail009	1915	1921	6	2237	12
	Tail010	1955	1962	7	2178	4
50x5	Tail001	2724	2724	0	2724	~ 0
	Tail002	2832	2833	1	2834	40
	Tail003	2600	2602	2	2621	~ 0
	Tail004	2740	2741	1	2751	~ 0
	Tail005	2854	2854	0	2863	~ 0
	Tail006	2824	2825	1	2829	63
	Tail007	2724	2724	0	2725	24
	Tail008	2683	2683	0	2683	~ 0
	Tail009	2532	2540	8	2552	6
	Tail010	2782	2782	0	2782	~ 0
50x10	Tail001	2951	2954	3	2991	25
	Tail002	2828	2831	3	2867	1
	Tail003	2808	2825	17	2839	2
	Tail004	3026	3030	4	3063	2
	Tail005	2916	2918	2	2976	1
	Tail006	2961	2963	2	3006	1
	Tail007	3063	3063	0	3093	1
	Tail008	3006	3006	0	3037	6
	Tail009	2795	2801	6	2897	61
	Tail010	3046	3048	2	3065	1
50x20	Tail001	3672	3676	4	3850	65
	Tail002	3545	3551	6	3704	6940
	Tail003	3486	3502	16	3640	3820
	Tail004	3449	3510	11	3723	1251
	Tail005	3489	3491	2	3611	1284
	Tail006	3467	3486	19	3681	253
	Tail007	3427	3445	18	3704	~ 0
	Tail008	3451	3456	5	3691	6
	Tail009	3457	3464	7	3743	7
	Tail010	3556	3567	11	3756	4125
100x5	Tail001	5493	5493	0	5493	1
	Tail002	5243	5244	1	5268	13
	Tail003	5168	5169	1	5175	1
	Tail004	4996	4999	3	5014	1
	Tail005	5244	5246	2	5250	~ 0
	Tail006	5128	5128	0	5135	1
	Tail007	5222	5224	2	5246	1
	Tail008	5073	5074	1	5094	10
	Tail009	5448	5448	0	5448	2
	Tail010	5313	5313	0	5322	~ 0
100x10	Tail001	5759	5768	9	5770	12
	Tail002	5345	5347	2	5349	22
	Tail003	5654	5662	8	5676	5
	Tail004	5732	5740	8	5781	29
	Tail005	5438	5445	7	5467	6
	Tail006	5292	5303	11	5303	113
	Tail007	5535	5539	4	5595	-
	Tail008	5598	5601	3	5617	4
	Tail009	5836	5836	0	5871	9850
	Tail010	5835	5836	1	5845	30

The considered test bed for $F|prmu|\sum C_j$ has been randomly generated in the following way. The number of jobs n has

been chosen equal to 10, 15, 20, 25 jobs. The number of machines has been equal to 3 and 5 machines. For each (n,m) combination, 10 instances have been generated, resulting in a total number of 80 instances. The processing time has been randomly generated using the uniform discrete distribution on [1, 100].

Table II summarizes the obtained results. For each instance we provide:

- ✓ LB_0 = the first valid lower bound value.
- ✓ LB = the new lower bound value.
- ✓ Gap = the gap between LB and LB_0.
- ✓ Time = CPU time of LB (in seconds).

The results displayed by Table II show that our new lower bound outperforms LB_0 in 63%. The required CPU time is negligible (~0) for the most of the instances of the data set.

TABLE II
COMPUTATIONAL RESULTS FOR $F|prmu|\sum C_j$

n x m	LB_0	LB	Gap	Time
10 x 3	2949	2950	1	~ 0
	2360	2363	3	1
	2872	2873	1	~ 0
	2944	2944	0	~ 0
	2075	2075	0	~ 0
	3684	3689	5	~ 0
	3371	3373	2	~ 0
	2649	2652	3	1
	1710	1710	0	~ 0
	2784	2784	0	~ 0
10 x 5	3446	3451	5	~ 0
	3697	3699	2	~ 0
	2951	2953	2	~ 0
	3689	3689	0	~ 0
	3484	3485	1	~ 0
	3291	3292	1	6
	4061	4061	0	~ 0
	3972	3972	0	~ 0
	2593	3597	4	~ 0
	2894	2894	0	~ 0
15 x 3	5602	5602	0	~ 0
	5324	5324	0	~ 0
	4742	4744	2	~ 0
	5883	5887	4	~ 0
	5226	5227	1	~ 0
	4626	4626	0	~ 0
	5092	5093	1	~ 0
	6322	6323	1	~ 0
	5973	5973	0	~ 0
	5029	5032	3	~ 0
15 x 5	7085	7088	3	~ 0
	5581	5581	0	~ 0
	6312	6314	2	~ 0
	6440	6422	2	~ 0
	8529	8536	7	~ 0
	6495	6495	0	~ 0
	5993	5998	5	~ 0
	8403	8404	1	~ 0
	8441	8441	0	2
	8619	8620	1	2
20 x 3	8018	8019	1	~ 0
	8888	8888	0	1
	11188	11188	0	~ 0
	9184	9188	4	1
	8024	8205	1	~ 0
	10415	10417	2	~ 0
	9729	9729	0	~ 0
	8949	8949	0	~ 0
	8730	8731	1	~ 0
	11138	11138	0	~ 0
20 x 5	10635	10636	1	~ 0
	11343	11343	0	~ 0
	10441	10442	1	~ 0
	11813	11813	0	2
	9145	9147	2	~ 0
	13380	13381	1	~ 0
	12691	12692	1	~ 0
	10294	10296	2	~ 0
	9604	9610	4	1
	9643	9643	0	1
25 x 3	13363	13364	1	~ 0
	12464	12464	0	~ 0
	14025	14027	2	5
	13418	13421	3	~ 0
	15839	15840	1	~ 0
	13739	13740	1	2
	13155	13155	0	~ 0
	14012	14015	3	~ 0
	15598	15599	1	~ 0
	16882	16883	1	~ 0
25 x 5	14718	14718	0	1
	15025	15025	0	~ 0
	17355	17356	1	~ 0
	13866	13873	7	~ 0
	14836	14837	1	8
	19083	19084	1	1
	16139	16139	0	~ 0
	13383	13386	3	~ 0
	14675	14684	9	~ 0
	15421	15422	1	~ 0

ACKNOWLEDGMENT

This paper is supported by the National Plan for Science & Technology (NPST) program at King Saud University (project number 11-MAT-1491-2).

REFERENCES

[1] S. Ashour. “An Experimental Investigation and Comparative Evaluation of flow shop Scheduling Techniques”, Operations Research, 1970, Vol. 18, pp.541–548.

[2] K. R. Baker. “Introduction to Sequencing and Scheduling”, Wiley Publishing, New York, 1974.

[3] J. Carlier, M. I. Rebai. “Two branch-and-bound algorithms for the permutation Fow shop problem “. European Journal of Operational Research, 1996, 90:238–51.

[4] F. Della Croce and V. T’kindt. “Improving the preemptive bound for the one machine dynamic total completion time scheduling problem”. Operations Research Letters, 2003, Vol. 31, pp.142–148.

[5] M. R. Garey, D. S. Johnson, and R. Sethi, R. “The complexity of flow shop and jobshop scheduling”, Mathematics of Operations Research, Vol. 1, 1976, pp.117–129.

[6] T. Gonzalez and S. Sahni. “Flow shop and job shop schedules”, Operation Research, 1978, 26; 36-53.

[7] J.N.D. Gupta, “Heuristic algorithms for multistage flow shop scheduling problem, AIIE Transactions 4, 1972, 11–18.

[8] W. Han and P. Dejax. “An efficient heuristic based on machine workload for the flow shop scheduling problem with setup and removal”, Annals of Operations Research, 1994, Vol 50(1), pp.263–279.

[9] M. Held and R. M. Karp. “A dynamic programming approach to sequencing problems”, Journal of the Society for Industrial and Applied Mathematics, 1962, Vol. 10, pp.196–210.

[10] J.C. Ho. “Flow shop sequencing with mean flow time objective”,European Journal of Operational Research, 1995, 81; 571–578.

[11] E. Ignall and L. Schrage. “Application of the branch and bound technique to some flow shop scheduling problems”, Operations Research, 1965, Vol. 13, pp.400–412.

[12] T. Ladhari and M. Haouari. “A computational study of the permutation Fow shop problem based on a tight lower bound”, Computers &Operations Research, 2005, 32: 1831–1847.

[13] T. Ladhari andM. Haouari. “Minimising maximum lateness in a two-machine Fow shop”. Journal of the Operational Research Society 51, 2000, 1100±1106.

[14] B. J. Lageweg, J. K. Lenstra and K. Rinnooy. “A general bounding scheme for the permutation Fow-shop problem”. Operations Research,1978, 26:53–67.].

[15] L. Lomnicki, L. “A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem”, Operational Research Quarterly, 1965, Vol. 16, pp.89–100.
[16] B. L. MacCarthy and J. Liu. “Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling”. International journal of production research, 1993, 31: 59-79.
[17] G. B. McMahon and P. G. Burton, P.G. “Flow shop scheduling with the branch-and-bound method”, Operations Research, 1967, Vol. 15, pp.473–481.
[18] S. Miyazaki, N. Nishiyama and F. Hashimoto. “An adjacent pairwise approach to the mean flow time scheduling problem”, Journal of the Operations Research Society of Japan, 1978, 21; 287–299.
[19] M. Pinedo. “Scheduling: theory, algorithms, and systems”. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[20] C. N. Potts. « An adaptive branching rule for the permutation Fow-shop problem ». European Journal of Operational Research, 1980, 5:19–25.
[21] C. N. Potts. “An adaptive branching rule for the permutation flow-shop problem”, European Journal of Operational Research, 1980, Vol. 5, pp.19–44.
[22] C. Rajendran. “Heuristic algorithm for scheduling in a flow shop to minimize total flow time”, International Journal of Production Economics, 1993, 29; 65–73.
[23] C. Rajendran and D. Chaudhuri. “An efficient heuristic approach to the scheduling of jobs in a flowshop”, European Journal of Operational Research, 1991, 61; 318–325.
[24] E. F. Stafford and F. T. Tseng. “On the Srikar–Ghosh MILP model for the N × M SDST flow shop problem”, International Journal of Production Research, 1990, Vol. 28, pp.1817–1830.
[25] E. Taillard. “Benchmarks for basic scheduling problems”. European Journal of Operational Research, 1993, 64:278–85.
[26] V.T’kindt, F.Della Croce and C. Esswein. “Revisiting Branch and Bound Search Strategies for Machine Scheduling Problems”. Journal of Scheduling, 2004, Vol. 7(6), pp.429–440.
[27] D.S. Woo and H.S. Yim. “A heuristic algorithm for mean flow time objective in flow shop scheduling”, Computers and Operations Research, 1998, 25; 175–182.