Sādhanā (2020)45:228
https://doi.org/10.1007/s12046-020-01444-9

© Indian Academy of Sciences

Permutation flowshop scheduling to obtain the optimal solution/a lower bound with the makespan objective

THAMARASSERY ABDULJALEEL JESSIN[1,2], SAKTHIVEL MADANKUMAR[3] and CHANDRASEKHARAN RAJENDRAN[1,*]

[1]Department of Management Studies, Indian Institute of Technology Madras, Chennai 600036, India
[2]TKM College of Engineering, Kollam 691005, India
[3]Trimble Information Technologies India Private Limited, Chennai 600113, India
e-mail: tajessin2008@yahoo.co.in; madankumar.sakthivel@gmail.com; craj@iitm.ac.in

MS received 24 January 2020; revised 25 April 2020; accepted 3 June 2020

Abstract. This paper focuses on developing the optimal solution or a lower bound for *N*-job, *M*-machine Permutation Flowshop Scheduling (PFS) problem in a manufacturing system with the objective of minimizing the makespan using Lagrangian Relaxation (LR) technique. Even though LR technique is considered, in general, as a good method to obtain a lower bound, research in this direction with respect to our problem under study appears scarce. We address this gap by developing two MILP based Lagrangian Relaxation models, namely, Lagrangian Relaxation Method 1 (called Proposed Lagrangian Lower Bound Program (PLLBP)) and Alternate Lagrangian Relaxation Method 1 (called ALR) to find the optimal solution or a lower bound on the makespan. Basically, we develop these LR methods to overcome the possible limitation of the general LR procedure involving the sub-gradient approach. Benchmark PFS problem instances are used to evaluate the performance of these methods. It is observed that the PLLBP outperforms the ALR, and it provides better lower bounds than the lower bounds (in most instances) reported in the literature. Even though the PLLBP is superior in terms of solution quality, it has a limitation in that it cannot execute problem instances beyond 500 jobs due to the associated computational effort.

Keywords. Manufacturing; permutation flowshop scheduling; makespan; mixed integer linear programming model; lower bound; Lagrangian relaxation.

1. Introduction

A flowshop is a conventional manufacturing system where all jobs are processed on all machines with the same sequence of operations on machines. The Permutation Flowshop Problem (PFS) is a special case of the flowshop problem where we schedule a set of *N* jobs non-preemptively on *M* machines in the same order on every machine. Since the publication of Johnson's seminal paper [1], the PFS problem has become a topic of interest for researchers. Among the performance measures, the minimization of makespan is the widely studied objective due to the implication that the minimization of makespan leads to the reduction in idle times of machines, especially the minimum idle time of the last machine and the production run length. The problem is denoted by $F|prmu|C_{max}$ (see Pinedo [2]). It is well known for the two-machine flowshop problem with the consideration of the makespan objective, Johnson's rule [1] can be used to generate an optimal schedule in $O(nlogn)$ time. For more than two machines, even though the problem is identified as NP-complete (see Garey *et al* [3]), many exact methods have been suggested in the literature to solve the problem: e.g. branch-and-bound technique and mathematical programming models.

In the literature, for optimally solving flowshop scheduling problems by using Mixed Integer Linear Programming (MILP) approach, the models developed by Wagner [4] and Manne [5] serve as the base models for research. Stafford *et al* [6] categorized the MILP models under Wagner family and Manne family. Wagner family consists of models developed by Wilson [7], Tseng and Stafford [8], and Stafford and Tseng [9]. Manne family includes models developed by Stafford and Tseng [10], and Liao and You [11]. Later, researchers such as Pan [12] and Tseng *et al* [13] made comparative evaluation of MILP models to find the best performing model. Tseng *et al* [13] observed that Wagner family is better than Manne family for finding optimal solutions to PFS problems with the objective of minimizing the makespan. MILP models can also be seen in literature for variants of the scheduling problems such as

*For correspondence
Published online: 07 September 2020

those by Bautista-Valhondo and Alfaro-Pozo [14], and Xu *et al* [15]. The branch-and-bound technique is another exact method for solving flowshop scheduling problems. Ignall and Schrage [16] and Lomnicki [17] independently developed branch and bound algorithms (also see Land and Doig [18], and Little *et al* [19] for solving the travelling salesman problem) to the PFS problem with the makespan objective. The computation of lower bounds is important in a branch-and-bound technique as it influences the computational quality of the technique. Thus, developing tight lower bounds has become an area of interest for researchers. Many researchers such as Brown and Lomnicki [20], McMahon and Burton [21], Baker [22], Bestwick and Hastings [23], Lageweg *et al* [24], Potts [25], Carlier and Rebai [26], Chung *et al* [27], Madhushini *et al* [28] and Kumar *et al* [29] developed bounding strategies for the PFS problem.

Due to the computational complexity of the exact methods for solving PFS problems, researches were carried out on developing heuristics to obtain optimal or near-optimal solutions. Some efficient heuristics for solving PFS problems were developed by Campbell *et al* [30], Nawaz *et al* [31], Rajendran [32], Liu and Reeves [33], Rossi *et al* [34], Fernandez-Viagas *et al* [35] and Pan *et al* [36]. Later, metaheuristics such as simulated annealing (SA), genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO) gained wide attention among researchers to solve PFS problems.

For the PFS problem with more than two machines, being NP-complete, it is not always possible to find the optimal solution as the problem size increases. In order to evaluate the quality of a heuristic solution, it is necessary to develop (tight) lower bounds on the objective function. While carrying out the literature review, it is observed that there exists scope to develop efficient lower bound algorithms to get tight lower bounds. Lagrangian Relaxation (LR) is a well-known technique to find lower bounds. Liu *et al* [37] adopted the LR technique to minimize the additive penalties on product tardiness and on releasing raw materials too early in PFS problems. Akkan and Karabati [38] adopted linear programming relaxation technique to obtain a lower bound for a two-machine flowshop total completion time problem (also see Hoogeveen *et al* [39]). Hamdi and Loukil [40] adopted the LR technique to obtain lower bounds for PFS problem to minimize the total tardiness with minimal and maximal time lags.

Even though it is evident from the literature review that the LR technique is an efficient approach to find optimal solutions or lower bounds, research in this domain appears scarce for obtaining optimal solutions or lower bounds for *N*-job, *M*-machine flowshop scheduling problems with the makespan objective. To address this, we focus on developing LR-based procedures. In the literature, an LR technique is the Lagrangian Decomposition Duality technique (see Fisher [41]; Guignard and Kim [42]; Liu *et al* [37]; and Tang *et al* [43]) where the problem is decomposed into individual subproblems (Lagrangian Decomposition) which are comparitively easier to solve, after identifying the parts of the problem to be split. Thereafter, the variables in each part are replaced by copies or substituted with new expressions. Finally the copy or substituted expression is dualized attaching the Lagrange multiplier (LM). The sub-gradient method is commonly used to solve Lagrangian dual problems, while updating the LM values (see Hamdi and Loukil [40]; Bazaraa and Goode [44] and Fisher [45]). We observe that such LR approaches, especially decomposing the problem into subproblems, are difficult to be carried out for Permutation Flowshop Scheduling (PFS) problems because identifying the parts of the problem to split is somewhat subjective and challenging, while the LR procedure similar to that adopted by Beasley [46] is simple and easy to implement. Such an LR procedure does not have the complicated decomposition procedure; instead it has a simple structured way of implementing the procedure. First, Lagrangian Lower Bound Program (LLBP) is solved and then we improve the lower bound obtained by adopting the sub-gradient approach, which involves simple steps like calculating the subgradient, finding step size, updating Lagrange multipliers and finally improving the lower bound (see section 3 for details). In comparison with the decomposition method, the procedure similar to Beasley's LR procedure is a simple method which is easy to implement (for example, see Kumar and Narendran [47] and Beasley and Christofides [48]), and hence we experiment this method (based on the sub-gradient approach) as our first attempt for PFS problems (called LLBP). Since it is not found to perform well, we subsequently develop new LR methods which are computationally easy and effective, in comparison to the LR procedure based on the sub-gradient method approach.

The rest of this paper is structured as follows. Section 2 presents the base MILP model for the minimization of the makespan of jobs in PFS problems. Section 3 presents the generic Lagrangian Relaxation method (called LLBP) based on the sub-gradient approach, followed by our Lagrangian Relaxation method 1 (called Proposed Lagrangian Lower Bound Program (PLLBP)) for obtaining a lower bound on the makespan. The computational experience is reported in section 4. Section 5 presents the discussion on the PLLBP and the generic LR technique. Section 6 gives additional computational details, including the Alternate Lagrangian Relaxation method 1, called ALR. Limitations of our proposed model are reported in section 7, and finally section 8 presents the conclusions and future scope of the work.

2. Base MILP Model for Minimization of the Makespan of Jobs in Flowshop Scheduling

In this section, we present the MILP model that forms the base model for our LR-based procedures. The base or the foundation for the MILP model presented in this paper is

attributed to Wilson [7]. Kumar *et al* [29] modified the model by introducing a bounding constraint for tightening the model (note that Kumar *et al* [29] considered the total flowtime objective). We now present indices, parameters and decision variables used in this paper.

2.1 *Indices and parameters*

N Number of jobs.
M Number of machines.
n Index for a job; $n = 1, \ldots, N$.
m Index for a machine; $m = 1, \ldots, M$.
$p_{n,m}$ Processing time of job n on machine m
l Index for the position; $l = 1, \ldots, N$.
$\gamma_{l,m}$ Lower bound on the completion time of job placed in postion l on machine m.
r Index for the problem instance; $r \in$ {set of problem instances by Taillard [49]; Vallada *et al* [50] (called VRF in our paper)}.
i Index for the LM (heuristically derived) setting; $i \in \{1, 10, 20, 30, 40, 50, 60, 80, 100, 150, 200, 500\}$.

2.2 *Decision variables*

$$y_{n,l} = \begin{cases} 1, & \textit{if job n is placed in position l in the permutation sequence;} \\ 0, & \textit{otherwise.} \end{cases}$$

$\propto_{l,m}$
Start time of the job processed in position l on machine m.

2.3 *MILP model*

Minimize

$$Z_1 = \alpha_{N,M} + \sum_{n=1}^{N} (p_{n,M} \times y_{n,N}) \tag{1}$$

subject to the following:

$$\sum_{l=1}^{N} y_{n,l} = 1, \quad \text{for} \quad 1 \le n \le N. \tag{2}$$

$$\sum_{n=1}^{N} y_{n,l} = 1, \quad \text{for} \quad 1 \le l \le N. \tag{3}$$

$$\propto_{1,1} = 0. \tag{4}$$

$$\propto_{1,m} = \propto_{1,m-1} + \sum_{n=1}^{N} (p_{n,m-1} \times y_{n,1}), \quad \text{for} \quad 2 \le m \le M. \tag{5}$$

$$\propto_{l,1} = \propto_{l-1,1} + \sum_{n=1}^{N} (p_{n,1} \times y_{n,l-1}), \quad \text{for} \quad 2 \le l \le N. \tag{6}$$

$$\propto_{l,m} \ge \propto_{l-1,m} + \sum_{n=1}^{N} (p_{n,m} \times y_{n,l-1}), \quad \text{for} \quad 2 \le m \le M, 2 \le l \le N. \tag{7}$$

$$\propto_{l,m} \ge \propto_{l,m-1} + \sum_{n=1}^{N} (p_{n,m-1} \times y_{n,l}), \quad \text{for} \quad 2 \le m \le M, 2 \le l \le N. \tag{8}$$

$$\propto_{l,m} + \sum_{n=1}^{N} (p_{n,m} \times y_{n,l}) \ge \gamma_{l,m}, \quad \text{for} \quad 2 \le m \le M, 2 \le l \le N. \tag{9}$$

$$y_{n,l} \in \{0, 1\}, \quad \text{for} \quad 1 \le n \le N, 1 \le l \le N. \tag{10}$$

$$\propto_{l,m} \ge 0, \quad \text{for} \quad 1 \le m \le M, 1 \le l \le N. \tag{11}$$

Equation (1) indicates the objective function of the MILP model which represents the sum of the start time of last job placed on the last machine and the process time of job placed in position N on machine M. Constraint (2) represents that each job can be placed only in one position. Constraint (3) indicates that each position in the permutation sequence can be occupied by only one job. Constraint (4) ensures that the start time of job, being processed on first machine placed in the first position of the job sequence, takes place at time zero. Constraint (5) represents the start time of job being processed on all machines except the first machine, placed in first position of the job sequence, is equal to the start time of job being processed on the previous machine placed in the first position of the job sequence plus the process time of job, on the previous machine, placed in the first position of the permutation sequence. Constraint (6) indicates that start time of job, being processed on the first machine placed in any position l excluding the first position, is equal to the start time of job being processed on the first machine placed in the previous position of the permutation sequence plus the process time of job on first machine placed in the previous position of the job sequence. Constraints (7) and (8) ensure that the start time of the job placed in any position$l$, excluding the first position being processed on machine m other than the first machine, is equal to larger of the start time of processing of the job placed in previous position $(l-1)$ on machine m plus the process time of the job placed in the previous position on machine m, and the start time of processing of the job placed in position l on previous machine $(m-1)$ plus the process time of the job on

previous machine $(m-1)$ in position l. Constraint (9) ensures that the sum of start time and process time of the job placed in position l on machine m is greater than or equal to the lower bound on completion time of job processed on machine m placed in position l (see Appendix A for details of this expression). Integrality and non-negativity constraints are represented by (10) and (11).

3. Discussion on the Generic Lagrangian Relaxation (LR) Technique

In this section, we first discuss the application of the Lagrangian Relaxation (LR) technique widely followed in the literature for optimization problems (e.g., see Beasley [46], Kumar and Narendran [47], and Beasley and Christofides [48]) and then we present our newly proposed LR-based procedures to obtain the optimal solution or a lower bound on the objective function.

Lagrangian Relaxation (LR) technique was developed by Held and Karp [51, 52] in the early 1970s for solving the traveling salesman problem. LR is an approach in which optimization problems with hard constraints are solved by relaxing these constraints and by introducing them into the objective function by attaching Lagrange multipliers. Such a relaxed problem provides a lower bound on the optimal solution to the original problem. We attempt to obtain a lower bound on the makespan by adopting Lagrangian Relaxation technique. In Lagrangian Relaxation (LR) technique, from the set of constraints, the binding constraint is chosen and is relaxed by introducing it to the objective function by attaching Lagrange multipliers. Since our objective is to minimize the makespan, from the set of constraints considered, the binding constraints to the objective functions are constraints (7) and (8), as they represent the completion time of a job placed in previous position $(l-1)$ on machine m excluding the first position and the first machine; and the completion time of the job placed in position l on previous machine $(m-1)$ excluding the first position and the first machine respectively. To choose the binding constraint among (7) and (8), we carry out trial runs by relaxing constraint (7) (the corresponding proposed LR method is called PLLBP; see Section 3.2) and constraint 8 (the corresponding proposed LR method is called ALR; see Section 6.1).

3.1 *The Lagrangian Lower Bound Program (LLBP)*

The Lagrangian lower bound formulation, based on the sub-gradient approach, is as follows.

Minimize

$$Z_{LB} = \alpha_{N,M} + \sum_{n=1}^{N}\left(p_{n,M} \times y_{n,N}\right) + \sum_{l=2}^{N}\sum_{m=2}^{M} LM_{l,m} \times \left(\left(\propto_{l-1,m} + \sum_{n=1}^{N}\left(p_{n,m} \times y_{n,l-1}\right)\right) - \propto_{l,m}\right) \quad (12)$$

subject to the following:

(2), (3), (4), (5), (6), (8), (9), (10) and (11).

Note that

$$LM_{l,m} > 0, \quad \text{for} \quad 2 \le m \le M, 2 \le l \le N. \quad (13)$$

The procedure adopted in our study, following Beasley [46], is described below.

Step 1: Initially set the user-defined parameter $\mathcal{E}$, satisfying $0 \le \mathcal{E} \le 2$ (in this study we set $\mathcal{E} = 1$).
Step 2: Initialise Z_{UB} to an upper bound.
Step 3: Set Z_{max} equal to $(-\infty)$.
Step 4: Set the initial set of Lagrange multipliers, $LM_{l,m}$, for $1 \le m \le M, 1 \le l \le N$ (in this study we experiment with $LM_{l,m} \in \{0.0001, 0.001, 0.01, 0.1, 1\}$).
Step 5: With the set of Lagrange multipliers, the LR problem is solved to get a solution Z_{LB}.
Step 6: Calculate $S_{l,m}$ as follows:

$$S_{l,m} = \left(\propto_{l-1,m} + \sum_{n=1}^{N}\left(p_{n,m} \times y_{n,l-1}\right)\right) - \propto_{l,m}, \quad \text{for } 2 \le m \le M, 2 \le l \le N. \quad (14)$$

Step 7: Step size, ST, is calculated as follows:

$$ST = \frac{\mathcal{E}(Z_{UB} - Z_{LB})}{\sum_{l=2}^{N}\sum_{m=2}^{M}(S_{l,m})^2} \quad (15)$$

Step 8: Lagrange multipliers are updated for the next iteration using the following equation:

$$LM_{l,m} = \max\left(0, LM_{l,m} + ST \times S_{l,m}\right), \text{for } 2 \le m \le M, 2 \le l \le N. \quad (16)$$

Step 9: Z_{max} is updated as follows:

$$Z_{max} = \max(Z_{max}, Z_{LB}). \quad (17)$$

Step 10: Terminate the algorithm if it reaches the pre-set time limit. Otherwise, go to step 5 and continue the iteration process (for this study we set the time limit as 600 seconds). First, we set $LM_{l,m} = 0.0001$, for $1 \le m \le M, 1 \le l \le N$ and execute the LR procedure. The best solution is returned. We repeat the execution for $LM_{l,m} \in \{0.001, 0.01, 0.1, 1\}$.

To evaluate the computational efficiency of the LR technique (LLBP), the method is coded in C ++ and executed on a computing facility with Intel Xeon 2.4 GHz (2 processors) and 64 GB RAM using the ILOG CPLEX Solver (Version 12.8). For illustration, we consider a small-sized sample problem with 4 jobs and 4 machines (processing time matrix is given in table 11), and experiment with different Lagrange multiplier settings, $LM_{l,m}$ = {0.0001, 0.001, 0.01, 0.1, 1}. For every LM setting, the solver gives unbounded solution status. For further investigation, we constrain $\propto_{l,m}$ to be less than or equal to 100, and execute the MILP model. It is observed that the lower bound (Z_{LB}) value given by the LLBP is fluctuating (i.e., oscillating) and the current implementation terminates at the end of 5562 iterations, and we could have restarted the same procedure once again with the Lagrangian multiplier values obtained from iteration 5562; but from the observation of the first set of iterations (5562), we observe that the lower bound values would be fluctuating and it may not have converged to optimal/improved lower bound (the detailed discussion is given in Appendix B with respect to the setting of $\propto_{l,m} \leq 100$).

The implementation of the Lagrangian Relaxation method yields unbounded solution, and adding an additional constraint to the mathematical model by constraining $\propto_{l,m}$ to be less than or equal to 100 is not found to be attractive. Therefore, we present the Lagrangian Relaxation technique by adding an additional constraint (and the corresponding LR term accounts for its non-negativity), called Lagrangian Relaxation Method 1 (called Proposed Lagrangian Lower Bound Program (PLLBP)). The PLLBP is now presented.

3.2 *Lagrangian Relaxation Method 1 (called Proposed Lagrangian Lower Bound Program (PLLBP))*

In this method, we introduce a variable $z_{l,m}$ that takes the value corresponding to $\left(\alpha_{l-1,m} + \sum_{n=1}^{N}\left(p_{n,m} \times y_{n,l-1}\right)\right) - \alpha_{l,m}$, when the respective expression results in a positive value, otherwise it takes the value zero. Note that when $z_{l,m}$ is greater than zero, it indicates infeasibility. Hence, when the variable $z_{l,m}$ is introduced into the objective function and multiplied by Lagrange multiplier $LM_{l,m}$, the resultant objective function corresponds to Lagrangian Relaxation objective function. The Proposed Lagrangian Lower Bound Program (PLLBP) is discussed below.

Minimize

$$Z_{PLLBP} = \alpha_{N,M} + \sum_{n=1}^{N}\left(p_{n,M} \times y_{n,N}\right) + \sum_{l=2}^{N}\sum_{m=2}^{M}\left(LM_{l,m} \times z_{l,m}\right) \tag{18}$$

subject to the following:

(2), (3), (4), (5), (6), (8), (9), (10), (11) and

$$z_{l,m} \geq \left(\alpha_{l-1,m} + \sum_{n=1}^{N}\left(p_{n,m} \times y_{n,l-1}\right)\right) - \alpha_{l,m}, \quad \text{for} \quad 2 \leq m \leq M, 2 \leq l \leq N, \tag{19}$$

with the non-negativity constraint

$$z_{l,m} \geq 0, \quad \text{for} \quad 2 \leq m \leq M, 2 \leq l \leq N. \tag{20}$$

Note that

$$LM_{l,m} > 0, \quad \text{for} \quad 2 \leq m \leq M, 2 \leq l \leq N. \tag{21}$$

Table 1. Pilot run: The lower bound on the makespan reported in VRF [50] (called GLB_{VRF}); the best lower bound on the makespan yielded by the PLLBP (called LB_{PLLBP}), the upper bound on the makespan yielded by the PLLBP (called UB_{PLLBP}), the corresponding LM (called LM_{best}) and the optimality gap (called OG-LB_{PLLBP}); the lower bound on the makespan yielded by the extended execution of PLLBP for one hour (called $LB_{Ext\text{-}PLLBP}$), the upper bound on the makespan yielded by the extended execution of PLLBP for one hour (called $UB_{Ext\text{-}PLLBP}$), and the optimality gap (called OG-$LB_{Ext\text{-}PLLBP}$).

N-M-Instance	GLB_{VRF}	LB_{PLLBP}	UB_{PLLBP}	LM_{best}	OG-LB_{PLLBP} (in %)	$LB_{Ext\text{-}PLLBP}$	$UB_{Ext\text{-}PLLBP}$	OG-$LB_{Ext\text{-}PLLBP}$ (in %)
60-20-1	3689	3876	4942	60	21.6	3878	4593	15.6
60-20-2	3804	3895	5121	100	23.9	3895	4712	17.3
60-20-3	3870	3994	5487	1	27.2	3994	4689	14.8
60-20-4	3731	3930	5265	20	25.4	3965	4478	11.5
60-20-5	3747	3839	5174	10	25.8	3839	4531	15.3
60-20-6	3764	3845	5327	60	27.8	3845	4542	15.3
60-20-7	3818	3956	4592	20	13.9	3959	4592	13.8
60-20-8	3758	3865	4937	10	21.7	3865	4586	15.7
60-20-9	3764	3885	5174	50	24.9	3903	4595	15.1
60-20-10	3779	3837	5535	500	30.7	3878	4589	15.5

Table 2. The lower bound (LB) on the makespan reported in Taillard [49] (called GLB_T), the best lower bound on the makespan obtained from the PLLBP over 12 LM settings (called LB_{PLLBP}), lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour (called $LB_{Ext\text{-}PLLBP}$), the upper bound (UB) reported in Chakravorty and Laha [53] (called UB_{CL}), for $N = \{20, 50\}$ and $M = \{5, 10, 20\}$; $N = 100$ and $M = \{5, 10, 20\}$; $N = 200$ and $M = \{10, 20\}$; $N = 500$ and $M = 20$, for Taillard benchmark problem instances.

Instance	GLB_T	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{CL}	Instance	GLB_T	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{CL}
20-5-1	1232	1278*	-	1278	50-5-1	2712	2724*	-	2724
20-5-2	1290	1359*	-	1359	50-5-2	2808	2834*	-	2836
20-5-3	1073	1081*	-	1081	50-5-3	2596	2621*	-	2621
20-5-4	1268	1293*	-	1293	50-5-4	2740	2751*	-	2751
20-5-5	1198	1235*	-	1235	50-5-5	2837	2863*	-	2863
20-5-6	1180	1195*	-	1195	50-5-6	2793	2829*	-	2829
20-5-7	1226	1234*	-	1239	50-5-7	2689	2725*	-	2725
20-5-8	1170	1206*	-	1206	50-5-8	2667	2683*	-	2683
20-5-9	1206	1230*	-	1230	50-5-9	2527	2552*	-	2554
20-5-10	1082	1108*	-	1108	50-5-10	2776	2782*	-	2782
20-10-1	1448	1562#	—	1582	50-10-1	2907	2970#	—	3037
20-10-2	1479	1636#	—	1659	50-10-2	2821	2829#	—	2911
20-10-3	1407	1465#	—	1496	50-10-3	2801	2832#	—	2873
20-10-4	1308	1376#	—	1378	50-10-4	2968	3063#	—	3067
20-10-5	1325	1404#	—	1419	50-10-5	2908	2933#	—	3021
20-10-6	1290	1397*	—	1397	50-10-6	2941	2997#	—	3021
20-10-7	1388	1484*	—	1484	50-10-7	3062	3088#	—	3124
20-10-8	1363	1527#	—	1538	50-10-8	2959	3034#	—	3048
20-10-9	1472	1593*	—	1593	50-10-9	2795	2885#	—	2913
20-10-10	1356	1561#	—	1591	50-10-10	3046#	3046#	—	3114
20-20-1	1911	2130#	—	2297	50-20-1	3480	3585#	3585#	3886
20-20-2	1711	1925#	—	2100	50-20-2	3424	3553#	3553#	3733
20-20-3	1844	2193#	—	2326	50-20-3	3351	3402	3404#	3689
20-20-4	1810	2035#	—	2223	50-20-4	3336	3448	3454#	3755
20-20-5	1899	2138#	—	2291	50-20-5	3313	3389#	3389#	3655
20-20-6	1875	2064#	—	2226	50-20-6	3460	3538	3540#	3719
20-20-7	1875	2106#	—	2273	50-20-7	3427	3506#	3506#	3730
20-20-8	1880	2051#	—	2200	50-20-8	3383	3454#	3454#	3744
20-20-9	1840	2065#	—	2237	50-20-9	3457	3481#	3481#	3790
20-20-10	1900	2047#	—	2178	50-20-10	3438	3540#	3540#	3791
100-5-1	5437	5493*	-	5493	200-10-1	10816	10840	10846#	10885
100-5-2	5208	5268*	-	5274	200-10-2	10422	10439#	10439#	10570
100-5-3	5130	5175*	-	5175	200-10-3	10886	10895#	10895#	10948
100-5-4	4963	5014*	-	5018	200-10-4	10794	10824#	10824#	10911
100-5-5	5195	5250*	-	5250	200-10-5	10437	10472	10482#	10537
100-5-6	5063	5135*	-	5135	200-10-6	10255	10286	10287#	10378
100-5-7	5198	5246*	-	5246	200-10-7	10761	10804#	10804#	10882
100-5-8	5038	5094*	-	5095	200-10-8	10663	10692	10697#	10778
100-5-9	5385	5448*	-	5448	200-10-9	10348	10381	10385#	10438
100-5-10	5272	5322*	-	5328	200-10-10	10616	10631	10643#	10727
100-10-1	5759	5759	5761#	5776	200-20-1	10979	11023#	11023#	11421
100-10-2	5345#	5345#	5345#	5362	200-20-2	10947	10971#	10971#	11547
100-10-3	5623	5654	5659#	5679	200-20-3	11150	11204	11207#	11537
100-10-4	5732	5759	5781#	5820	200-20-4	11127	11160#	11160#	11566
100-10-5	5431	5443	5444#	5491	200-20-5	11132	11154#	11154#	11463
100-10-6	5246	5286	5299#	5308	200-20-6	11085	11122	11142#	11416
100-10-7	5523	5556#	5556#	5602	200-20-7	11194	11220#	11220#	11594
100-10-8	5556	5600#	5600#	5640	200-20-8	11126	11187#	11187#	11582
100-10-9	5779	5836	5848#	5891	200-20-9	10965	10997	11004#	11482
100-10-10	5830	5835	5837#	5860	200-20-10	11122	11209#	11209#	11547
100-20-1	5851	5921#	5921#	6345	500-20-1	25922	25069	25934#	26509
100-20-2	6099	6118#	6118#	6323	500-20-2	26353	25311	26386#	26941
100-20-3	6099	6147	6159#	6385	500-20-3	26320	24951	26330#	26714

Table 2 continued

Instance	GLB_T	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{CL}	Instance	GLB_T	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{CL}
100-20-4	6072	6138	6156#	6331	500-20-4	26424	25779	26438#	26815
100-20-5	6009	6127#	6127#	6405	500-20-5	26181	26149	26234#	26635
100-20-6	6144	6188#	6188#	6487	500-20-6	26401#	26039	26039	26872
100-20-7	5991	6040	6044#	6393	500-20-7	26300	25632	26324#	26573
100-20-8	6084	6153#	6153#	6514	500-20-8	26429	26058	26475#	26893
100-20-9	5979	6048#	6048#	6386	500-20-9	25891	25138	25904#	26300
100-20-10	6298	6356	6361#	6544	500-20-10	26315	25995	26366#	26748

Legends: * : optimal solution is given by PLLBP.

: best lower bound among the proposed method and the corresponding value reported in the literature.

- : $LB_{Ext\text{-}PLLBP}$ is not executed as LB_{PLLBP} yields the optimal solution.

— : even though LB_{PLLBP} has not yielded optimal solution for the problem instances, we have not executed $LB_{Ext\text{-}PLLBP}$ because optimality gap is not large ($\leq$ about 12%).

The non-negative variable $z_{l,m}$ (LR term) is introduced in the binding constraint (7), and this is taken into the objective function with the associated Lagrange multiplier.

4. Computational experiments

We now present the details of computational experiments pertaining to the LR methods, LM settings and the computational experience with the benchmark problem instances.

4.1 *Settings of Lagrange Multipliers in the PLLBP*

In most of the LR procedures reported in the literature, the value of the Lagrange multiplier (LM) is set initially to a small value and gradually increased. For every setting of LM, we solve the problem to obtain the corresponding objective function. However we observe that in the PFS problem under study it becomes harder to solve a problem instance using an LR procedure as the problem size increases; typically, in large-sized problems, we observe that the PLLBP could not optimally be solved (for a given LM setting) even after execution for one hour. In such cases, we either get a lower bound and an upper bound on the objective function (associated with an optimality gap) or we do not get a lower bound/an upper bound. Depending upon the problem instance, the performance of the PLLBP with a given LM setting may be different with respect to another LM setting (with a given CPU time restriction for the execution of the PLLBP).

In our study, we set a constant value for LM (i.e., $LM_{l,m} = LM$, for all l and m). An aspect of the Lagrangian Relaxation method is the selection of LM values. In this paper, we investigate different LM values using the heuristic runs conducted for the test set of VRF [50] benchmark problem instances with 60 jobs and 20 machines in the pilot study. From the VRF benchmark problem instances, those with 60 jobs and 20 machines are selected for conducting heuristic pilot runs because they are the largest among the medium-sized problem sets and close to the larger-sized problems. It is observed from table 1 that PLLBP gives the best solution for $LM \in \{1, 10, 20, 50, 60, 100, 500\}$. Hence we heuristically choose LM values as 1, 10, 20, 30, 40, 50, 60, 80, 100, 150, 200 and 500, considering the range [1, 500]. We do not claim that the LM settings selected by us are indeed the best; the primary purpose of this study is to propose an LR based Lower Bound Program to obtain an optimal solution (or a lower bound on the optimal solution) to the PFS problem (with the assumed CPU time restriction), and not to propose conclusively the best LR procedure. We believe that our PLLBP serves the purpose of obtaining optimal solutions or good lower bounds on optimal solutions (with the stated restricted execution of the PLLBP), and that seems to be evident from our computational experience.

For every such LM setting, we execute the PLLBP for 10 minutes and truncate the execution, and thereafter return the lower bound on the objective function. The best of the 12 settings (in respect of the best lower bound on the objective function called LB_{PLLBP} with the corresponding best Lagrange multiplier, called LM_{best}) is chosen, and with this LM setting, we execute the PLLBP for one hour and report the lower bound on the objective function (called $LB_{Ext\text{-}PLLBP}$). It is to be noted that when we execute PLLBP with a given LM setting (up to 10 minutes) and if we obtain the lower bound and the upper bound with an optimality gap of zero and all $z_{l,m}$'s are equal to zero, then the corresponding lower bound and upper bound will be the same, and the solution is indeed optimal. This observation is true even when we execute the PLLBP with the chosen LM for up to one hour. Therefore it is evident that when one of the executions of the PLLBP with the corresponding LM is an optimal solution, there is no need to go for the extended run of PLLBP for one hour. For this reason, in some of the tables, we do not report $LB_{Ext\text{-}PLLBP}$ when we have optimality yielded by LB_{PLLBP} (for example, see table 2, where

Table 3. LB and UB on the makespan reported in Vallada *et al* [50] (called GLB_{VRF} and UB_{best} respectively) for $N = \{10, 20, 30, 40, 50, 60\}$, $M = \{5, 10, 15, 20\}; N = \{100\}, M = \{20, 40, 60\}; N = \{200, 300\}, M = \{20, 40\}; N = \{400, 500\}, M = \{20\}$, for VRF instances.

Instance	GLB_{VRF}	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{best}	Instance	GLB_{VRF}	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{best}
10-5-1	523	695*	-	695	20-5-1	1095	1192*	-	1192
10-5-2	556	698*	-	698	20-5-2	1173	1275*	-	1275
10-5-3	588	728*	-	728	20-5-3	1224	1323*	-	1323
10-5-4	565	697*	-	697	20-5-4	1047	1127*	-	1127
10-5-5	578	713*	-	713	20-5-5	1244	1339*	-	1339
10-5-6	617	748*	-	748	20-5-6	994	1066*	-	1066
10-5-7	602	728*	-	728	20-5-7	1078	1154*	-	1154
10-5-8	568	683*	-	683	20-5-8	1030	1102*	-	1102
10-5-9	633	761*	-	761	20-5-9	1231	1317*	-	1317
10-5-10	554	664*	-	664	20-5-10	1164	1243*	-	1243
10-10-1	797	1097*	-	1097	20-10-1	1290	1501#	—	1532
10-10-2	845	1146*	-	1146	20-10-2	1292	1499#	—	1525
10-10-3	831	1124*	-	1124	20-10-3	1352	1529#	—	1592
10-10-4	769	1038*	-	1038	20-10-4	1226	1387#	—	1442
10-10-5	817	1093*	-	1093	20-10-5	1371	1548#	—	1604
10-10-6	812	1085*	-	1085	20-10-6	1348	1495#	—	1576
10-10-7	839	1115*	-	1115	20-10-7	1363	1561#	—	1591
10-10-8	840	1113*	-	1113	20-10-8	1353	1543#	—	1574
10-10-9	789	1045*	-	1045	20-10-9	1312	1487#	—	1530
10-10-10	832	1099*	-	1099	20-10-10	1279	1484#	—	1489
10-15-1	921	1307*	-	1307	20-15-1	1525	1807#	—	1936
10-15-2	988	1399*	-	1399	20-15-2	1526	1845#	—	1905
10-15-3	996	1398*	-	1398	20-15-3	1453	1673#	—	1798
10-15-4	1041	1452*	-	1452	20-15-4	1466	1659#	—	1813
10-15-5	992	1373*	-	1373	20-15-5	1517	1805#	—	1875
10-15-6	964	1329*	-	1329	20-15-6	1587	1866#	—	1960
10-15-7	1049	1445*	-	1445	20-15-7	1566	1933#	—	1933
10-15-8	1048	1443*	-	1443	20-15-8	1477	1668#	—	1822
10-15-9	1058	1428*	-	1428	20-15-9	1575	1788#	—	1940
10-15-10	1085	1461*	-	1461	20-15-10	1511	1722#	—	1861
10-20-1	1191	1652*	-	1652	20-20-1	1741	2209#	—	2270
10-20-2	1273	1759*	-	1759	20-20-2	1682	2008#	—	2170
10-20-3	1254	1726*	-	1726	20-20-3	1772	2096#	—	2277
10-20-4	1236	1678*	-	1678	20-20-4	1685	2025#	—	2165
10-20-5	1259	1700*	-	1700	20-20-5	1736	2107#	—	2225
10-20-6	1400	1889*	-	1889	20-20-6	1790	2082#	—	2291
10-20-7	1251	1678*	-	1678	20-20-7	1785	2253#	—	2282
10-20-8	1235	1655*	-	1655	20-20-8	1707	1993#	—	2178
10-20-9	1280	1706*	-	1706	20-20-9	1851	2229#	—	2354
10-20-10	1248	1663*	-	1663	20-20-10	1732	2140#	—	2199
30-5-1	1727	1805*	-	1805	40-5-1	2292	2396*	-	2396
30-5-2	1510	1575*	-	1575	40-5-2	2351	2442*	-	2442
30-5-3	1608	1673*	-	1673	40-5-3	2106	2174*	-	2174
30-5-4	1716	1781*	-	1781	40-5-4	2082	2149*	-	2149
30-5-5	1645	1707*	-	1707	40-5-5	2179	2247*	-	2247
30-5-6	1807	1875*	-	1875	40-5-6	2091	2154*	-	2154
30-5-7	1686	1749*	-	1749	40-5-7	2143	2207*	-	2207
30-5-8	1646	1706*	-	1706	40-5-8	2350	2414*	-	2414
30-5-9	1674	1735*	-	1735	40-5-9	2247	2305*	-	2305
30-5-10	1582	1637*	-	1637	40-5-10	2289	2348*	-	2348
30-10-1	1721	1877#	—	1944	40-10-1	2258	2395#	—	2480
30-10-2	1860	1958#	—	2098	40-10-2	2237	2327#	—	2444
30-10-3	1857	1983#	—	2077	40-10-3	2244	2320#	—	2412
30-10-4	1747	1808#	—	1945	40-10-4	2313	2403#	—	2472
30-10-5	1818	1954#	—	2023	40-10-5	2276	2384#	—	2425
30-10-6	1830	1974#	—	2043	40-10-6	2387	2460#	—	2547
30-10-7	1767	1866#	—	1967	40-10-7	2344	2419#	—	2501
30-10-8	1701	1807#	—	1896	40-10-8	2338	2413#	—	2491
30-10-9	1712	1838#	—	1908	40-10-9	2275	2342#	—	2411
30-10-10	1722	1847#	—	1915	40-10-10	2337	2465#	—	2478
30-15-1	1999	2158	2165#	2381	40-15-1	2624	2763#	2763#	3011
30-15-2	1952	2129#	2129#	2318	40-15-2	2491	2596	2630#	2821
30-15-3	1950	2089	2131#	2304	40-15-3	2572	2719#	2719#	2906
30-15-4	2079	2265#	2265#	2444	40-15-4	2576	2684	2710#	2919
30-15-5	2062	2230	2237#	2423	40-15-5	2615	2722	2762#	2945
30-15-6	1968	2058	2088#	2306	40-15-6	2494	2599	2604#	2805
30-15-7	1978	2179#	2179#	2316	40-15-7	2555	2684#	2684#	2868

Table 3 continued

Instance	GLB_{VRF}	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{best}	Instance	GLB_{VRF}	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{best}
30-15-8	2019	2173	2183#	2366	40-15-8	2578	2698	2714#	2900
30-15-9	1929	2020	2024#	2259	40-15-9	2407	2522	2523#	2708
30-15-10	2047	2180#	2180#	2385	40-15-10	2625	2744	2745#	2945
30-20-1	2119	2277	2280#	2643	40-20-1	2770	3018#	3018#	3326
30-20-2	2284	2490	2516#	2835	40-20-2	2697	2819	2830#	3226
30-20-3	2265	2462	2472#	2783	40-20-3	2713	2848	2870#	3233
30-20-4	2213	2385	2399#	2680	40-20-4	2724	2888	2902#	3233
30-20-5	2205	2331	2370#	2672	40-20-5	2571	2717#	2717#	3055
30-20-6	2245	2387	2396#	2715	40-20-6	2699	2821#	2821#	3192
30-20-7	2244	2437	2453#	2712	40-20-7	2749	2874#	2874#	3244
30-20-8	2328	2511	2538#	2812	40-20-8	2764	2908	2924#	3266
30-20-9	2318	2477	2488#	2795	40-20-9	2834	2983	2992#	3335
30-20-10	2329	2456	2462#	2805	40-20-10	2653	2796	2815#	3122
50-5-1	2970	3055*	-	3055	60-5-1	3276	3350*	-	3350
50-5-2	2784	2853*	-	2853	60-5-2	2990	3054*	-	3054
50-5-3	2682	2746*	-	2746	60-5-3	3147	3214*	-	3214
50-5-4	2773	2836*	-	2836	60-5-4	3202	3266*	-	3266
50-5-5	2806	2866*	-	2866	60-5-5	3139	3197*	-	3197
50-5-6	2782	2841*	-	2841	60-5-6	3058	3107*	-	3107
50-5-7	2548	2600*	-	2600	60-5-7	3263	3315*	-	3315
50-5-8	2631	2684*	-	2684	60-5-8	3386	3438*	-	3438
50-5-9	2570	2621*	-	2621	60-5-9	3074	3121*	-	3121
50-5-10	2781	2834*	-	2834	60-5-10	3608	3663*	-	3663
50-10-1	2746	2846#	—	2926	60-10-1	3256	3335#	—	3435
50-10-2	2841	2949#	—	3035	60-10-2	3489	3601#	—	3655
50-10-3	2836	2924#	—	3019	60-10-3	3261	3394#	—	3423
50-10-4	2838	2930#	—	3003	60-10-4	3305	3425#	—	3455
50-10-5	3070	3206#	—	3252	60-10-5	3359	3449#	—	3505
50-10-6	2973	3115#	—	3149	60-10-6	3448	3532#	—	3594
50-10-7	2722	2809#	—	2842	60-10-7	3501	3522#	—	3654
50-10-8	2932	2978#	—	3072	60-10-8	3402	3506#	—	3552
50-10-9	2858	2968#	—	3022	60-10-9	3529	3628#	—	3685
50-10-10	2906	2964#	—	3056	60-10-10	3346	3459#	—	3492
50-15-1	2988	3082	3102#	3316	60-15-1	3623	3764	3779#	3940
50-15-2	3037	3129	3133#	3347	60-15-2	3582	3666	3667#	3888
50-15-3	2998	3100	3111#	3301	60-15-3	3590	3644	3676#	3880
50-15-4	3192	3325	3327#	3521	60-15-4	3413	3530	3532#	3716
50-15-5	3039	3127	3156#	3334	60-15-5	3611	3663	3679#	3881
50-15-6	3042	3193#	3193#	3346	60-15-6	3598	3705#	3705#	3893
50-15-7	3181	3275#	3275#	3490	60-15-7	3529	3602#	3602#	3809
50-15-8	3135	3240	3243#	3430	60-15-8	3456	3550#	3550#	3749
50-15-9	2928	3022#	3022#	3205	60-15-9	3512	3583	3613#	3800
50-15-10	3104	3200	3212#	3399	60-15-10	3610	3752#	3752#	3902
50-20-1	3164	3291	3295#	3693	60-20-1	3689	3876	3878#	4163
50-20-2	3224	3354	3361#	3719	60-20-2	3804	3895#	3895#	4290
50-20-3	3284	3429	3440#	3784	60-20-3	3870	3994#	3994#	4365
50-20-4	3231	3367	3369#	3709	60-20-4	3731	3930	3965#	4193
50-20-5	3157	3308	3314#	3632	60-20-5	3747	3839#	3839#	4196
50-20-6	3295	3405	3422#	3795	60-20-6	3764	3845#	3845#	4202
50-20-7	3219	3358	3365#	3696	60-20-7	3818	3956	3959#	4263
50-20-8	3295	3405	3417#	3783	60-20-8	3758	3865#	3865#	4180
50-20-9	3337	3437#	3437#	3816	60-20-9	3764	3885	3903#	4221
50-20-10	3301	3408	3416#	3769	60-20-10	3779	3837	3878#	4202
100-20-1	5705	5785	5786#	6198	200-40-1	11812	11895#	11895#	13132
100-20-2	5836	5941#	5941#	6306	200-40-2	11735	11443	11832#	13102
100-20-3	5771	5861#	5861#	6238	200-40-3	11879	11389	12003#	13264
100-20-4	5783	5888#	5888#	6245	200-40-4	11862	12077#	12077#	13232
100-20-5	5876	5946	5952#	6296	200-40-5	11662	11822#	11822#	13043
100-20-6	5913	5997	6016#	6321	200-40-6	11788	11827#	11827#	13124
100-20-7	6004	6042	6064#	6434	200-40-7	11926	12005#	12005#	13299
100-20-8	5694	5784#	5784#	6104	200-40-8	11864	11951#	11951#	13238
100-20-9	5928	5970	5979#	6354	200-40-9	11817	11439	11889#	13166
100-20-10	5766	5874#	5874#	6145	200-40-10	11921	12020#	12020#	13228
100-40-1	6611	6769	6772#	7881	300-20-1	15773	15828#	15828#	16149
100-40-2	6738	6878	6905#	8007	300-20-2	16123	16202#	16202#	16512
100-40-3	6698	6818#	6818#	7935	300-20-3	15835	15883#	15883#	16173
100-40-4	6715	6838#	6838#	7932	300-20-4	15860	15956#	15956#	16181
100-40-5	6778	6876	6884#	8011	300-20-5	15987	16056#	16056#	16342
100-40-6	6816	6956	6959#	8023	300-20-6	15804	15889#	15889#	16137
100-40-7	6793	7004#	7004#	8006	300-20-7	15932	15999#	15999#	16266

Table 3 continued

Instance	GLB_{VRF}	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{best}	Instance	GLB_{VRF}	LB_{PLLBP}	$LB_{Ext\text{-}PLLBP}$	UB_{best}
100-40-8	6807	6917	6921#	7979	300-20-8	16093	16146#	16146#	16416
100-40-9	6758	6889	6906#	7931	300-20-9	16059	16077#	16077#	16376
100-40-10	6774	6842	6853#	7952	300-20-10	16517	16626#	16626#	16899
100-60-1	7502	7630	7660#	9395	300-40-1	16764	16350	16946#	18298
100-60-2	7720	7974#	7974#	9596	300-40-2	17015#	16626	16626	18454
100-60-3	7523	7728	7732#	9349	300-40-3	16983#	16499	16499	18457
100-60-4	7596	7805#	7805#	9426	300-40-4	16914#	16598	16598	18351
100-60-5	7638	7843	7871#	9465	300-40-5	17070#	16578	16578	18484
100-60-6	7809	7979	7989#	9667	300-40-6	16978	16348	17057#	18449
100-60-7	7576	7807	7835#	9391	300-40-7	17049	16368	17132#	18419
100-60-8	7697	7961#	7961#	9534	300-40-8	16927#	16667	16667	18392
100-60-9	7706	7841#	7841#	9527	300-40-9	16987	16640	17080#	18394
100-60-10	7774	7887	7902#	9598	300-40-10	17007	16695	17074#	18401
200-20-1	10928	10968#	10968#	11305	400-20-1	20727	20007	20768#	21120
200-20-2	10898	11002#	11002#	11265	400-20-2	21092	20456	21176#	21457
200-20-3	10958	11065#	11065#	11327	400-20-3	21133	20200	21215#	21441
200-20-4	10857	10912#	10912#	11208	400-20-4	20942	20295	20963#	21247
200-20-5	10861	10889#	10889#	11208	400-20-5	21203	21266#	21266#	21553
200-20-6	10981	11038#	11038#	11367	400-20-6	20944	20142	20948#	21214
200-20-7	11034	11073	11077#	11380	400-20-7	21331	21365#	21365#	21625
200-20-8	10783	10881	10886#	11141	400-20-8	21029	20943	21043#	21277
200-20-9	10773	10829#	10829#	11123	400-20-9	21059	20308	21135#	21346
200-20-10	10965	11005#	11005#	11310	400-20-10	21235	20879	21347#	21538
500-20-1	26071	25732	26133#	26411	500-20-6	26152#	25766	25766	26443
500-20-2	26358#	25999	25999	26681	500-20-7	26163#	25765	25765	26433
500-20-3	26116	25700	26147#	26409	500-20-8	26062#	25204	25204	26318
500-20-4	25844	24880	25894#	26124	500-20-9	26199#	26147	26147	26442
500-20-5	26505#	26004	26004	26781	500-20-10	25838#	25343	25343	26072

Legends: * : optimal solution is given by PLLBP.

: best lower bound among the proposed method and the corresponding lower bound value reported in the literature.

- : $LB_{Ext\text{-}PLLBP}$ is not executed as LB_{PLLBP} yields the optimal solution.

— : even though LB_{PLLBP} has not yielded optimal solution for the problem instances, we have not executed $LB_{Ext\text{-}PLLBP}$ because optimality gap is not large ($\leq$ about 12%).

we have not reported $LB_{Ext\text{-}PLLBP}$ for some problem instances indicating the optimality of LB_{PLLBP}). In the same table we also report $LB_{Ext\text{-}PLLBP}$, where both $LB_{Ext\text{-}PLLBP}$ and LB_{PLLBP} are the same in some cases. In such cases, note that the execution of PLLBP with a given LM setting yields LB_{PLLBP} with the optimality gap greater than zero, and hence we execute PLLBP with the chosen LM setting executed for one hour, the LB thus obtained is called $LB_{Ext\text{-}PLLBP}$. It is observed that the upper bound decreases and eventually the solver returns the same value with a less optimality gap (see table 1). This is the reason why in the same table we have LB_{PLLBP} and $LB_{Ext\text{-}PLLBP}$ having the same values for some problem instances. It is also evident from tables 2 and 3 that the PLLBP with the run of one hour with the chosen LM setting yields $LB_{Ext\text{-}PLLBP}$ that is better than LB_{PLLBP}. Furthermore, it is to be noted that we do not opt for the one-hour execution of PLLBP when we obtain a lower bound and an upper bound with an optimality gap of less than 12% (see table 2). This is done so to save the computational effort in our exercise. It is to be noted that when the problem is not solved to optimality (note that when $z_{l,m} = 0$ and the optimality gap is zero, it indicates that the solution is feasible and optimal), the lower bound values returned by the solver, after truncation, are reported in the tables.

In order to find out the performance of LM settings, Average Relative Error (ARE_i) for each LM setting i is calculated. The equation for ARE_i is given below:

$$ARE_i = \frac{\sum_{r=1}^{I} \frac{\left((LB_{PLLBP})_r - (PLLBP_i)_r\right)}{(LB_{PLLBP})_r}}{I}, \quad \text{for} \quad 1 \leq i \leq 12. \tag{22}$$

In the above equation, $(LB_{PLLBP})_r$ represents the best lower bound on the makespan yielded by the proposed LB_{PLLBP} for the problem instance r across all LM settings, $(PLLBP_i)_r$ denotes the lower bound on the makespan yielded by the proposed LB_{PLLBP} for the problem instance r corresponding to LM setting i and I is the total number of problem instances.

Table 4. Average relative error (ARE_i) for Taillard [49] problem instances for$i$: 1 to 12.

LM: i	1	10	20	30	40	50	60	80	100	150	200	500	Overall ARE
ARE_i	5.731	0.189	0.211	0.182	0.177	0.199	0.197	0.200	0.204	0.175	0.167	0.209	0.167

Table 5. Average relative error (ARE_i) for VRF [50] problem instances for i : 1 to 12.

LM: i	1	10	20	30	40	50	60	80	100	150	200	500	Overall ARE
ARE_i	7.864	0.463	0.429	0.450	0.490	0.457	0.382	0.436	0.416	0.456	0.418	0.405	0.382

Table 6. Number of times the optimal/the best upper bound is obtained for a particular LM for each $N \times M$, for Taillard [49] problem instances.

	LM											
$N \times M$	1	10	20	30	40	50	60	80	100	150	200	500
20×5	10	10	10	10	10	10	10	10	10	10	10	10
20×10	3	2	3	3	4	2	2	4	1	2	3	3
20×20	3	1	0	0	2	2	0	1	0	0	1	0
50×5	10	10	10	10	10	10	10	10	10	10	10	10
50×10	3	4	2	3	3	1	1	1	1	1	2	1
50×20	2	1	2	2	3	1	0	3	2	1	1	2
100×5	9	9	9	8	8	9	8	8	8	9	7	9
100×10	4	5	5	3	5	5	6	6	6	5	4	3
100×20	1	1	3	2	0	1	1	1	0	1	3	1
200×10	5	3	4	5	3	5	5	5	3	3	4	3
200×20	9	9	9	6	6	7	10	6	8	8	6	7
500×20	1	10	10	10	10	10	10	10	10	10	10	10
Total	60	65	67	62	64	63	63	65	59	60	61	59

4.2 *Computational experience*

For numerical testing of the PLLBP model, the proposed method has been coded in C ++ and executed on a computing facility with Intel Xeon 2.4 GHz (2 processors) and 64 GB RAM using ILOG CPLEX Solver (Version 12.8), considering Taillard benchmark problem instances with $N \in \{20, 50, 100, 200, 500\}$ and $M \in \{5, 10, 20\}$; VRF benchmark small-sized and medium-sized problem instances with $N \in \{10, 20, 30, 40, 50, 60\}$ and $M \in \{5, 10, 15, 20\}$ and large-sized problem with $N \in \{100, 200, 300, 400, 500\}$ and $M \in \{20, 40, 60\}$. In order to evaluate the efficiency of our algorithm, we are reporting the upper bound obtained from efficient heuristic algorithms found in the literature.

Table 2 presents the lower bound on makespan reported in Taillard [49] (called GLB_T), the best lower bound on the makespan obtained from the PLLBP over 12 LM settings (called LB_{PLLBP}), the lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour (called $LB_{Ext\text{-}PLLBP}$) and the upper bound (UB) reported in Chakravorty and Laha [53] (called UB_{CL}) for Taillard benchmark problem instances with $N \in \{20, 50, 100\}$ and $M \in \{5, 10, 20\}$; $N = 200$ and $M \in \{10, 20\}$; $N = 500$ and $M = 20$. It is seen that for the Taillard series, all the problem instances except the instances with 500 jobs and 20 machines, lower bound on makespan values obtained by the PLLBP, run for 10 minutes, are better than the values reported in the literature. For 500 jobs and 20 machines, the $LB_{Ext\text{-}PLLBP}$ lower bounds are better except for one instance. For evaluating the performance of our PLLBP, we have considered the upper bound on makespan reported in Chakravorty and Laha [53]. The authors developed a hybrid algorithm by combining Immune Algorithm with the Simulated Annealing (HIA) for PFS problems with makespan objective. To evaluate the performance of the algorithm, HIA has been compared with 13 metaheuristic algorithms with respect to computation time and solution quality and concluded that HIA is very competitive with the state-of-the-art procedures. Hence we reproduce their results in our paper. Thus from table 2, it is obvious that our PLLBP yields a lower bound that is indeed quite close to the upper bound. Table 3 presents the best global lower bound found in literature (see Vallada *et al* [50] for GLB_{VRF}), the lower bound on the makespan

Table 7. Number of times the optimal/the best upper bound is obtained for a particular LM for each $N \times M$, for VRF [50] problem instances.

	LM											
$N \times M$	1	10	20	30	40	50	60	80	100	150	200	500
10×5	10	10	10	10	10	10	10	10	10	10	10	10
10×10	10	10	10	10	10	10	10	10	10	10	10	10
10×15	10	10	10	10	10	10	10	10	10	10	10	10
10×20	10	10	10	10	10	10	10	10	10	10	10	10
20×5	10	10	10	10	10	10	10	10	10	10	10	10
20×10	2	1	2	1	2	2	1	1	1	1	1	3
20×15	3	0	0	0	1	1	1	1	1	0	1	1
20×20	0	0	0	1	0	3	1	0	1	1	2	1
30×5	10	10	10	10	10	10	10	10	10	10	10	10
30×10	3	2	2	1	1	1	1	2	1	1	2	2
30×15	1	2	1	0	2	1	0	2	1	1	1	0
30×20	2	2	0	1	2	2	2	1	2	2	1	3
40×5	10	10	10	10	10	10	10	10	10	10	10	10
40×10	4	2	0	1	0	0	0	0	1	0	1	1
40×15	1	0	1	0	1	0	0	4	0	2	2	0
40×20	3	2	1	3	1	1	1	1	2	1	3	3
50×5	10	10	10	10	10	10	10	10	10	10	10	10
50×10	0	1	1	0	1	2	1	2	4	1	1	3
50×15	1	0	2	1	0	2	1	2	0	0	1	1
50×20	1	2	0	3	1	1	0	0	0	2	2	3
60×5	10	10	10	10	10	10	10	10	10	10	10	10
60×10	0	0	3	3	2	1	2	0	1	1	1	1
60×15	2	2	0	1	1	0	2	0	2	2	3	0
$60 \times 20^{\$}$	1	3	4	2	2	3	4	2	2	2	2	3
100×20	4	3	2	5	4	1	3	4	3	3	3	2
100×40	8	9	8	8	7	7	9	6	9	8	8	8
100×60	10	10	10	10	10	10	10	10	10	10	10	10
200×20	7	8	7	8	9	8	9	9	7	8	8	6
200×40	2	6	5	7	4	5	4	4	6	4	7	7
300×20	7	6	6	3	5	7	7	9	6	6	6	6
300×40	0	10	10	10	10	10	10	10	10	10	10	10
400×20	1	8	8	8	8	9	9	9	9	9	10	9
500×20	1	10	10	10	10	10	10	10	10	10	10	10
Total	154	179	173	177	174	177	178	179	179	175	186	183

Legend: $: Problem set selected for pilot run.

obtained by LB_{PLLBP} and $LB_{Ext\text{-}PLLBP,}$ and the best upper bound on makespan reported in Vallada *et al* [50] for VRF series (called UB_{best}). It is seen that for all the 240 small-sized and medium-sized problem instances, the lower bound on makespan values obtained by the LB_{PLLBP} are better than the values reported in the literature. For the 240 large-sized problem instances, all instances of $N = 100$ and $M \in \{20, 40, 60\}$; $N \in \{200, 300\}$ and $M = \{20\}$; and most problem instances in table 3, LB_{PLLBP} lower bounds are better than the values reported in the literature. For all instances of $N = 200$, $M = 40$ and $N = 400$, $M = 20$, and for many instances of $N = 300$, $M = 40$ and $N = 500$, $M = 20$, $LB_{Ext\text{-}PLLBP}$ lower bounds are better than the values reported in the literature; and also from table 3 we observe that our PLLBP yields a lower bound that is indeed close to the upper bound. We therefore conclude that our proposed PLLBP is a very competitive method.

Tables 4 and 5 present the Average Relative Error (ARE_i) for each LM setting $i\,(\text{for}\, i : 1 \,\text{to}\, 12)$ for Taillard problem instances [49] and VRF problem instances [50]. It is seen that for Taillard problem instances, 200 seems to be the best performing LM setting, and for VRF problem instances, 60 appears to be the best performing LM setting. We cannot conclude which LM setting has the best performance from the identified 12 LM settings. Tables 6 and 7 present the number of times an optimal/the best upper

Table 8. Lower Bound on the makespan reported in Taillard [49] (called GLB_T), lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour (called $LB_{Ext\text{-}PLLBP}$) and lower bound on the makespan obtained from ALR (called $LB_{Ext\text{-}ALR}$) run for one hour, for a set of Taillard benchmark problem instances across different problem sizes.

Instance	GLB_T	$LB_{Ext\text{-}PLLBP}$	$LB_{Ext\text{-}ALR}$
50-20-8	3383	3454#	3450
100-10-4	5732	5781#	5775
100-20-4	6072	6156#	6126
200-10-1	10816	10846	10848#
200-10-10	10616	10643#	10635
500-20-1	25922	25934#	25446
500-20-5	26181	26234#	26233

Legend: #: The best lower bound on the makespan among the lower bound on the makespan reported in Taillard [49], the lower bound on the makespan obtained from PLLBP that is executed for one hour, and the lower bound on the makespan obtained from ALR that is run for one hour.

Table 9. Lower Bound on the makespan reported in Taillard [49] (called GLB_T), lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour (called $LB_{Ext\text{-}PLLBP}$), lower bound on the makespan obtained from the MILP model (called LB_{MILP}) and LP model (called LB_{LP}) that are run for one hour each (at the maximum), for a set of Taillard benchmark problem instances across different problem sizes.

Instance	GLB_T	$LB_{Ext\text{-}PLLBP}$	LB_{MILP}	LB_{LP}
50-20-8	3383	3454#	3440	3417
100-10-4	5732	5781#	5781#	5745
100-20-4	6072	6156#	6147	6108
200-10-1	10816	10846#	10840	10828
200-10-10	10616	10643	10645#	10628
500-20-1	25922	25934#	25446	25933
500-20-2	26353	26386#	445	26276
500-20-5	26181	26234#	26233	26233

Legend: #: The best lower bound on the makespan among lower bound on the makespan reported in Taillard [49], the lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour, lower bound on the makespan obtained from the MILP model run for one hour and lower bound on the makespan obtained from the LP model run for one hour. The reported lower bound values correspond to the best values returned by the solver for the respective problem instances.

Table 10. Lower Bound on the makespan reported in VRF [50], lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour (called $LB_{Ext\text{-}PLLBP}$) and lower bound on the makespan obtained from the MILP model (called LB_{MILP}) and LP model (called LB_{LP}) run for one hour each (at the maximum), for a set of VRF benchmark problem instances across different problem sizes.

Instance	GLB_{VRF}	$LB_{Ext\text{-}PLLBP}$	LB_{MILP}	LB_{LP}
200-40-2	11735	11832#	11443	11832#
300-20-6	15804	15889#	15889#	15889#
300-40-1	16764	16946#	16528	16457
300-40-9	16987	17080#	16640	16817
400-20-1	20727	20768#	20768#	20768#
400-20-4	20942	20963#	20963#	20962
500-20-1	26071	26133#	25751	26071

Legend: #: The best lower bound on the makespan among lower bound on the makespan reported in VRF [50], the lower bound on the makespan obtained from PLLBP with the best LM setting run for one hour, lower bound on the makespan obtained from the MILP model run for one hour and lower bound on the makespan obtained from the LP model run for one hour. The reported lower bound values correspond to the best values returned by the solver for the respective problem instances.

bound is obtained for a particular LM for each $N \times M$ for VRF [50] and Taillard [49] problem instances, respectively.

It is observed from tables 2 and 3 that tighter lower bounds that are superior to the lower bounds seen in the literature are obtained with our Proposed Lagrangian Lower Bound Program (PLLBP); though we cannot always claim this because the lower bounds from our proposed Lagrangian Relaxation based procedure are not necessarily better for all problem instances, especially for large-sized jobs more than 500. But looking at the solution quality of the lower bounds obtained from PLLBP and also due to the fact that we are able to obtain optimal solutions or good lower bounds by adopting PLLBP method, our PLLBP is a superior method to find a lower bound on the makespan. Interested readers may contact the authors for the CPLEX code and output files. Further discussion on the PLLBP and the generic LR technique is given in section 5.

5. Discussion on the PLLBP and the Generic LR technique

In the PLLBP, we introduce an additional set of variables $(z_{l,m})$ to keep track of the violation quantities when the relaxed constraints are infeasible $(\propto_{l,m} \leq (\propto_{l-1,m} + \sum_{n=1}^{N} (p_{n,m} \times y_{n,l-1}))$ to the original problem; and the violation quantities $(z_{l,m})$ are 0 when the respective relaxed constraints are feasible to the original problem.

Proposition 1 *Lower bound obtained from the current iteration* (Z_{PLLBP}^{iter}) $\geq$ *Lower bound obtained from the previous iteration* (Z_{PLLBP}^{iter-1}).

Proof In the objective function of the PLLBP, non-negative variables $z_{l,m}$ are multiplied with the corresponding positive Lagrange multipliers $LM_{l,m}$; and since this is a minimization problem, the optimizer/solver attempts to make the violation quantities $(z_{l,m})$to take minimum values (possibly 0) depending upon the values of the Lagrange multipliers $(LM_{l,m})$. So in the proposed method, if we increase the values of the Lagrange multipliers $(LM_{l,m})$ in the current iteration (when compared to its previous iteration), then correspondingly, the Z_{PLLBP} values would also improve (non-decreasing in our case) in the current iteration (when compared to its previous iteration). □

Proposition 2: *The solution obtained from the PLLBP will converge to optimal solution when the Lagrange multipliers* ($LM_{l,m}$) *are assigned with very large values.*

Proof When the Lagrange multipliers ($LM_{l,m}$) are assigned with very large values, then the solver would force all the relaxed constraints to be feasible so that the respective violation quantities $(z_{l,m})$ are zero at the optimal solution (in the PLLBP); and since all the constraints are feasible to the original problem, then the corresponding optimal solution from the PLLBP is also optimal to the original problem. □

In the generic LR (discussed in section 3), in order to obtain the lower bound on the makespan, relaxed constraints are taken into the objective function $(\sum_{l=2}^{N} \sum_{m=2}^{M} LM_{l,m} \times ((\propto_{l-1,m} + \sum_{n=1}^{N} (p_{n,m} \times y_{n,l-1})) - \propto_{l,m}))$. It is to be noted that there is no bound on the value for the variable which controls the start time of the last job on last machine ($\propto_{N,M}$). Since this is a minimization problem, the solver assigns a large value to $\propto_{N,M}$ in order to minimize the Z value, thus resulting in an unbounded solution to the optimization problem. To overcome unbounded condition, we constrain $\propto_{l,m}$ to be less than or equal to 100, and observe the possible fluctuating (i.e., oscillating) behavior of Z_{LB} values (see table 12). This is not unexpected. In the literature with the tardiness objective, the solution oscillation is overcome by introducing a quadratic penalty term to the objective function (e.g., see Liu *et al* [37] and Czerwinski and Luh [54]). However for the generic LR technique with the makespan objective, we are unable to overcome the solution oscillation. Hence we have focused on developing variants of LR techniques (PLLBP and ALR).

We also observe that the PLLBP gives improved lower bounds for the problem instances considered in the study. In order to carry out our experiments, we use ILOG CPLEX Solver (Version 12.8) on a computing facility with Intel Xeon 2.4 GHz (2 processors) and 64 GB RAM; and we define/set the values for the following parameters, while solving the problem instances in CPLEX solver as follows.

- EPGAP: We have set the value as 0 for this parameter, and this parameter mainly sets the tolerance value for the gap between the best integer value obtained so far and the best objective value of the remaining node (unexplored) in the tree.
- MIPSearch: We have used the default setting for the search strategy, and this helps CPLEX to use the best strategy among 'dynamic search' or 'conventional branch and cut' based on the characteristics of the model and the problem instance under consideration.
- TiLim: We have used this parameter to control the CPU time while solving the respective model/instance under consideration (e.g. up to 10 minutes in the case of LB_{PLLBP} and LB_{ALR} for a given LM setting, and 60 minutes for $LB_{Ext\text{-}PLLBP}$ and $LB_{Ext\text{-}ALR}$ for the chosen LM setting).

We have also carried out additional experimentations to show that our PLLBP is a good approach. The additional experimentations are reported in Section 6.

6. Additional experimentations

We consider a set of benchmark problem instances both from Taillard and VRF problem instances across different problem sizes to carry out the additional experimentation. To evaluate the efficiency of our Lagrangian Relaxation method 1 (PLLBP), we develop one more Lagrangian Relaxation method, Alternate Lagrangian Relaxation method 1 (ALR). The method is given below.

6.1 *Alternate Lagrangian Relaxation Method 1 (ALR)*

In our PLLBP, we consider constraint (7), and variable $z_{l,m}$ is introduced in constraint (7) and added to the objective function along with the corresponding Lagrange multiplier; whereas in ALR we consider constraint (8), retain constraint (7); and the variable $z_{l,m}$ is introduced in constraint (8) and is taken into the objective function with the associated Lagrange multiplier. In ALR, $z_{l,m}$ takes the value of, $\left(\alpha_{l,m-1} + \sum_{n=1}^{N}\left(p_{n,m-1} \times y_{n,l}\right)\right) - \alpha_{l,m}$, when the respective expression results in a positive value, and zero otherwise. The formulation of ALR is given below.

Minimize

$$Z_{ALR} = \alpha_{N,M} + \sum_{n=1}^{N}\left(p_{n,M} \times y_{n,N}\right) + \sum_{l=2}^{N}\sum_{m=2}^{M}\left(LM_{l,m} \times z_{l,m}\right) \tag{23}$$

subject to the following:

(2), (3), (4), (5), (6), (7), (9), (10), (11), (20), (21) and

$$z_{l,m} \geq \left(\alpha_{l,m-1} + \sum_{n=1}^{N}\left(p_{n,m-1} \times y_{n,l}\right)\right) - \alpha_{l,m}, \quad \text{for } 2 \leq m \leq M, 2 \leq l \leq N. \tag{24}$$

For analysis, ALR is coded in C ++ and executed on a computing facility with Intel Xeon 2.4 GHz (2 processors) and 64 GB RAM using ILOG CPLEX Solver (Version 12.8). The experimentation is carried out on the previously identified set of Taillard benchmark problem instances. It is observed that from the seven instances, PLLBP performs better than ALR for six instances (see table 8).

To do more analysis, we consider our Ext-PLLBP, MILP (see section 2) and the LP-relaxation model (called LP model – by considering the MILP model and with all binary variables relaxed as continuous variables in the interval [0, 1]), and executed for one hour each (maximum CPU time limit). At the end of one hour of execution, the lower bounds returned by the CPLEX solver for the MILP and LP models are noted. It is observed that for two instances all the models are giving the same solutions, while in most of the cases our Ext-PLLBP is giving better results. The computational results are presented in tables 9 and 10. Even though we have not carried out the work exhaustively, from the fifteen samples considered, it is observed that for nine problem instances, $LB_{\text{Ext-PLLBP}}$ gives better lower bounds than LB_{MILP} and LB_{LP}. It is to be noted that for ten instances LB_{MILP} is inferior to $LB_{\text{Ext-PLLBP}}$; for twelve instances LB_{LP} is inferior to $LB_{\text{Ext-PLLBP}}$; and only for one problem instance $LB_{\text{Ext-PLLBP}}$ is inferior to LB_{MILP}.

These observations bring out the main contribution of our paper in terms of obtaining tighter lower bounds, in general, that are superior to the lower bounds obtained from the truncated execution (after 1 hour of execution) of MILP model; though we cannot always claim this because the lower bounds from our proposed Lagrangian relaxation based procedure are not necessarily better for all problem instances, especially for large-sized jobs more than 500. We observe that this aspect necessitates further research work in future.

7. Limitations

Although it is evident from the tables that our Lagrangian Relaxation Method 1 (PLLBP) gives encouraging results, the method has certain limitations. One of the major limitation is that we are not solving PLLBP to optimality and we are truncating the execution of PLLBP after reaching a CPU time of 10 minutes. Hence we are not in a position to study the relationship between the objective function of LB_{PLLBP} and Lagrangian multipliers, given the limitation on the computational facility available with us. It is also evident that we are not guaranteed of getting an optimal solution when we execute the proposed PLLBP with the restriction of CPU time for a given LM value. It is possible that if Ext-PLLBP had been run for more than one hour, we would have got better results, but it could not be done due to operational constraints in the place of research work.

8. Conclusions and future scope

Even though it is seen in literature that Lagrangian Relaxation is an efficient method to find a lower bound, not much research has been carried out in this domain for obtaining lower bound for N-job M-machine permutation flowshop scheduling problems. Hence through our work, we have addressed this gap by considering the PFS problem with the makespan objective by using Lagrangian Relaxation technique to obtain lower bound. First, we have focused on the Lagrangian Relaxation method based on the sub-gradient approach, called LLBP. Since the approach is

not found to be good, we have developed two Lagrangian relaxation methods, called PLLBP and ALR, to find better lower bounds on the makespan for Taillard and VRF problem instances. To evaluate the efficiency of the LR procedures, we have utilized Taillard [49] and VRF [50] benchmark problem instances. It is seen that for all 120 Taillard problem instances, except for one instance, the proposed method gives better lower bounds than the corresponding values reported in the literature. For VRF problem instances, better lower bounds are obtained for all the 240 small-sized and medium-sized problem instances, and for most of the 240 large-sized problem instances. We are not able to execute problem instances beyond 500 jobs and 20 machines due to operational constraints. As stated earlier, we do not claim that the proposed procedures are computationally the best to obtain lower bounds on the makespan; our primary purpose is to obtain and report tight lower bounds on the makespan. Hence as future work, we suggest developing efficient algorithms to find lower bounds on the desired objective function for any size problem instance to be obtained in polynomial time. We also suggest exploring the different variants of the LR procedure to obtain tight lower bounds for different problem sizes with restricted computational effort.

Acknowledgement

We are thankful to the reviewers and the Editor for their valuable comments and suggestions to improve our manuscript.

Appendix A: Bounding Strategies Proposed by Kumar *et al* [29]

Parameters used

$\tau_{l,m}$ Processing time of job, based on the SPT order, placed in position l on machine m.
$B^1_{l,m}$ Bound 1 on the completion time of job placed in position l on machine m
$B^2_{l,m}$ Bound 2 on the completion time of job placed in position l on machine m
$B^3_{l,m}$ Bound 3 on the completion time of job placed in position l on machine m
$B^4_{l,m}$ Bound 4 on the completion time of job placed in position l on machine m

The expression for the lower bound on completion time of the job, placed in all positions on each machine, is adapted from Kumar *et al* [29]. Equations (25) and (26) present the expressions for finding the lower bound on completion time of job placed in, position l on the first machine and the first position on machine m, respectively. Equations (27), (28), (29) and (30) present the four bounds on completion time of the job placed in position l on machine m developed by Kumar *et al* [29]. Equation (31) presents the lower bound on completion time of the job placed in position l on machine m.

$$\gamma_{l,1} = \sum_{l'=1}^{l'=l} \tau_{l',1}, \quad \text{for} \quad 1 \le l \le N \tag{25}$$

$$\gamma_{1,m} = min_{1 \le n \le N} \left\{ \sum_{m'=1}^{m'=m} p_{n,m'} \right\}, \quad \text{for} \quad 1 \le m \le M \tag{26}$$

$$B^1_{l,m} = max_{1 \le m' \le m} \left\{ \gamma_{(l-1),m'} + min_{1 \le n \le N} \left\{ \sum_{m''=m'}^{m''=m} p_{n,m''} \right\} \right\}, \quad \text{for} \quad 2 \le l \le N, 2 \le m \le M \tag{27}$$

$$B^2_{l,m} = max_{1 \le m' \le (m-1)} \left\{ \gamma_{l,m'} + min_{1 \le n \le N} \left\{ \sum_{m''=(m'+1)}^{m''=m} p_{n,m''} \right\} \right\}, \quad \text{for} \quad 2 \le l \le N, 2 \le m \le M \tag{28}$$

$$B^3_{l,m} = min_{1 \le n \le N} \left\{ \sum_{m'=1}^{m'=(m-1)} p_{n,m'} \right\} + \sum_{l'=1}^{l'=l} \tau_{l',m}, \quad \text{for} \quad 2 \le l \le N, 2 \le m \le M \tag{29}$$

$$B^4_{l,m} = min_{1 \le n \le N} \left\{ \sum_{m'=1}^{m'=m} p_{n,m'} \right\} + \sum_{l'=1}^{l'=l-1} \tau_{l',m}, \quad \text{for} \quad 2 \le l \le N, 2 \le m \le M \tag{30}$$

$$\gamma_{l,m} = max\left(B^1_{l,m}, B^2_{l,m}, B^3_{l,m} B^4_{l,m}\right), \quad \text{for} \quad 2 \le l \le N, 2 \le m \le M \tag{31}$$

Appendix B: Discussion on the Generic LR Technique, LLBP (with $\propto_{l,m} \le 100$)

We consider a small-sized sample problem with 4 jobs and 4 machines (processing time matrix is given in table 11), and experiment with $LM_{l,m}$= 0.0001 and constrained $\propto_{l,m}$ to be less than or equal to 100, and execute the MILP model. It is observed that the lower bound value (Z_{LB}) given by LLBP is oscillating and eventually after 5562 iterations, the solver terminates the program due to lack of memory space. The details of the analysis are given below.

It is observed from table 12 that as iterations proceed, Z_{LB} value is oscillating. This is mainly due to the (negative) relaxed terms (see tables 13 and 16) and Lagrange multiplier terms. The values of variables (start time of job processed in position l on machine m) obtained after the first iteration of the LR procedure and after 5562th iteration of the LR procedure, are reported in tables 15 and 18

Table 11. Processing time matrix of sample problem instance $(N = 4, M = 4)$.

	m			
n	1	2	3	4
1	10	15	20	20
2	25	5	15	10
3	15	10	5	20
4	10	15	20	15

Table 12. Value of Z_{LB}, for the given iteration of the LR procedure (by constraining $\propto_{l,m} \leq 100$) with initial LM setting, $LM_{l,m}$= 0.0001 (for $2 \leq l \leq N, 2 \leq m \leq M$) for the sample problem with $N = 4$ and $M = 4$.

Iterations	Z_{LB}
1	94.999
2	94.513
3	76.120
4	92.688
5	89.352
25	97.692
50	96.396
100	98.353
500	98.276
1000	98.488
2000	93.324
3000	97.349
4000	92.584
5000	98.197
5562	95.716

Table 13. Value of relaxed term, $(\propto_{l-1,m} + \sum_{n=1}^{N}(p_{n,m} \times y_{n,l-1})) - \propto_{l,m}$, for $2 \leq l \leq N, 2 \leq m \leq M$, obtained after the first iteration of the LR procedure (by constraining $\propto_{l,m} \leq 100$) with initial LM setting, $LM_{l,m}$= 0.0001(for $2 \leq l \leq N, 2 \leq m \leq M$) for the problem with $N = 4$ and $M = 4$.

	m			
n	1	2	3	4
1	–	–	–	–
2	–	−40	−50	−50
3	–	15	20	15
4	–	15	30	35

Table 14. Value of $LM_{l,m}$, for $2 \leq l \leq N, 2 \leq m \leq M$, obtained after the first iteration of the LR procedure (by constraining $\propto_{l,m} \leq 100$) with initial LM setting, $LM_{l,m}$= 0.0001 (for $2 \leq l \leq N, 2 \leq m \leq M$) for the problem with $N = 4$ and $M = 4$.

	m			
l	1	2	3	4
1	–	–	–	–
2	–	0	0	0
3	–	0.0154	0.0205	0.0154
4	–	0.0154	0.0307	0.0358

Table 15. Value of $\propto_{l,m}$, for $1 \leq l \leq N, 1 \leq m \leq M$, obtained after the first iteration of the LR procedure (by constraining $\propto_{l,m} \leq 100$) with initial LM setting, $LM_{l,m}$= 0.0001 (for $2 \leq l \leq N, 2 \leq m \leq M$) for the problem with $N = 4$ and $M = 4$.

	m			
l	1	2	3	4
1	0	15	25	30
2	15	65	80	100
3	25	65	80	100
4	35	65	70	85

Table 16. Value of relaxed term, $(\propto_{l-1,m} + \sum_{n=1}^{N}(p_{n,m} \times y_{n,l-1})) - \propto_{l,m}$, for $2 \leq l \leq N, 2 \leq m \leq M$, for the 5562th iteration of the LR procedure (by constraining $\propto_{l,m} \leq 100$) with initial LM setting, $LM_{l,m}$= 0.0001 (for $2 \leq l \leq N, 2 \leq m \leq M$) for the problem with $N = 4$ and $M = 4$.

	m			
n	1	2	3	4
1	–	–	–	–
2	–	0	5	21
3	–	−30	−35	−36
4	–	15	30	30

Table 17. Value of $LM_{l,m}$, for $2 \le l \le N, 2 \le m \le M$, obtained for the 5562th iteration of the LR procedure (by constraining $\propto_{l,m} \le 100$) with initial LM setting, $LM_{l,m}$= 0.0001 (for $2 \le l \le N, 2 \le m \le M$) for the problem with $N = 4$ and $M = 4$.

	m			
l	1	2	3	4
1	–	–	–	–
2	–	0.0100	0.0079	0.4731
3	–	0	0.2542	0.6560
4	–	0.0236	0.2652	0.7731

Table 18. Value of $\propto_{l,m}$, for $1 \le l \le N, 1 \le m \le M$, obtained for the 5562th iteration of the LR procedure (by constraining $\propto_{l,m} \le 100$) with initial LM setting, $LM_{l,m}$= 0.0001 (for $2 \le l \le N, 2 \le m \le M$) for the problem with $N = 4$ and $M = 4$.

	m			
l	1	2	3	4
1	0	10	25	45
2	10	25	40	44
3	25	65	80	100
4	35	65	70	85

respectively. It is observed from the tables 14 and 17 that as iterations proceeds, LM values are updated. While updating the Lagrange multipliers (see equation (16)), it either takes zero (if the value is negative) or positive value. Thus, the Lagrange multiplier terms along with the negative relaxed term values contribute to the oscillation of the objective function of the generic LR procedure with $\propto_{l,m} \le 100$. In the literature it is reported that, for flowshop scheduling problems with the objective of minimizing tardiness, the solution oscillation is addressed by adding quadratic penalty terms in the objective function (e.g., see Liu *et al* [37] and Czerwinski and Luh [54]). But for the generic LR technique (LLBP) with the objective of minimizing makespan, we are unable to overcome the oscillating behaviour of the solution. Hence we focus on developing variants of LR techniques (PLLBP and ALR).

References

[1] Johnson S M 1954 Optimal two-and three-stage production schedules with setup times included. *Nav. Res. Logist. Q.* 1: 61–68

[2] Pinedo M 1995 *Scheduling: theory, algorithms and applications.* Englewood Cliffs, New Jersey: Prentice Hall

[3] Garey M R, Johnson D S and Sethi R 1976 The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* 1: 117-129

[4] Wagner H M 1959 An integer linear programming model for machine scheduling. *Nav. Res. Logist. Q.* 6: 131-140

[5] Manne A S 1960 On the jobshop scheduling problem. *Oper. Res.* 8: 219-223

[6] Stafford E F, Tseng F T and Gupta J N D 2005 Comparative evaluation of MILP flowshop models. *J. Oper. Res. Soc.* 56: 88-101

[7] Wilson J M 1989 Alternative formulations of a flowshop scheduling problem. *J. Oper. Res. Soc.* 40: 395-399

[8] Tseng F T and Stafford E F 2001 Two MILP models for the N × M SDST flowshop sequencing problem. *Int. J. Prod. Res.* 39: 1777-1809

[9] Stafford E F and Tseng F T 2002 Two models for a family of flowshop sequencing problems. *Eur. J. Oper. Res.* 142: 282-293

[10] Stafford E F and Tseng F T 1990 On the Srikar-Ghosh MILP model for the iVx M SDST flowshop problem. *Int. J. Prod. Res.* 28: 1817-1830

[11] Liao C J and You C T 1992 An improved formulation for the jobshop scheduling problem. *J. Oper. Res. Soc.* 43: 1047-1054

[12] Pan C H 1997 A study of integer programming formulations for scheduling problems. *Int. J. Syst. Sci.* 28: 33-41

[13] Tseng F T, Stafford E F and Gupta J N D 2004 An empirical analysis of integer programming formulations for the permutation flowshop. *Omega* 32: 285-293

[14] Bautista-Valhondo J and Alfaro-Pozo R 2018 Mixed integer linear programming models for flowshop scheduling with a demand plan of job types. *Cent. Eur. J. Oper. Res.* 23: 5–23

[15] Xu Z, Xu D, He J, Wang Q, Liu A and Xiao J 2018 Mixed integer programming formulations for two-machine flowshop scheduling with an availability constraint. *Arab. J. Sci. Eng*. 43: 777-788

[16] Ignall E and Schrage L 1965 Application of the branch and bound technique to some flowshop scheduling problems. *Oper. Res.* 13: 400-412

[17] Lomnicki Z A 1965 A "branch and bound" algorithm for the exact solution of the three-machine scheduling problem. *J. Oper. Res. Soc.* 16: 89-100

[18] Land A H and Doig A G 1960 An automatic method of solving discrete programming problems. *Econometrica: J. Econ. Soc*. 28: 497-520

[19] Little J D C, Murty K G, Sweeney D W and Karel C 1963 An algorithm for the traveling salesman problem. *Oper. Res.* 11: 972-989

[20] Brown A P G and Lomnicki Z A 1966 Some applications of the "branch and bound" algorithm to the machine scheduling problem. *J. Oper. Res. Soc.*17: 173-186

[21] McMahon G B and Burton P G 1967 Flowshop scheduling with the branch and bound method. *Oper. Res.* 15: 473-481

[22] Baker K R 1975 A comparative study of flowshop algorithms. *Oper. Res.* 23: 62-73

[23] Bestwick P F and Hastings N A J 1976 A new bound for machine scheduling. *J. Oper. Res. Soc.* 27: 479-487

[24] Lageweg B J, Lenstra J K and Rinnooy Kan A H G 1978 A general bounding scheme for the permutation flowshop problem. *Oper. Res.* 26: 53-67

[25] Potts C N 1980 An adaptive branching rule for the permutation flowshop problem. *Eur. J. Oper. Res*. 5: 19-25

[26] Carlier J and Rebai I 1996 Two branch and bound algorithms for the permutation flowshop problem. *Eur. J. Oper. Res.* 90: 238-251

[27] Chung C S, Flynn J and Kirca O 2002 A branch and bound algorithm to minimize the total flowtime for M-machine permutation flowshop problems. *Int. J. Prod. Econ.* 79: 185-196

[28] Madhushini N, Rajendran C and Deepa Y 2009 Branch and bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs. *J. Oper. Res. Soc.* 60: 991-1004

[29] Kumar S S, Rajendran C and Leisten R (in press) Bounding strategies for obtaining lower bound for N-job and M-machine flowshop scheduling problem with objective of minimizing the total flowtime of jobs, *Int. J. Oper. Res.*

[30] Campbell H G, Dudek R A and Smith M L 1970 A heuristic algorithm for the N-job, M-machine sequencing problem. *Manage. Sci.* 16: 630-637

[31] Nawaz M, Enscore E E and Ham I 1983 A heuristic algorithm for the M-machine, N-job flowshop sequencing problem *Omega* 11: 91–95

[32] Rajendran C 1994 A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria. *Int. J. Prod. Res,* 32: 2541-2558

[33] Liu J and Reeves C R 2001 Constructive and composite heuristic solutions to the P//$\sum$ Ci scheduling problem. *Eur. J. Oper. Res.* 132: 439-452

[34] Rossi F L, Nagano M S and Neto R F T 2016 Evaluation of high performance constructive heuristics for the flowshop with makespan minimization. *Int. J. Adv. Manuf. Technol.* 87: 125-136

[35] Fernandez-Viagas V, Molina-Pariente J M and Framinan J M 2018 New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Syst. Appl.* 114: 345-356

[36] Pan Q K, Gao L, Wang L, Liang J and Li X Y 2019 Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Syst. Appl.* 124: 309-324

[37] Liu G, Luh P B and Resch R 1997 Scheduling permutation flowshops using the Lagrangian relaxation technique. *Ann. Oper. Res.* 70: 171-189

[38] Akkan C and Karabati S 2004 The two-machine flowshop total completion time problem: Improved lower bounds and a branch and bound algorithm. *Eur. J. Oper. Res.* 159: 420-429

[39] Hoogeveen H, Van Norden L and Van de Velde S 2006 Lower bounds for minimizing total completion time in a two-machine flowshop. *J. Sched.* 9: 559-568

[40] Hamdi I and Loukil T 2014 Lagrangian relaxation for the permutation flowshop scheduling problem with minimal and maximal time lags. In: *2014 International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 42-47

[41] Fisher M L 1973 Optimal solution of scheduling problems using Lagrange multipliers: Part I. *Oper. Res.* 21: 1114-1127

[42] Guignard M and Kim S 1987 Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Math. Program.* 39: 215-228

[43] Tang L, Xuan H and Liu J 2006 A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. *Comput. Oper. Res.* 33: 3344-3359

[44] Bazaraa M S and Goode J J 1977 The traveling salesman problem: A duality approach. *Math. Program.* 13: 221-237

[45] Fisher M L 1981 The Lagrangian relaxation method for solving integer programming problems. *Manage Sci.* 27: 1-18

[46] Beasley J E 1993 Lagrangian relaxation. In: Reeves C R (Ed.) *Modern heuristic techniques for combinatorial problems.* Blackwell Scientific Publications, Oxford, pp. 243–303

[47] Kumar P N R and Narendran T T 2011 On the usage of Lagrangean Relaxation for the convoy movement problem. *J. Oper. Res. Soc.* 62: 722-728

[48] Beasley J E and Christofides N 1989 An algorithm for the resource constrained shortest path problem. *Networks* 19: 379-394

[49] Taillard E 1993 Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* 64: 278-285

[50] Vallada E, Ruiz R and Framinan J M 2015 New hard benchmark for flowshop scheduling problems minimising makespan. *Eur. J. Oper. Res.* 240: 666-677

[51] Held M and Karp R M 1970 The traveling salesman problem and minimum spanning trees. *Oper. Res.* 18: 1138-1162

[52] Held M and Karp R M 1971 The traveling salesman problem and minimum spanning trees: Part II. *Math. Program.*1: 6-25

[53] Chakravorty A and Laha D 2017 A heuristically directed immune algorithm to minimize makespan and total flowtime in permutation flowshops. *Int. J. Adv. Manuf. Technol.* 93: 3759–3776

[54] Czerwinski C S and Luh P B 1994 Scheduling products with bills of materials using an improved Lagrangian relaxation technique *IEEE Trans. Robotics and Automation*. 10: 99–111