Applying machine based decomposition in 2-machine Flow Shops

Saral Mukherjee* and A. K. Chatterjee[1]

Indian Institute of Management Ahmedabad, India,

Indian Institute of Management Calcutta, India

Abstract

The Shifting Bottleneck (SB) heuristic is among the most successful approximation methods for solving the Job Shop problem. It is essentially a machine based decomposition procedure where a series of One Machine Sequencing Problems (OMSPs) are solved. However, such a procedure has been reported to be highly ineffective for the Flow Shop problems (Jain and Meeran 2002). In particular, we show that for the 2-machine Flow Shop problem, the SB heurisitc will deliver the optimal solution in only a small number of instances. We examine the reason behind the failure of the machine based decomposition method for the Flow Shop. An optimal machine based decomposition procedure is formulated for the 2-machine Flow Shop, the time complexity of which is worse than that of the celebrated Johnson's Rule. The contribution of the present study lies in showing that the same machine based decomposition procedures which are so successful in solving complex Job Shops can also be suitably modified to optimally solve the simpler Flow Shops.

Keywords: Shifting Bottleneck heuristic, Machine based decomposition, Johnson's Rule

1. Introduction

A Job Shop is a manufacturing scenario where a number of general-purpose machines are available for processing a variety of jobs. Each job consists of a number of operations,

* Corresponding author e-mail: saralm@iimahd.ernet.in
1 e-mail: ac@iimcal.ac.in

which must be processed according to a predefined sequence. Each operation can be processed on only one machine and the time required for processing is deterministic. While there are precedence constraints among operations belonging to the same job, there are no precedence constraints among operations belonging to different jobs. Preemption of operations is not allowed. Each machine can process only one operation at a time and at any instant no more than one operation of a particular job can be undergoing processing. The problem is to find the schedule, which minimizes the maximum completion time of all operations, also termed as makespan. The schedule can be either defined in terms of an exhaustive list of start times of all operations, or can be equivalently defined by the job sequence of each machine.

The Shifting Bottleneck heuristic proposed by Adams et al. (1988) has emerged as one of the most successful approximation methods for solving the general Job Shop. It uses a decomposition procedure that focuses on the *machines*, not the jobs. In contrast, a priority dispatching rule-based system looks for the *job/operation* with the highest priority (according to say SPT or FCFS rule etc.) among the *jobs/operations* available for scheduling on a machine at a scheduling instant. Similarly, a schedule is characterised as active, semi-active or non-delay based on particular characteristics of the *operations*. This focus on jobs and operations is distinctly different from the machine based decomposition procedure of the SB heuristic.

A Flow Shop can be viewed as a special case of a Job Shop where the processing sequence is same for all jobs. Flow Shops arise in real life situations whenever there is a material handling system like conveyor belt that feeds the machines, or in chemical process industries where job passing is not allowed. Following Conway (1967), we denote the n job 2 machine Flow Shop problem with minimisation of makespan objective as the $n/2/F/C_{max}$ problem. In his seminal paper in 1954, Johnson dealt with the problem of minimising makespan in the $n/2/F/C_{max}$ problem and the $n/3/F/C_{max}$ problem under some special cases. The optimal procedure of Johnson can be seen as a job-based procedure. Heuristics have been developed by several authors for the $n/m/F/C_{max}$ problem along similar lines, the focus being on characteristics of jobs. Since a Flow Shop is a particular case of the general Job Shop, it should be a natural candidate for the application of SB heuristic. However, machine based decomposition procedures have not yielded good results for the Flow Shop problem (Demirkol et. al 1997), (Jain and Meeran 2002). In particular, Jain and Meeran (2002) note that Shifting Bottleneck implementations of Demirkol et. al (1997) "have difficulty solving F shop problems". It is interesting to note that a successful solution procedure for the more general Job Shop problem is unable to provide good solutions to the more restricted Flow Shop problem. This is the inspiration for the current work. Can a machine based decomposition procedure obtain the known results for the "simplest" of the Flow Shop problems without taking any help from the established literature on dominance conditions?

In the following sections we first show that the SB heuristic will fail to find optimal results for a majority of problem instances for the $n/2/F/C_{\max}$ problem and then provide an optimal solution procedure for the same. Section 2 presents a brief survey of literature followed by some preliminary results in section 3. In section 4, we present a modification of the Schrage scheduling heuristic and illustrate the workings with the help of an example. This modification is an essential part of the DSP algorithm, presented in section 5, for solving the $n/2/F/C_{\max}$ problem. In section 6, we describe several characteristics of the schedule returned by the DSP algorithm, which are helpful in proving the optimality of the DSP algorithm in section 7. The convergence and complexity results for the DSP algorithm are presented in section 8. Finally, we conclude with section 9.

2. Literature Review

The solution approaches to Job Shop scheduling can be classified as either exact or approximation methods. Branch and Bound (B&B) approaches have been the most successful among exact methods, the other approaches which have been tried out with limited success being Lagrangian relaxation based (Fisher et al. 1983). Prominent B&B applications include those by Carlier and Pinson (1989) and Brucker et al. (1994).

The most widely used approximation method has been the application of a huge number of priority dispatching rules. The survey paper of Panwalkar and Iskander (1977) describes 113 such rules. Perhaps the most well known approximation algorithm in scheduling research has been the Shifting Bottleneck (SB) heuristic of Adams et al. (1988). It sets up and solves a series of One Machine Sequencing Problems (OMSPs). Carlier (1982) provided a very efficient branch and bound solution procedure for solving the OMSP when all operations are independent. Dauzere-Peres and Lasserre (1993) incorporated delayed precedence constraints while solving the OMSP heuristically. Subsequently an optimal branch and bound procedure was provided by Balas et al. (1995). Computational studies of several variants of the SB heuristic have been reported by Holtsclaw and Uzsoy (1996) and Demirkol et al. (1997). The Generalised SB heuristic of Ramudhin and Marier (1996) implemented SB based approaches in diverse production scenarios. Other versions of this heuristic applied to different scheduling problems include Demirkol and Uzsoy (1998) for re-entrant Flow Shops with sequence dependent set up times, Pinedo and Singer (1996) for minimizing total weighted tardiness in Job Shops and Sun and Noble (1999) for minimizing the weighted sum of squared tardiness in a Job Shop with sequence dependent set up times.

A Flow Shop can be viewed as a special case of a Job Shop where the processing sequence is same for all jobs. Apart from providing optimal methods for the $n/2/F/C_{\max}$ problem and certain special cases of the $n/3/F/C_{\max}$ problem, Johnson (1954) showed that it is sufficient to consider only permutation schedules for flow shop cases whenever there are fewer than four machines. Subsequently, the research on Flow Shops concentrated on finding various dominance conditions for the $n/3/F/C_{\max}$ problem (Szwarc 1978). Prominent solution methods for the $n/m/F/C_{\max}$ problem include, among others, those by Campbell et al. (1970), Nawaz et al. (1983), Hundal and Rajagopal (1988) and Osman and Potts (1989). Compared to the amount of attention devoted to the permutation Flow Shop, comparatively less attention has been paid to the non-permutation Flow Shop. Since the SB heuristic has no in-built explicit mechanism to return permutation sequences, it seems to be a natural candidate for solving the non-permutation Flow Shop. Demirkol et al (1997) report the effect of different versions of the SB heuristic on Flow Shops. However, the results are not encouraging and substantial improvements in makespan are reported by the application of a multi level hybrid framework in Jain and Meeran (2002).

3. Preliminary results

The SB heuristic can be seen as a particular sequence of solving machine based decompositions. Starting with an empty schedule (one where none of the machines is sequenced), SB heuristic constructs a final schedule by fixing the sequence on each machine, one at a time. At each iteration a bottleneck machine, M_k, among those not yet sequenced, is identified and the sequence on this machine is fixed by the sequence S_k. Identification of bottleneck machine M_k and the sequence S_k are achieved by solving a certain one machine scheduling problem (OMSP). The OMSP is concerned with scheduling a set of operations on a machine so that the maximum lateness of any operation is minimized. Each operation O_{ik} has a release time r_{ik}, a processing time p_{ik} and a due date f_{ik}. Dauzere-Peres and Lasserre (1993) have shown that delayed precedence constraints could exist between operations to be processed on a particular machine in a Job Shop, when the OMSP is set up as part of the SB heuristic. We now prove that such a situation cannot arise in a Flow Shop. Since, in a m-machine Flow Shop, the sequence M_1-M_2-…-M_m of machines that a job needs for processing is fixed for all jobs; we define a machine M_j to be a upstream (downstream) machine from M_i if $i > j$ ($i < j$). It is apparent from the disjunctive graph representation of the Flow Shop that a conjunctive arc will always connect an operation on an upstream machine with another operation on the immediate downstream machine.

Lemma 1: Delayed Precedence Constraints (DPCs) cannot arise in an OMSP created for a machine in a Flow Shop

Proof: We denote by O_{ik} the operation of job J_i to be processed on machine M_k. The existence of DPCs would mean there exists at least two paths between operations O_{ik} and O_{jk} to be processed on the machine M_k. One path will consist entirely of disjunctive arcs joining operations to be performed on machine M_k. All the other paths will include at least one conjunctive arc. Hence, such a path will have an operation O_{rq} to be performed on a machine M_q which is both upstream (since O_{rq} is a predecessor of O_{jk}) and downstream (since O_{rq} is a successor of O_{ik}) from machine M_k. But a machine can not be both upstream and downstream from another machine in a Flow Shop. Hence the contradiction. ■

Lemma 1 implies that the procedure of Carlier (1982) is optimal for solving an OMSP created in a Flow Shop. Carlier's algorithm utilises the Schrage schedule to provide an initial solution and then branches by identifying a critical job. We give below an outline of the Schrage schedule created for the OMSP. The input to the Schrage schedule is an OMSP where for each operation O_{ik} of Job J_i to be scheduled on that machine M_k, the release time r_{ik}, processing time p_{ik} and the "tail" q_{ik} are known.

Schrage schedule for machine M_k

Let U be the set of operations already scheduled and U' the set of operations yet to be scheduled, t is the scheduling instant and I_k is the index set of all operations to be scheduled on M_k.

1. Set $t = \text{Min}_{i\in I}\, r_{ik}$; $U = \phi$ and $U' = I_k$.
2. At time t, schedule amongst the ready operations (i.e. operation O_{ik} such that $r_{ik} \leq t$, $i\in U'$), the operation O_{jk} with greatest q_{jk} (or any one in case of ties).
3. Set $U = U \cup \{j\}$ and $U' = U' \setminus \{j\}$. Set $t = \text{Max}\,(t + p_{jk}; \text{Min}_{i\in U'}\, r_{ik},)$. If $U' = \phi$. STOP. Else go to 2.

Note that in the situation where release dates are same for all jobs, the Schrage schedule results in the Earliest Due Date (EDD) sequence. However, for the case where all due dates are same, the Schrage schedule is difficult to characterise as ordering of operations is not uniquely defined when all operations have the same q_{jk} values and hence would depend on the actual implementation. In case the First Come First Served (FCFS) rule is utilised for choosing the next operation in Step 2, the Schrage heuristic is equivalent to scheduling the operations according to the non-decreasing order of their release times (equivalently, the FCFS rule). It is further known (Pinedo 1995) that the EDD rule is optimal for the OMSP when all release times are same. When all due dates are same, the optimality of the Schrage schedule and equivalently that of the FCFS rule can be derived from the branch and bound procedure of Carlier (1982).

The SB heuristic begins with none of the machines scheduled. At every step it sets up and solves an OMSP with minimisation of L_{max} objective for each unscheduled machine and then fixes the sequence for the machine with the highest L_{max} value. Since for the $n/2/F/C_{max}$ problem there are only two machines, let the machine with the highest L_{max} value be identified as M_{HL} and the other machine as M_{LL}.

Lemma 2: If the SB heuristic chooses sequence $S = (J_1,J_2,\ldots J_n)$ on machine M_{HL} then the sequence for the machine M_{LL} will also be S.

Proof: Without loss of generality, let machine M_2 be the machine with the highest L_{max} value and the sequence $S_2 = O_{12}$-O_{22}-…-O_{n2} is set on machine M_2. Since each job has only one operation to be performed on M_2, the sequence S_2 can also be equivalently represented by J_1-J_2-…-J_n. The disjunctive graph representation at this stage is shown in Fig. 1.

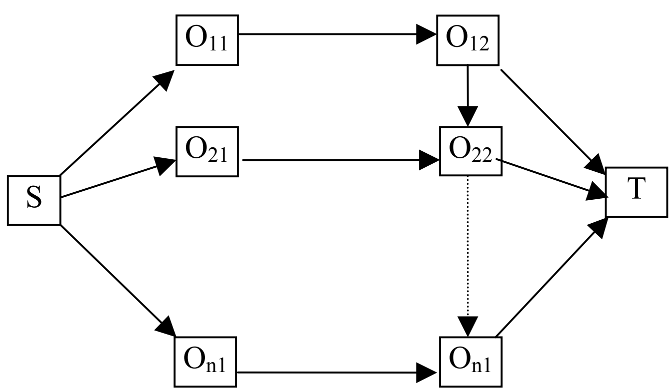

Fig. 1

Let C_{max} be the makespan value at this stage. We now consider the OMSP for machine M_1. The OMSP data is shown in Table 1.

Table 1: OMSP data for machine M_1

	r_{i1}	p_{i1}	q_{i1}
O_{11}	0	p_{11}	$p_{12}+p_{22}+\ldots+p_{n2}$
O_{12}	0	p_{21}	$p_{22}+p_{32}+\ldots+p_{n2}$
…	…	…	…
…	…	…	…
O_{n2}	0	p_{n1}	p_{n2}

Since all release dates are same, the optimal schedule is given by the EDD rule. Since processing times are nonnegative and the due date f_{ik} and q_{ik} values for operation O_{ik} are related by the equation $f_{ik} = C_{max} - q_{ik}$, the optimum sequence for the OMSP is given by $S_1 = J_1\text{-}J_2\text{-}\ldots\text{-}J_n$. Thus $S_1 = S_2$. A similar analysis can be done to show $S_1 = S_2$ when machine M_1 is the machine with highest L_{max} value, noting that the optimal schedule for M_2 would be given by the FCFS rule and all processing times are non negative. Hence the result follows. ■

Lemma 2 implies that once a sequence is set for any machine in a 2-machine Flow Shop, this sequence dictates the sequence on the other machine. It follows then that the reoptimisation step of the SB heuristic will fail to alter the sequence on any machine. Thus the result of applying the SB heuristic to 2-machine Flow Shops is either a sequence obtained by scheduling the operations on M_1 according to EDD rule or a sequence obtained by scheduling the operations on M_2 according to the FCFS rule.

The next lemma (Pinedo 1995) provides a characterisation of the optimal schedule. Partition the set of jobs J into two sets S_1 and S_2 such that S_1 contains all jobs with $p_{i1} < p_{i2}$, where p_{ij} represents the processing time of operation O_{ij} of Job J_i performed on machine M_j. Similarly, S_2 contains all jobs with $p_{i1} > p_{i2}$. The jobs with $p_{i1} = p_{i2}$ may be in either set. We denote by $SPT_1(S_1)\text{-}LPT_2(S_2)$ a schedule formed by arranging the jobs in S_1 according to the Shortest Processing Time (SPT) rule applied on p_{i1}, $i \in S_1$ and then the jobs in S_2 according to the Longest Processing Time (LPT) rule applied on p_{i2}, $i \in S_2$.

Lemma 3: Any $SPT_1(S_1)$-$LPT_2(S_2)$ schedule is optimal for $n/2/F/C_{max}$ problem. (Pinedo 1995)

Let $SPT_1(J)$ denote the schedule obtained by ordering all jobs in set J according to the shortest processing times on machine M_1. $LPT_2(J)$ is similarly defined. For the next theorem, we consider that the implementation of the SB heuristic incorporates the FCFS rule while choosing among ready operations in Step 2 of the Schrage heuristic.

Theorem 1: The SB heuristic will deliver the optimal schedule for the $n/2/F/C_{max}$ problem only if any one of the following conditions hold: (i) $S_1 = \phi$ and $M_{HL} = M_1$ (ii) $S_2 = \phi$ and $M_{HL} = M_2$ (iii) $SPT_1(S_2) = LPT_2(S_2)$ and $p_{i1} > p_{j1}$ $\forall i \in S_2$ and $\forall j \in S_1$ and $M_{HL} = M_2$ and (iv) $LPT_2(S_1) = SPT_1(S_1)$ and $p_{i2} > p_{j2}$ $\forall i \in S_1$ and $\forall j \in S_2$ and $M_{HL} = M_1$.

Proof: Let S be the sequence selected for the bottleneck machine in first step. Then using Lemma 2, the same sequence S will be selected for the other machine and there would be no further changes in the sequence on either machine during the subsequent steps of the SB heuristic. It is well known (Pinedo 1995) that the optimal sequence S for an OMSP with same release time for all jobs is given by the EDD rule. Since at this scheduling instant all machines are unscheduled, $q_{i1} = p_{i2}$ and $f_{i1} = C_{max} - q_{i1}$ for any operation O_{i1} to be performed on M_1. Thus sequencing of operations on M_1 according to EDD rule is equivalent to sequencing the operations according the $LPT_2(J)$ rule. Hence if the first bottleneck machine is M_1 then S is according to $LPT_2(J)$. Similarly, if the first bottleneck machine is M_2 then S is according to non-decreasing release times of operations on M_2, which is equivalent to applying the $SPT_1(J)$ rule. Hence the SB heuristic will result in either a $SPT_1(J)$ or an $LPT_2(J)$ schedule, depending on the machine identified as the bottleneck in the first step. However, as pointed out in Lemma 3, the optimum sequence for the $n/2/F/C_{max}$ problem has the structure $SPT_1(S_1)$-$LPT_2(S_2)$. Hence the SB heuristic will result in an optimal solution only when any one of the following conditions hold: (i) $S_1 = \phi$ and $M_{HL} = M_1$ (ii) $S_2 = \phi$ and $M_{HL} = M_2$ (iii) $SPT_1(S_2) = LPT_2(S_2)$ and $p_{i1} > p_{j1}$

$\forall i \in S_2$ and $\forall j \in S_1$ and $M_{HL} = M_2$ and (iv) $LPT_2(S_1) = SPT_1(S_1)$ and $p_{i2} > p_{j2}$ $\forall i \in S_1$ and $\forall j \in S_2$ and $M_{HL} = M_1$. ■

Let the processing time of an operation be chosen as one of {1, 2,…, N}. Then the total number of ways in which two numbers p_{i1} and p_{i2} can be chosen such that $p_{i1} \geq p_{i2}$ is given by $T = 1 + \ldots + N = N(N+1)/2$. Hence the probability that $p_{i1} \geq p_{i2}$, $\forall i \in J$ is $(N(N+1)/2N^2\}^n = ((N+1)/2N)^n \cong (1/2)^n$ for large N. Hence the probability that $S_1 = \phi$ is given by $(1/2)^n$. For large n, this probability is negligible. It can be shown similarly that the probability of $S_2 = \phi$ and hence that of occurrence of the second condition is negligible for large n.

For calculating the probability of occurrence of the third condition, let u = card(S_1) and v = card(S_2) be the cardinality of S_1 and S_2 respectively. We consider the case where, given the set S_1, the processing times $\{p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32} \ldots p_{v1}, p_{v2}\}$ of jobs belonging to S_2 are chosen such that (i) $p_{11} > p_{12}, p_{21} > p_{22}, \ldots, p_{v1} > p_{v2}$ and (ii) $p_{12} \geq p_{22} \geq p_{32} \ldots \geq p_{v2}$ and (iii) $p_{11} \leq p_{21} \leq p_{31} \ldots \leq p_{v1}$. Furthermore, let $k = \max_{i \in S_1} p_{i1}$. Thus the probability of occurrence of $SPT_1(S_2) = LPT_2(S_2)$ and $p_{i1} > p_{j1}$ $\forall i \in S_2$ and $\forall j \in S_1$ is given by

[Prob(card(S_1) = u)]*[Prob($p_{11} > p_{12}$)*Prob($p_{11} > k$)]*[Prob($p_{21} > p_{11}$)*Prob($p_{22} \leq p_{12}$)]*…*[Prob($p_{v1} > p_{(v-1)1}$)*Prob($p_{v2} \leq p_{(v-1)2}$)]

$$= [(\frac{1}{2})^u] \times [(\frac{1}{2}) \times \frac{N-k}{N})] \times [\frac{N-p_{11}}{N} \times \frac{p_{12}}{N}] \times [\frac{N-p_{21}}{N} \times \frac{p_{22}}{N}] \times \ldots\ldots \times [\frac{N - p_{(v-1)1}}{N} \times \frac{p_{(v-1)2}}{N}]$$

$$\leq [(\frac{1}{2})^u] \times [(\frac{1}{2}) \times \frac{N-p_{(v-1)1}}{N}] \times [\frac{N-p_{(v-1)1}}{N} \times \frac{p_{12}}{N}] \times [\frac{N-p_{(v-1)1}}{N} \times \frac{p_{12}}{N}] \times \ldots \times [\frac{N - p_{(v-1)1}}{N} \times \frac{p_{12}}{N}]$$

$$= (\frac{1}{2})^{u+1} \times (\frac{N-p_{(v-1)1}}{N})^v \times (\frac{p_{12}}{N})^{v-1}$$

$$\leq (\frac{1}{2})^{u+1} \times (\frac{N-p_{12}}{N})^v \times (\frac{p_{12}}{N})^{v-1}$$

$= (\frac{1}{2})^{u+1} \times (\frac{N - p_{12}}{N}) \times Y^{v-1}$ where $Y = \frac{N - p_{12}}{N} \times \frac{p_{12}}{N}$.

$\cong (\frac{1}{2})^{u+1} \times Y^{v-1}$ for large N.

$\leq (\frac{1}{2})^{n+v-1}$ since Y is maximised when $p_{12} = N/2$.

Hence for large n, the probability of occurrence of $SPT_1(S_2) = LPT_2(S_2)$ and $p_{i1} > p_{j1}$ $\forall i \in S_2$ and $\forall j \in S_1$ is negligible. A similar analysis can be done to show that the probability of occurrence of the fourth condition is also negligible. Thus the SB heuristic will fail to deliver the optimal solution in most instances of the $n/2/F/C_{max}$ problem.

We will now show that a slight modification of the SB heuristic will solve the $n/2/F/C_{max}$ problem optimally. For this we need to take advantage of the structural nature of the flow shop. We adapt Schrage schedule to this environment and propose a completely different branching scheme.

4. Modifications to the Schrage schedule

We use some structural properties of the Flow Shop to modify the Schrage scheduling heuristic. It is well known that for the $n/2/F/C_{max}$ problem it is sufficient to consider permutation sequences for optimality (Johnson 1954). Suppose we are interested in solving an OMSP for machine M_2. Lemma 2 indicates that fixing the operation sequence for M_2 automatically fixes the same sequence for M_1. Hence at any scheduling instant of the Schrage algorithm, we can take help of this additional information and dynamically update the release time data for the operations on M_2. This is primary inspiration for modifying the Schrage algorithm for the $n/2/F/C_{max}$ problem. We call this modified approach as the Dynamic Schrage heuristic. We select the operation O_{i2} with largest processing time p_{i2} in case of a tie.

For solving the $n/2/F/C_{max}$ problem, the steps involve setting up an OMSP for machine M_2 and then applying the Dynamic Schrage heuristic to schedule it. We could have as well chosen machine M_1, which would have necessitated defining a Dynamic Schrage heuristic for machine M_1, where instead of dynamically updating the release times, we would have dynamically updated the q values.

Dynamic Schrage heuristic for M_2 in $n/2/F/C_{max}$ problem

Let U be the set of operations already scheduled on machine M_2 and U' the set of operations yet to be scheduled, t is the scheduling instant and I_2 is the index set of all operations.

1. Set $t = \text{Min}_{i \in I}\, r_{i2}$; $U = \phi$ and $U' = I_2$.
2. At time t schedule amongst the ready operations (i.e. operation O_{i2} such that $r_{i2} \leq t$, $i \in U'$ and all predecessors of O_{i2} have been scheduled), the operation O_{j2} with greatest q_{j2} (or the operation with the greatest p_{j2} in case of ties).
3. Set $U = U \cup \{j\}$ and $U' = U' \setminus \{j\}$. Update release dates of all unscheduled operations i.e. set $r_{i2} = r_{i2} + p_{j1}$; $\forall i \in U'$. Set $t = \text{Max}\ (t + p_{j2};\ \text{Min}_{i \in U'}\, r_{i2})$. If $U' = \phi$, set the sequence returned on both the machines. STOP. Else go to 2.

Corresponding to the operation sequence returned by the Dynamic Schrage heuristic for M_2, the equivalent job sequence is identified and implemented on M_1 to obtain a complete schedule for the $n/2/F/C_{max}$ problem. As noted in Lemma 1, DPCs do not exist in Flow Shops. Still, we explicitly check for any predecessor operation in Step 2 as the DSP algorithm, which we propose in section 5, may constrain some jobs to succeed some other jobs.

Example 1: Consider a 4 job 2 machine flow shop problem with processing time data as given in Table 2. The OMSP data for machine M_2 is given in Table 3.

Table 2: Processing times for a 4 job 2 machine Flow Shop

	$\mathbf{M_1}$	$\mathbf{M_2}$
$\mathbf{J_1}$	6	3
$\mathbf{J_2}$	5	9
$\mathbf{J_3}$	4	3
$\mathbf{J_4}$	1	3

Table 3: OMSP data for machine M_2

	r_{i2}	p_{i2}	q_{i2}
O_{12}	6	3	0
O_{22}	5	4	0
O_{32}	4	9	0
O_{42}	1	3	0

Initialisation Step: $U = \phi$ and $U' = \{O_{12}, O_{22}, O_{32}, O_{42}\}$ with $t = 1$. At time $t = 1$, only operation O_{42} is available for scheduling. Hence operation i_{42} is scheduled. Set $U = \{O_{42}\}$ and $U' = \{O_{12}, O_{22}, O_{32}\}$. The release times of jobs O_{12}, O_{22} and O_{32} are updated to 7,6 and 5 respectively. Scheduling instant t is updated to 5. At time $t = 5$, only operation O_{32} is available for scheduling. Hence operation O_{32} is scheduled. Set $U = \{O_{42}, O_{32}\}$ and $U' = \{O_{12}, O_{22}\}$. The release times of operations O_{12} and O_{22} are updated to 11 and 10 respectively. Scheduling instant t is updated to 14. At time $t = 14$, both operations O_{12} and O_{22} are available for scheduling. Both these operations have $q_{ij} = 0$ but operation O_{22} has a higher processing time. Hence operation O_{22} is scheduled. Set $U = \{O_{42}, O_{32}, O_{22}\}$ and $U' = \{O_{12}\}$. The release times of operation O_{12} is updated to 16. Scheduling instant t is updated to 18. At time $t = 18$, only operation O_{12} is available for scheduling. Hence operation O_{12} is scheduled. Set $U = \{O_{42}, O_{32}, O_{22}, O_{12}\}$ and $U' = \{\phi\}$. STOP. The job sequence returned by the Dynamic Schrage heuristic is J_4-J_3-J_2-J_1. This sequence is implemented on both the machines, i.e. sequence $S_1 = O_{41}$-O_{31}-O_{21}-O_{11} on M_1 and $S_2 = O_{42}$-O_{32}-O_{22}-O_{12} on M_1.

In general the sequence returned by the Dynamic Schrage heuristic need not be the optimal schedule. We would be presenting in the next section an algorithm that will always return an optimal schedule for the $n/2/F/C_{max}$ problem. This algorithm will use the Dynamic Schrage heuristic as its core. Before we describe that algorithm we need to define certain terminology. Let the same sequence S' be implemented on both machines of an n job 2 machine Flow Shop. The resulting critical path(s) (CPs) will have one of the three possible structures shown in Fig. 2, Fig. 3 and Fig. 4.

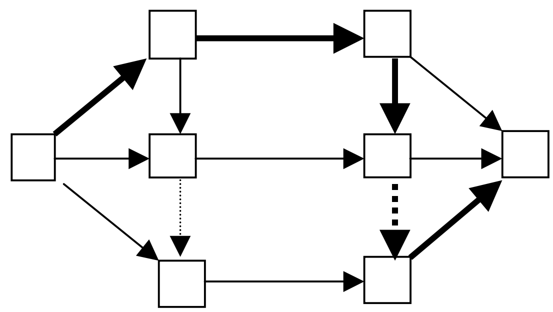

Fig. 2: Structure RD…D

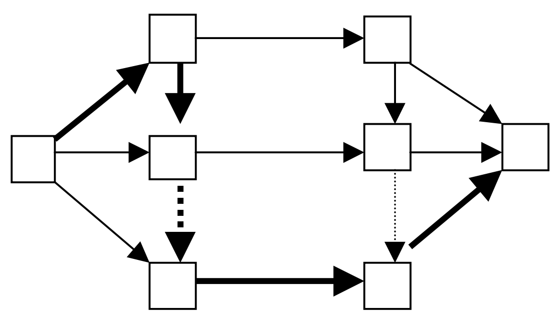

Fig. 3: Structure D…DR

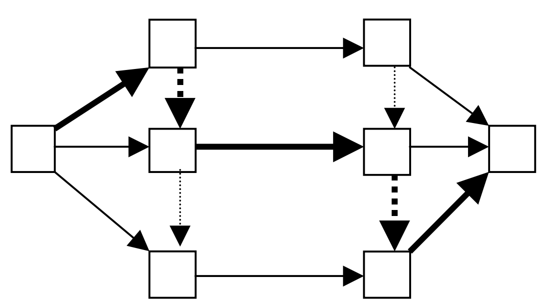

Fig. 4: Structure D…DRD…D

In each of these figures, the critical path is shown by bold lines. The names of the structures are a description of the path by which one can move from the start node to the finish node along the critical path. Here *D* stands for a downward movement and *R* stands for moving right. We next define a pivot job as follows.

Pivot Job: For the *n* job 2 machine Flow Shop with the sequence $S = (J_1, J_2, \ldots, J_n)$ implemented on both the machines, the pivot job is identified as the job through whose conjunctive arc the critical path passes from machine M_1 to M_2. In case of the existence of multiple critical paths, the pivot job is the first job in the sequence through whose conjunctive arc the critical path passes from machine M_1 to M_2.

Thus the pivot job is the job with both of its operations on the critical path. Note that at least one pivot job will exist for any feasible solution to the *n* job 2 machine flow shop. Multiple pivot jobs can exist if the schedule gives rise to multiple critical paths in the resulting digraph.

5. Proposed optimal method for solving the $n/2/F/C_{max}$ problem

We first present the optimal method and then provide two examples. The first example shows that the optimal schedule returned by this procedure can result in a sequence different from that provided by Johnson's Rule. The second example is included to illustrate the steps involved.

Dynamic Schrage with pivoting (DSP)

Step 1. Apply Dynamic Schrage on M_2.

Step 2 If *S* is the sequence obtained, apply *S* on both the machines. Identify the critical paths in the resulting digraph. If any CP structure is RD...D, the schedule is optimal. STOP. Else identify the pivot job J_k. Among the jobs that precede J_k in the CP, find job J_j such that (1) $p_{j2} < p_{j1}$ and (2) $p_{j2} < p_{k2}$. If J_j does not exist, the current sequence is optimal. STOP. Else, constrain all such jobs to be processed after J_k in all subsequent iterations. Go to Step 1.

Example 2: The processing times for 7 jobs on 2 machines is given in Table 4.

Table 4: Processing times for a 7 job 2 machine Flow Shop

	M₁	**M₂**
J_1	6	3
J_2	2	9
J_3	4	3
J_4	1	8
J_5	7	1
J_6	4	5
J_7	7	6

Johnson's Rule gives the optimum sequence as J_4-J_2-J_6-J_7-J_1-J_3-J_5 or J_4-J_2-J_6-J_7-J_3-J_1-J_5 with C_{max} = 36. Dynamic Schrage for machine M_2 returns the solution J_4-J_2-J_7-J_6-J_1-J_3-J_5. By implementing this sequence for both machines we see that the CP structure is RD…D. Hence this sequence is optimal. Indeed, this solution gives C_{max} = 36. Note that in this example an optimal solution is found out which is different from those identified by Johnson's Rule.

Example 3: The processing times for 5 jobs on 3 machines is provided in Table 5.

Table 5: Processing times for a 5 job 2 machine Flow Shop

	M_1	**M_2**
J_1	1	2
J_2	4	3
J_3	8	4
J_4	9	5
J_5	13	6

Johnson's Rule gives the optimum sequence as J_1-J_5-J_4-J_3-J_2 with C_{max} = 38. Dynamic Schrage for machine M_2 returns the solution J_1-J_2-J_3-J_4-J_5. The structure of CP is D…DR. The pivot job is J_5. Jobs J_2, J_3, J_4 are constrained to occur after J_5. Dynamic Schrage on M_2 now gives J_1-J_5-J_2-J_3-J_4. The structure of CP is D…DRD…D. The pivot job is J_3. Job J_2 is constrained to occur after J_3. Dynamic Schrage on M_2 now gives J_1-J_5-J_3-J_2-J_4. The structure of CP is D…DR. The pivot job is J_4. Job J_3 is constrained to occur after J_4. Dynamic Schrage on M_2 now gives J_1-J_5-J_4-J_3-J_2. The structure of CP is both D…DR and D…DRD…D. The pivot job is J_3. No job to select. Hence this sequence is optimal.

6. Characteristics of the schedule returned by the DSP algorithm

In this section we describe several characteristics of the schedule returned by the DSP algorithm. Let J_k be the pivot job in the schedule returned by the proposed method and let set JS_1 contain the jobs preceding the pivot job and JS_2 contain the jobs succeeding the pivot job. Hence if J denotes the set of jobs to be processed, then $J = JS_1 \cup JS_2 \cup \{J_k\}$. The set JS_2 comprises of two types of jobs – jobs that were scheduled after J_k because they were constrained to occur after J_k in an earlier iteration and jobs for which no such constraint existed. Let JS_C denote the set of jobs, which were constrained to occur after J_k in the final schedule and let JS_D denotes all other jobs. Then, $JS_2 = JS_C \cup JS_D$.

Lemma 4 In the schedule returned by the DSP algorithm, (i) $p_{i1} \le p_{k1}$, $\forall i \in JS_1$ and (ii) $p_{i1} \ge p_{k1}$, $\forall i \in JS_D$.

Proof: Let the operation O_{j2} immediately precede the operation O_{k2} in the final schedule. Let t' and t'' be the scheduling instants when the operations O_{j2} and O_{k2} were scheduled on M_2. We note that according to Step 3 of the Dynamic Schrage heuristic, the scheduling instant is updated setting $t'' = \max\,(t' + p_{j2};\ \min_{i \in U'} r_{i2},)$ and in Step 2 of the next iteration, the next operation to be scheduled is selected from the ready operations (i.e. operation O_{i2}

such that $r_{i2} \leq t''$, $i \in U'$). Since the critical path passes through conjunctive arc O_{k1}-O_{k2}, $t'' > t' + p_{j2}$ and the machine M_2 was idle at time t''-1.

Part (i): Let there exist, in the final schedule, a job J_m preceding pivot job J_k, such that $p_{m1} > p_{k1}$. Let t''' be the scheduling instant when O_{m2} was scheduled on M_2 by the Dynamic Schrage heuristic and since all processing times are strictly positive, $t''' < t''$. Since $p_{k1} < p_{m1}$, the job J_k was a ready job at scheduling instant t'''. Hence the job J_k was available for scheduling at scheduling instant t''-1, which is a contradiction.

Part (ii): Since at time t'', the set of ready jobs consisted of all jobs i for which $p_{i1} = \min_{m \in J \setminus U'} p_{m1}$; we have as a result $p_{i1} \geq p_{k1}$, $\forall i \in JS_D$. ■

Lemma 5 For the scheduled returned by the DSP algorithm, if $p_{k1} \leq p_{k2}$, then $p_{i1} \leq p_{i2}$; $\forall i \in JS_1$.

Proof: Let there exist $i \in JS_1$ for which $p_{i2} < p_{i1}$. Then, $p_{i2} < p_{i1} \leq p_{k1} \leq p_{k2}$ using Lemma 4. Then, (i) $p_{i2} < p_{i1}$ and (ii) $p_{i2} < p_{k2}$ would imply that another iteration of the DSP algorithm should have been carried out where job J_i would have been constrained to occur after J_k, which is a contradiction. Hence $p_{i1} \leq p_{i2}$; $\forall i \in JS_1$. ■

Lemma 6 For the scheduled returned by the DSP algorithm, if $p_{k1} \leq p_{k2}$, then $p_{i1} > p_{i2}$; $\forall i \in Q$, where $Q = \{i: i \in \text{JS}_2 \text{ and } p_{i*} < p_{k*}\}$.

Proof: Since $JS_2 = JS_C \cup JS_D$, and Q is a subset of JS_2, either $i \in JS_C$ or $i \in JS_D$, $\forall i \in Q$. If $i \in JS_C$, $p_{i1} > p_{i2}$; $\forall i \in Q$ since $p_{i1} > p_{i2}$; $\forall i \in JS_C$ by construction. Else if $i \in JS_D$, then $p_{i1} \geq p_{k1}$, $\forall i \in JS_D$ using Lemma 7.4. Additionally $p_{i*} < p_{k1}$ $\forall i \in Q$, since $p_{k1} \leq p_{k2}$. But since $p_{i1} \geq p_{k1}$, $\forall i \in JS_D$, $p_{i2} < p_{i1}$; $\forall i \in Q$. ■

Lemma 7 For the scheduled returned by the DSP algorithm, if $p_{k1} \geq p_{k2}$, then $p_{i1} < p_{i2}$; $\forall i \in Q$, where $Q = \{i: i \in JS_1 \text{ and } p_{i*} < p_{k*}\}$.

Proof: Let there exist $i \in Q$ for which $p_{i1} > p_{i2}$. Hence $p_{i*} = p_{i2}$ and $p_{k*} = p_{k2}$ since $p_{k1} \geq p_{k2}$. Then, (i) $p_{i2} < p_{i1}$ and (ii) $p_{i*} < p_{k*}$ would imply that another iteration of the DSP algorithm should have been carried out where job J_i would have been constrained to occur after J_k, which is a contradiction. Hence $p_{i1} \leq p_{i2}$; $\forall i \in Q$. ■

Lemma 8 For the scheduled returned by the DSP algorithm, if $p_{k1} \geq p_{k2}$, then (i) $JS_D = \phi$, (ii) $p_{i2} < p_{i1}$; $\forall i \in JS_2$ and (iii) $p_{i2} < p_{k2}$; $\forall i \in JS_2$

Proof: Let job J_m immediately succeed pivot job J_k and job J_n immediately succeed job J_m. Since the critical path passes through the conjunctive arc O_{k1}-O_{k2}, the length of the path from O_{k1} to O_{m2} via O_{m1} is smaller than the path from O_{k1} to O_{m2} via O_{k2}.

$\Rightarrow p_{k1} + p_{m1} < p_{k1} + p_{k2}$

$\Rightarrow p_{m1} < p_{k2}$

$\Rightarrow p_{m1} < p_{k1}$ since $p_{k2} \leq p_{k1}$

Hence $m \in JS_C$, as otherwise, if $m \in JS_D$ then using Lemma 4 $p_{m1} \geq p_{k1}$, which is a contradiction.

Again, since the critical path passes through the conjunctive arc O_{k1}-O_{k2}, the length of the path from O_{k1} to O_{n2} via O_{n1} is smaller than the path from O_{k1} to O_{n2} via O_{k2}.

$\Rightarrow p_{k1} + p_{m1} + p_{n1} < p_{k1} + p_{k2} + p_{m2}$

$\Rightarrow p_{n1} + (p_{m1} - p_{m2}) < p_{k2}$

$\Rightarrow p_{n1} < p_{k2}$ since $m \in JS_C$ and $p_{r2} < p_{r1}$, $\forall r \in JS_C$ by construction.

$\Rightarrow n \in JS_C$ using similar arguments as earlier.

A similar recursive analysis for all jobs succeding job J_m can be carried out to show that $i \in JS_C$; $\forall i \in JS_2$. Since $JS_2 = JS_C \cup JS_D$, $JS_D = \phi$. The results then follow noting that $p_{i2} < p_{i1}$; $\forall i \in JS_C$ by construction. ■

Lemma 9 For the scheduled returned by the DSP algorithm, there exists no job J_q such that (i) $q \in JS_2$; (ii) $p_{q1} < p_{q2}$ and (iii) $p_{q1} < p_{k1}$.

Proof: Let there exist job J_q for which conditions (i), (ii) and (iii) hold. Since $JS_2 = JS_C \cup JS_D$, and $q \in JS_2$, either $q \in JS_C$ or $q \in JS_D$. If $q \in JS_C$, $p_{q1} > p_{q2}$; since $p_{i1} > p_{i2}$; $\forall i \in JS_C$ by construction. This contradicts condition (ii). Else if $q \in JS_D$, then $p_{q1} \geq p_{k1}$, $\forall i \in JS_D$ using Lemma 4. This contradicts condition (iii). Hence the result follows. ■

7. Proof of optimality of the DSP algorithm

We first state and prove a lower bound for the $n/2/F/C_{max}$ problem. Then we show that the schedule returned by the proposed method actually equals this lower bound.

Lower Bound for $n/2/F/C_{max}$ problem

Let $J(n)$ denote the set of n jobs to be processed. Then $LB_1^{J(n)} = \sum_{i \in J(n)} p_{i1} + \min_{i \in J(n)} p_{i2}$ and $LB_2^{J(n)} = \min_{i \in J(n)} p_{i1} + \sum_{i \in J(n)} p_{i2}$ are two lower bounds for the $n/2/F/C_{max}$ problem and so is $LB^{J(n)} = \max\{LB_1^{J(n)}, LB_2^{J(n)}\}$.

We can tighten this lower bound further. Let $p_{j*} = \min\{p_{j1}, p_{j2}\}$, then $d_n = \min_{j \in J(n)} p_{j*}$, denotes the smallest processing time for $J(n)$. Let this smallest processing time belong to job J_j. We delete this job J_j to obtain the n-1 job 2 machine flow shop $J(n\text{-}1)$.

Lemma 10 $LB_{recursive} = \max\{\{d_{n+1} + LB^{J(n)}\},\{d_n + LB^{J(n-1)}\}, \{d_n + d_{n-1} + LB^{J(n-2)}\},\ldots,\{d_n + d_{n-1},..,d_1 + LB^{J(0)}\}\}$ is a lower bound for the $n/2/F/C_{max}$ problem with $d_{n+1} = 0$ and $LB^{J(0)} = 0$.

Proof: While the lower bound $LB_{recursive}$ has been defined by recursively deleting jobs, while proving the Lemma 10 we will generate the lower bound by recursively adding jobs. It is trivial to note that Lemma 10 holds for a 1 job 2 machine Flow Shop. Let $LB^{J(k)}$ denote the lower bound for a k job 2 machine Flow Shop, where k is an integer, $k \geq 1$.

Suppose we add a job J_q with processing times p_{q1} and p_{q2} on M_1 and M_2 such that $p_{q*} = \min\{p_{q1}, p_{q2}\}$ is smaller than all processing times for the k job 2 machine Flow Shop. Then for the k+1 job 2 machine problem $J(k+1)$ where the k+1th job is J_q,

$$LB^{J(k+1)} = \max\{LB_1^{J(k+1)}, LB_2^{J(k+1)}\}$$

$$\Rightarrow LB^{J(k+1)} = \max\{\min(p_{11},..., p_{k1}, p_{q1}) + \sum_{i=1,...,k+1} p_{i2} ; \sum_{i=1,...,k+1} p_{i1} + \min(p_{12},..., p_{k2}, p_{q2})\}$$

Case I: $p_{q1} = \min(p_{11},..., p_{k1}, p_{q1})$

$$\Rightarrow LB^{J(k+1)} = \max\{p_{q1} + \sum_{i=1,...,k+1} p_{i2} ; \sum_{i=1,...,k+1} p_{i1} + \min(p_{12},..., p_{k2}, p_{q2})\},$$

Case IA: $p_{q2} \neq \min(p_{12},..., p_{k2}, p_{q2})$

$$\Rightarrow LB^{J(k+1)} = \max\{p_{q1} + \sum_{i=1,...,k+1} p_{i2} ; p_{q1} + \sum_{i=1,...,k} p_{i1} + \min_{i=1,...,k}(p_{i2})\}$$

$$\Rightarrow LB^{J(k+1)} = \max\{LB_2^{J(k+1)} ; p_{q1} + LB_1^{J(k)}\}$$

Case IB: $p_{q2} = \min(p_{12},..., p_{k2}, p_{q2})$

$$\Rightarrow LB^{J(k+1)} = \max\{p_{q1} + \sum_{i=1,...,k+1} p_{i2} ; \sum_{i=1,...,k+1} p_{i1} + p_{q2})\}$$

$$\Rightarrow LB^{J(k+1)} = \max\{LB_2^{J(k+1)} ; LB_1^{J(k+1)}\}$$

Combining Case IA and Case IB,

$$LB^{J(k+1)} = \max\{LB_1^{J(k+1)} ; LB_2^{J(k+1)} ; p_{q1} + LB_1^{J(k)}\}, \quad (1)$$

Similarly for Case II, $p_{q2} = \min(p_{12},..., p_{k2}, p_{q2})$ and we can show that

$$LB^{J(k+1)} = \max \{ LB_1^{J(k+1)} ; LB_2^{J(k+1)} ; p_{q2} + LB_2^{J(k)} \}, \quad (2)$$

Combining equations (1) and (2), for any d_{k+1} defined earlier,

$$LB^{J(k+1)} = \max \{ LB_1^{J(k+1)} ; LB_2^{J(k+1)} ; \min\{ p_{q1}, p_{q2}\} + LB^{J(k)}\}$$

$$\Rightarrow LB^{J(k+1)} = \max \{ LB_1^{J(k+1)} ; LB_2^{J(k+1)} ; p_{q*} + LB^{J(k)}\}$$

$$\Rightarrow LB^{J(k+1)} = \max \{ LB_1^{J(k+1)} ; LB_2^{J(k+1)} ; d_{k+1} + LB^{J(k)}\}$$

Hence the result follows by recursion. ■

The following example shows that this new $LB_{\text{recursive}}$ is a tighter bound than $LB^{J(n)}$.

Example 4: The processing time data for a 5 job 2 machine Flow Shop is presented in Table 6.

Table 6: Processing times for a 5 job 2 machine Flow Shop

	M_1	**M_2**
J_1	3	10
J_2	7	6
J_3	8	4
J_4	1	2
J_5	9	7

For this $5/2/F/C_{\max}$ problem, we have the following lower bounds.

$$LB^{J(5)} = \max \{ LB_1^{J(5)}, LB_2^{J(5)} \}$$

$$= \max \{30, 30 \}$$

$$= 30$$

$LB_{\text{recursive}}$ = max $\{\{LB^{J(5)}\},\{d_5 + LB^{J(4)}\}, \{d_5 + d_4 + LB^{J(3)}\}, \{d_5 + d_4 + d_3 + LB^{J(2)}\}, \{d_5 + d_4 + d_3 + d_2 + LB^{J(1)}\}, \{d_5 + d_4 + d_3 + d_2 + d_1\}\}$

= max{{30},{1+31}{1+3+28},{1+3+4+22},{1+3+4+6+16},{1+3+4+6+7}}

= 32

Theorem 2: The schedule generated by the DSP algorithm equals $LB_{\text{recursive}}$

Proof: The critical path CP for the solution returned by the Dynamic Schrage heuristic can be one of 3 types – RD…D, D…DR and D…DRD…D. We take up each case separately. Let l(CP) denote the sum of processing times of the operations constituting the critical path CP.

Case 1: CP is RD…D

Here the pivot job J_k is the first job on the critical path and $l(\text{CP}) = p_{k1} + \sum_{i \in J} p_{i2}$. Since $p_{k1} = \min_{i \in J} \{p_{i1}\}$; CP = $LB_2^{J(n)}$. The claim follows since the length of the critical path equals a lower bound for the $n/2/F/C_{max}$ problem.

Case2: CP is D…DR or D…DRD…D

Here the pivot job J_k is an intermediate job on the critical path. Then $l(\text{CP}) = \sum_{i \in JS_1} p_{i1}$ + $p_{k1} + p_{k2} + \sum_{j \in JS_2} p_{j2}$. Note that if CP is D…DR, then $JS_2 = \phi$.

Case 2A: $p_{k1} < p_{k2}$

Construct set Q such that $q \in Q$, if $p_{q*} < p_{k*}$ and J_q is scheduled after J_k. Let set R contain all other jobs occurring after pivot job i.e. $R = JS_2 \setminus Q$. Then $p_{k1} = \min_{u \in R \cup \{k\}} \{p_{u1}\}$, by construction. If $JS_2 = \phi$, then $Q = \phi$ and $R = \phi$. Hence

$$l(\text{CP}) = \sum_{i \in JS_1} p_{i1} + p_{k1} + p_{k2} + \sum_{u \in R} p_{u2} + \sum_{q \in Q} p_{q2}$$

$$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1} p_{i1} + \min_{u \in R \cup \{k\}} \{p_{u1}\} + \sum_{u \in R \cup \{k\}} p_{u2} + \sum_{q \in Q} p_{q2}$$

$$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1} p_{i1} + LB_2^{R \cup \{k\}} + \sum_{q \in Q} p_{q2}$$

Using Lemma 5, $p_{i1} \leq p_{i2}$; $\forall i \in JS_1$; and $p_{q2} < p_{q1}$; $\forall q \in Q$ using Lemma 6.

$\Rightarrow p_{i1} = p_{i*}$; $\forall i \in JS_1$ and $p_{q2} = p_{q*}$; $\forall q \in Q$.

$$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1} p_{i*} + LB_2^{R \cup \{k\}} + \sum_{q \in Q} p_{q*}$$

$$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1 \cup Q} p_{i*} + LB_2^{R \cup \{k\}}$$

$$\Rightarrow l(\mathrm{CP}) = \sum_{i \in J \setminus \{R \cup \{k\}\}} p_{i*} + LB_2^{R \cup \{k\}}$$

Additionally, using Lemma 4, $p_{i1} \leq p_{k1}$; $\forall i \in JS_1$. The construction of set Q implies that p_{k1} is the smallest processing time for $R \cup \{k\}$. Hence p_{i*} is smaller than all processing times in $R \cup \{k\}$, $\forall i \in JS_1$. Similarly, it is apparent that p_{q*} is smaller than all processing times in $R \cup \{k\}$ because (i) p_{k1} is the smallest processing time for $R \cup \{k\}$ and (ii) $p_{q*} < p_{k*}$ $\forall q \in Q$ by construction. Thus p_{i*}; $\forall i \in J \setminus \{R \cup \{k\}\}$ is smaller than p_{j1}, p_{j2}; $\forall j \in R \cup \{k\}$.

$$\Rightarrow l(\mathrm{CP}) = LB_{\mathrm{recursive}}$$

Case 2B: $p_{k1} > p_{k2}$

Construct set Q such that job $q \in Q$, if $p_{q*} < p_{k*}$ and J_q comes before J_k on the critical path. Let set R contain all other jobs occurring before pivot job i.e. $R = JS_1 \setminus Q$. Then $p_{k2} = \min_{u \in R \cup \{k\}} \{p_{u2}\}$, by construction. Hence

$$l(\mathrm{CP}) = \sum_{q \in Q} p_{q1} + \sum_{u \in R} p_{u1} + p_{k1} + p_{k2} + \sum_{i \in JS_2} p_{i2}$$

$$\Rightarrow l(\mathrm{CP}) = \sum_{q \in Q} p_{q1} + \sum_{u \in R \cup \{k\}} p_{u1} + \min_{u \in R \cup \{k\}} \{p_{u2}\} + \sum_{i \in JS_2} p_{i2}$$

$\Rightarrow l(\mathrm{CP}) = \sum_{q \in Q} p_{q1} + LB_1^{R \cup \{k\}} + \sum_{i \in JS_2} p_{i2}$

Using Lemma 7, $p_{q1} \le p_{q2}$; $\forall q \in Q$; and $p_{i2} < p_{i1}$; $\forall i \in JS_2$ using Lemma 8.

$\Rightarrow p_{q1} = p_{q*}$; $\forall q \in Q$ and $p_{i2} = p_{i*}$; $\forall i \in JS_2$.

$\Rightarrow l(\mathrm{CP}) = \sum_{q \in Q} p_{q*} + LB_1^{R \cup \{k\}} + \sum_{i \in JS_2} p_{i*}$

$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_2 \cup Q} p_{i*} + LB_1^{R \cup \{k\}}$

$\Rightarrow l(\mathrm{CP}) = \sum_{i \in J \setminus \{R \cup \{k\}\}} p_{i*} + LB_1^{R \cup \{k\}}$

Additionally, using Lemma 8, $p_{i2} < p_{k2}$; $\forall i \in JS_2$. The construction of set Q implies that p_{k2} is the smallest processing time for $R \cup \{k\}$. Hence p_{i*} is smaller than all processing times in $R \cup \{k\}$, $\forall i \in JS_2$. Similarly, it is apparent that p_{q*} is smaller than all processing times in $R \cup \{k\}$ $\forall q \in Q$ because (i) p_{k2} is the smallest processing time for $R \cup \{k\}$ and (ii) $p_{q*} < p_{k*}$ $\forall q \in Q$ by construction. Thus p_{i*}; $\forall i \in J \setminus \{R \cup \{k\}\}$ is smaller than p_{j1}, p_{j2}; $\forall j \in R \cup \{k\}$.

$\Rightarrow l(\mathrm{CP}) = LB_{\mathrm{recursive}}$

Case 2C: $p_{k1} = p_{k2}$

$l(\mathrm{CP}) = \sum_{i \in JS_1} p_{i1} + p_{k1} + p_{k2} + \sum_{j \in JS_2} p_{j2}$

$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1} p_{i1} + LB^{k} + \sum_{j \in JS_2} p_{j2}$

Using Lemma 5, $p_{i1} \le p_{i2}$; $\forall i \in JS_1$; and $p_{i2} < p_{i1}$; $\forall i \in JS_2$ using Lemma 8.

$\Rightarrow p_{i1} = p_{i*}$; $\forall i \in JS_1$ and $p_{i2} = p_{i*}$; $\forall i \in JS_2$.

$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1} p_{i*} + LB^{k} + \sum_{j \in JS_2} p_{j*}$

$\Rightarrow l(\mathrm{CP}) = \sum_{i \in JS_1 \cup JS_2} p_{i*} + LB^{k}$

$\Rightarrow l(\mathrm{CP}) = \sum_{i \in J \setminus \{k\}} p_{i*} + LB^{k}$

Additionally, using Lemma 4, $p_{i1} \leq p_{k1}$; $\forall i \in JS_1$ and using Lemma 8, $p_{i2} < p_{k2}$; $\forall i \in JS_2$. Thus p_{i*}; $\forall i \in J \backslash \{k\}$ is smaller than p_{k*}.

$\Rightarrow l(\text{CP}) = LB_{\text{recursive}}$ ■

8. Convergence and complexity results

In this section we show that the DSP algorithm converges within finite number of steps. At each stage of the algorithm, a pivot job J_k is identified along with the set *CJS* such that $p_{j2} < p_{j1}$ and $p_{j2} < p_{k2}$; $\forall j \in CJS$. We call *CJS* the *candidate job set* and each member of *CJS* a *candidate job*. At each iteration *i*, let J_k^i be the pivot job and the set CJS_i the corresponding candidate job set. Similarly, JS_1^i, JS_2^i, JS_D^i are defined for each iteration paralleling the definition of JS_1, JS_2, JS_D in section 6. We now focus on the sequence of pivot jobs identified through the iteration process. Either, there is no repetition within this sequence, and hence the number of iterations is less than $n+1$. Or, let *r* be the iteration at which a pivot is repeated.

Lemma 11: $CJS_r = \phi$

Proof: If possible, let $CJS_r \neq \phi$, and job J_p be a candidate job at the *r*th iteration with job J_k as the pivot job. Let *q* be the iteration, $q < r$, when job J_k was also the pivot job. Then either of two possibilities exists.

Case 1: $p \in \mathbf{JS_1^q}$

Two possibilities exist. If $p \in CJS_q$, then the job J_p would have been constrained to occur after J_k in the *(q+1)*th iteration. Hence, J_p cannot be scheduled before J_k in the *r*th iteration, which is a contradiction. Or, if $p \notin CJS_q$, then since the pivot jobs are the same for the two iterations, $p \notin CJS_r$, which is a contradiction.

Case 2: $\boldsymbol{p \in JS_2^q}$

Since job J_p occurs after J_k in the qth iteration and before J_k in the rth iteration, two possibilities exist. Either job J_k was constrained to occur after J_p in an intermediate iteration, in which case $p_{k2} < p_{j2}$, which is a contradiction. Or job J_k was scheduled after J_p by the Dynamic Schrage heuristic in the rth iteration and not due to the enforcement of any precedence between the two jobs. But, since $p_{j2} < p_{k2}$ and $p_{j1} \geq p_{k1}$ (using Lemma 4 at the qth iteration, noting that $j \in JS_D^q$), job J_p would always be scheduled after J_k by the Dynamic Schrage heuristic, which is a contradiction. ■

Lemma 12: $(n+1)$ is an upper bound on the number of iterations.

Proof: If there is no repetition of pivot jobs, there can be n iterations in the worst case. Else if there is a repetition at the rth iteration, Lemma 11 indicates that the DSP algorithm would stop at that iteration. Since a repetition is guaranteed if the number of iterations exceeds n, the result follows. ■

We now consider the complexity status of both the procedure for calculating the lower bound and the DSP algorithm.

Lemma 13: The lower bound $LB_{\text{recursive}}$ can be computed in $\boldsymbol{O}(\text{n}^2)$ time.

Proof: The procedure for calculating $LB_{\text{recursive}}$ can be divided into 3 steps. In Step 1, a sorting is done on the jobs according to their p_{i*} value in $\boldsymbol{O}(n\log n)$ time. In Step 2, $\{d_{n+1} + LB^{J(n)}\}, \{d_n + LB^{J(n-1)}\}, \{d_n + d_{n-1} + LB^{J(n-2)}\}, \ldots, \{d_n + d_{n-1} +,.., + d_1 + LB^{J(0)}\}$ values are calculated, each taking $\boldsymbol{O}(n)$ time. Hence Step 2 can be completed in $\boldsymbol{O}(n^2)$ time. Calculation of the maximum of these values can be done in $\boldsymbol{O}(n)$ time in Step 3. Hence, the lower bound can be calculated in $\boldsymbol{O}(n^2)$ time. ■

Lemma 14: The worst case complexity of the DSP algorithm is $\boldsymbol{O}(n^3)$.

Proof: The Schrage heuristic is essentially a sort and hence can be done in $\boldsymbol{O}(n\log n)$ time as shown in Carlier (1982). The Dynamic Schrage heuristic differs from the Schrage heuristic in the additional task of updation of release times for all unscheduled jobs, which needs $\boldsymbol{O}(n^2)$ time on the whole. Hence the complexity of the Dynamic Schrage heuristic is $\boldsymbol{O}(n^2)$. Lemma 12 suggests that in the worst case (n+1) iterations of the Dynamic Schrage heuristic would be needed. Hence the result follows. ■

9. Concluding Remarks

We have examined the apparent failure of the SB heuristic in providing optimal solutions to Flow Shop problems. An alternative optimal machine based decomposition procedure has been provided for the $n/2/F/C_{\max}$ problem along with complexity results. The contribution of the present study lies in showing that the same machine based decomposition procedures which are so successful in solving complex Job Shops can also be suitably modified to optimally solve the simpler Flow Shops. It is hoped that this paper will stimulate research in the application of machine based decomposition procedures for the general $n/m/F/C_{\max}$ problem.

References

Adams, J., Balas, E., and Zawack, D. "The Shifting Bottleneck Procedure for Job Shop Scheduling," *Management Science*, 34,3(1988), 391-401.

Balas, E., Lenstra, J.K., and Vazacopoulos, A. "The One Machine Sequencing Problem with Delayed Precedence Constraints and its Use in Job Shop Scheduling," *Management Science*, 41 (1995), 94-109.

Brucker, P., Jurisch, B., and Sievers, B. "A branch and bound algorithm for the job-shop scheduling problem", *Discrete Applied Mathematics*, 49 (1994), 107-127.

Campbell, H.G., Dudek, R.A., and Smith, M.L. “A heuristic algorithm for the *n* job *m* machine sequencing problem,” *Management Science*, 16, 630-637.

Carlier, J. “The One Machine Sequencing Problem,” *European Journal of Operational Research*, 11 (1982) 42-47.

Carlier, J., and Pinson, E. “An Algorithm for Solving the Job Shop Problem”, *Management Science*, 35 (1989), 164-176.

Conway, R.N., Maxwell, W. L., and Miller, L. W. *Theory of Scheduling*, Addison-Wesley, Reading, MA, 1967.

Dauzere-Peres, S., and Lasserre, J.B. “A modified shifting bottleneck procedure for job-shop scheduling”, *International Journal of Production Research*, 31 (1993), 923-932.

Demirkol, E., and Uzsoy, R. “Decomposition methods for scheduling re-entrant flow shops with sequence dependent set up times”, Research Report, School of Industrial Engineering, Purdue University, 1998.

Demirkol, E., Mehta, S.V., and Uzsoy, R. “A computational study of shifting bottleneck procedures for shop scheduling problems”, *Journal of Heuristics*, 3 (1997), 111-137.

Fisher, M.L., Lageweg, B.J., Lenstra, J.K., and Rinnooy Kan, A.H.G. “Surrogate duality relaxation for job shop scheduling”, *Discrete Applied Mathematics*, 5 (1983), 65-75.

Holtsclaw, H.H., and Uzsoy, R. “Machine criticality measures and subproblem solution procedures in shifting bottleneck methods: a computational study”, *Journal of the Operational Research Society*, 47 (1996), 666-677.

Hundal, T.S., and Rajgopal, J. “An extension of Palmer heuristic for the flow-shop scheduling problem”, *International Journal of Production Research*, 26 (1988), 1119-1124.

Jain, A.S., and Meeran, S. “A multi-level hybrid framework applied to the general flow-shop scheduling problem”, *Computers and Operations Research*, 29 (2002), 1873-1901.

Johnson, S.M. “Optimal two- and three-stage production schedules with set up times included”, *Naval Research Logistics Quarterly*, 1 (1954), 61-68.

Nawaz, M., Enscore, E.E., and Ham, I. “A heuristic algorithm for the *m*-machine *n*-job flow shop sequencing problem”, *Omega*, 11 (1983), 91-95.

Osman, I.H., and Potts, C.N. “Simulated annealing for permutation flow shop scheduling”, *Omega*, 17 (1989), 551-557.

Panwalker, S.S. and Iskander, W. “A survey of scheduling rules”, *Operations Research*, 25 (1977), 45-61.

Pinedo, M. *Scheduling : Theory, Algorithms and Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

Pinedo, M., and Singer, M. “A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop”, Research Report, Department of Industrial Engineering and Operations Research, Columbia University, 1996.

Ramudhin, A., and Marier, P. “The generalized shifting bottleneck procedure”, *European Journal of Operational Research*, 93 (1996), 34-48.

Sun, X., and Noble, J.S. “An approach to job shop scheduling with sequence-dependent setups”, *Journal of Manufacturing Systems*, 18, 6 (1999), 416-430.

Szwarc, W. “Dominance conditions for the three-machine flow shop problem”, *Operations Research*, 26 (1978), 203-206