Available online at www.sciencedirect.com
ScienceDirect

Electronic Notes in Discrete Mathematics 36 (2010) 1057–1064

Electronic Notes in DISCRETE MATHEMATICS

www.elsevier.com/locate/endm

# Extending the Single Machine-Based Relaxation Scheme for the Job Shop Scheduling Problem

Anis Gharbi [1,2]

*Princess Fatimah Al-Nijriss Research Chair for Advanced Manufacturing Technologies, Department of Industrial Engineering, College of Engineering King Saud University, Riyadh, Saudi Arabia*

Mohamed Labidi [3]

*ROI - Combinatorial Optimization Research Group Ecole Polytechnique de Tunisie, La Marsa, Tunisia*

## Abstract

The contribution of this paper to the job shop related literature is twofold. First, we provide an efficient way forf solving the job shop scheduling problem with release dates, delivery times and delayed precedence constraints. It is shown that the latter problem is equivalent to a classical job shop with precedence constraints. Second, an effective extension of the standard single machine-based relaxation scheme is derived. Preliminary computational results conducted on a set of benchmark instances show that effective multiple machine-based lower bounds can be computed within reasonable CPU time.

*Keywords:* Lower Bounds, Release Dates, Delivery Times, Delayed Precedence Constraints.


doi:10.1016/j.endm.2010.05.134

# 1 Introduction

We address the job shop scheduling problem, denoted by $J||C_{max}$, where $n$ jobs $(J_1, \ldots, J_n)$ have to be scheduled without preemption on $m$ machines $M_1, \ldots, M_m$, with the objective of minimizing the makespan. We denote by $O_{ij}$ the operation of job $J_j$ that has to be processed on machine $M_i$, and by $p_{ij}$ its processing time. Each job is characterized by its proper routing on the machines.

Numerous heuristics have been proposed for this $\mathcal{NP}$-hard problem. However, only few exact branch-and-bound algorithms have been published. It is well-known that the performance of branch-and-bound algorithms strongly relies on the effectiveness/efficiency of the implemented lower bounds. Unfortunately, the lack of effective relaxation schemes seems to be the Achilles' heel in job shop-related research. Indeed, the overwhelming majority of the published lower bounds are based on the single machine relaxation scheme. Although this latter scheme has been enhanced by various adjustment and propagation techniques, the best results obtained so far for open benchmark instances still fail to close the gap.

This paper constitutes a first attempt to extend the single machine-based relaxation scheme to the multiple machines variant. We propose an efficient procedure to solve the multiple machines subproblem with release dates, delivery times, and delayed precedence constraints. The instance is nontrivially transformed into a (classical) $J|prec|C_{\max}$ one with an expanded number of jobs and machines, but with a maximum of three operations per job. Interestingly, the obtained instance proves to be efficiently solvable by the branch-and-bound algorithm of Brucker et al. [1]. Indeed, our preliminary computational results show that such instances with more than 180 jobs and 120 machines are optimally solved within an average CPU time of 0.12 seconds.

# 2 The single machine subproblem

A classical and widely used job shop relaxation scheme consists in relaxing the constraint that each machine can process at most one job at a time, for all machines but one (say $M_k$). The obtained problem is a single machine

[1] The first author is grateful to Princess Fatimah Al Nijriss Research Chair for Advanced Manufacturing Technologies for providing financial support.

[2] Email: a.gharbi@ksu.edu.sa

[3] Email: med.labidi@gmail.com

problem (denoted by $1|r_j, q_j|C_{max}$) with release dates and delivery times. The release date $r_j$ of a job $j$ is equal to the length of the longest path from the source node to node $O_{kj}$ in the disjunctive graph. Similarly, the delivery time $q_j$ is such that $p_{kj} + q_j$ is equal to the length of the longest path from the $O_{kj}$ node to the sink node in the disjunctive graph. Although the latter problem is strongly $\mathcal{NP}$-hard, its optimal makespan, denoted hereafter by $C_0^*(M_k)$, can be efficiently computed by using effective existing branch-and-bound algorithms [2], [4]. This yields the standard single machine-based lower bound:

$$LB_0 = max_{k=1,...,m} C_0^*(M_k)$$

Interestingly, several procedures exist which aim at identifying precedence relationships between operations that have to be scheduled on the same machine in a $J||C_{\max}$ instance (see for instance [1]). An immediate output of using these techniques is that delayed precedence constraints may occur between operations on the same machine. Indeed, let $L_{ij}$ denote the length of the longest path between operations $O_{ki}$ and $O_{kj}$ in the disjunctive graph. Then, $O_{kj}$ cannot start processing before $a_{ij} = L_{ij} - p_{ki}$ units of time after the completion of $O_{ki}$. Dauzère-Pérès [3] devised an efficient branch-and-bound algorithm for solving the single machine problem with heads, tails and delayed precedence constraints (denoted by $1|r_j, q_j, del - prec|C_{\max}$). Let $C_1^*(M_k)$ denote the optimal makespan of the $1|r_j, q_j, del - prec|C_{\max}$ defined on $M_k$. A valid lower bound for the $J||C_{\max}$ that dominates $LB_0$ is

$$LB_1 = max_{k=1,...,m} C_1^*(M_k)$$

## 3 The multiple machines subproblem

Consider the relaxation scheme which consists in relaxing the capacity constraint of all the machines except for a subset of $K$ machines (say for simplicity $M_1, M_2, ..., M_K$). The obtained problem is a $J_K|r_j, q_j, del - prec|C_{\max}$ where the release dates $r_j$, the delivery times $q_j$, and the time lags $a_{ij}$ are computed in a similar way than it is described in the previous section. Let $\Delta$ denote the set of delayed precedence constraints. For the sake of clarity, we will denote w.l.o.g. by $O_1, O_2, ..., O_T$ the operations that have to be processed on the non relaxed machines $M_1, M_2, ..., M_K$. Interestingly, we state the following result:

**Theorem 1:** The $J|r_j, q_j, del - prec|C_{\max}$ is equivalent to the $J|prec|C_{\max}$.

**Proof:** Let $(P_1)$ denote a $J|r_j, q_j, del - prec|C_{\max}$ instance. In the following, we show how to construct a $J|prec|C_{\max}$ instance, denoted hereafter by $(P_2)$, which is equivalent to $(P_1)$. The set of machines of $(P_2)$ includes machines $M_1, M_2, ..., M_K$. Moreover, to each operation $O_i$ $(i = 1, ..., T)$ in $(P_1)$ are associated two machines $M_i^p$ and $M_i^s$ in $(P_2)$. Machine $M_i^p$ has to process all the operations that precede $O_i$, and machine $M_i^s$ has to process all the operations that succeed to $O_i$. In addition, to each delayed precedence constraint between operations $O_i$ and $O_j$ in $(P_1)$ is associated a machine $D_{ij}$. The set of jobs of $(P_2)$ is partitioned into the four following subsets:

- Subset 1 ($T$ jobs): each job is composed of three operations $O_i^p, O_i$, and $O_i^s$ $(i = 1, ..., T)$ which have to be processed (in that order) on machines $M_i^p, M_{(O_i)}$, and $M_j^s$, respectively, where $M_{(O_i)} \in \{M_1, M_2, ..., M_K\}$ denotes the machine on which operation $O_i$ has to be processed in $(P_1)$. Operations $O_i^p$ and $O_i^s$ have zero processing times while $O_i$ requires $p_i$ units of processing time.
- Subset 2 ($|\Delta|$ jobs): each job consists of three operations $\delta_{ij}^p$, $\delta_{ij}$, and $\delta_{ij}^s$ $((O_i, O_j) \in \Delta)$ which have to be processed on machines $M_i^s, D_{ij}$ and $M_j^p$, respectively. Machine $D_{ij}$ has to process only operation $\delta_{ij}$ for $a_{ij}$ units of time. Operations $\delta_{ij}^p$ and $\delta_{ij}^s$ have zero processing times. Moreover, $\delta_{ij}^p$ has to succeed to $O_i^s$, and $\delta_{ij}^s$ has to precede $O_j^p$.
- Subset 3 ($T$ jobs): each job consists of a single operation $\alpha_i$ $(i = 1, ..., T)$ which has to be processed before all operations on machine $M_i^p$ for $r_i$ units of time.
- Subset 4 ($T$ jobs): each job consists of a single operation $\beta_i$ $(i = 1, ..., T)$ which has to be processed after all operations on machine $M_i^s$ for $q_i$ units of time.

It suffices to prove that in an optimal (active) schedule of $(P_2)$, the start time of operation $O_i$ is equal to $\max\{r_i, \max_{(O_j,O_i)\in\Delta}(C(O_j) + a_{ji})\}$ (where $C(.)$ stems for completion time), and operation $\beta_i$ starts processing at $C(O_i)$ (which expresses the delivery time constraint). The remainder of the proof is omitted for lack of available space.

An immediate consequence of Theorem 1 is that the $J|r_j, q_j, del-prec|C_{\max}$ can be optimally solved by applying a branch-and-bound algorithm on the equivalent $J|prec|C_{\max}$. In our implementation, the latter problem is solved by using the branch-and-bound algorithm of Brucker et al. [1]. Although the latter algorithm is initially designed for the $J||C_{\max}$, it can be immediately adapted to the $J|prec|C_{\max}$. Indeed, the disjunctive graph of the $J|prec|C_{\max}$

is no more than that of the $J||C_{\max}$ where some of the disjunctive arcs are fixed.

The reader may have remarked the huge size of the obtained $J|prec|C_{\max}$ instance ($3T + |\Delta|$ jobs and $2T + K + |\Delta|$ machines). On the other hand, it should be noticed that each job has a maximum number of 3 operations to be performed, which substantially reduces the computational burden. Interestingly, the problem size can be fairly reduced by considering the non-delayed precedence constraints. That is, for a given $(O_i, O_j) \in \Delta$ such that $a_{ij} = 0$, there is no need for operation $\delta_{ij}$ neither for machine $D_{ij}$ (i.e. the corresponding subset 2 job has only operations $\delta^p_{ij}$ and $\delta^s_{ij}$). If, in addition, $O_i$ and $O_j$ are processed on the same machine, then operations $\delta^p_{ij}$ and $\delta^s_{ij}$ can be removed and replaced by a conjunctive arc from $O_i$ to $O_j$. Furthermore, the following results (stated with omitted proofs) consistently improve the performance of the optimization procedure. The first two lemmas show that the only disjunctive arcs that need to be fixed by the branch-and-bound algorithm are those which are related to machines $M_1, M_2, ..., M_K$; while the third one proves that the preemptive single machine-based lower bound $C_0^{(p)}(.)$ (which is implemented in the branch-and-bound algorithm of [1]) only needs to be computed over machines $M_1, M_2, ..., M_K$.

**Lemma 2:** Any sequence of operations $\delta^s_{ji}$ ($(O_j, O_i) \in \Delta$) on $M^p_i$ is optimal ($i = 1, ..., T$).

**Lemma 3:** Any sequence of operations $\delta^p_{ij}$ ($(O_i, O_j) \in \Delta$) on $M^s_i$ is optimal ($i = 1, ..., T$).

**Lemma 4:** $\max\{C_0^{(p)}(M^p_i), C_0^{(p)}(M^s_i), \max_{(O_i,O_j)\in\Delta}(C_0^{(p)}(D_{ij}))\} \leq C_0^{(p)}(M_{(O_i)})$ for $i = 1, ..., T$.

**Example 1:** Figure 1 displays the disjunctive graph of the $J|prec|C_{\max}$ that is derived from a 5 operation-2 machine $J|r_j, q_j, del-prec|C_{\max}$ instance, where $\Delta = \{(O_1, O_2), (O_1, O_3), (O_3, O_4), (O_5, O_4)\}$. For the sake of clarity, the source node and some of the sink-related conjunctive arcs are not displayed.

For a given value of $K$, let $\pi_h$ ($h = 1, ..., \binom{K}{m}$) denote the $h^{th}$ $J_K|r_j, q_j, del-prec|C_{\max}$ instance that is derived from the $J_m||C_{\max}$ one, and let $C^*(\pi_h)$ denote its optimal makespan. Therefore, a valid $K$-machine-based lower bound for the $J_m||C_{\max}$ is:

$$LB_K = \max_{h=1,...,\binom{K}{m}} C^*(\pi_h)$$

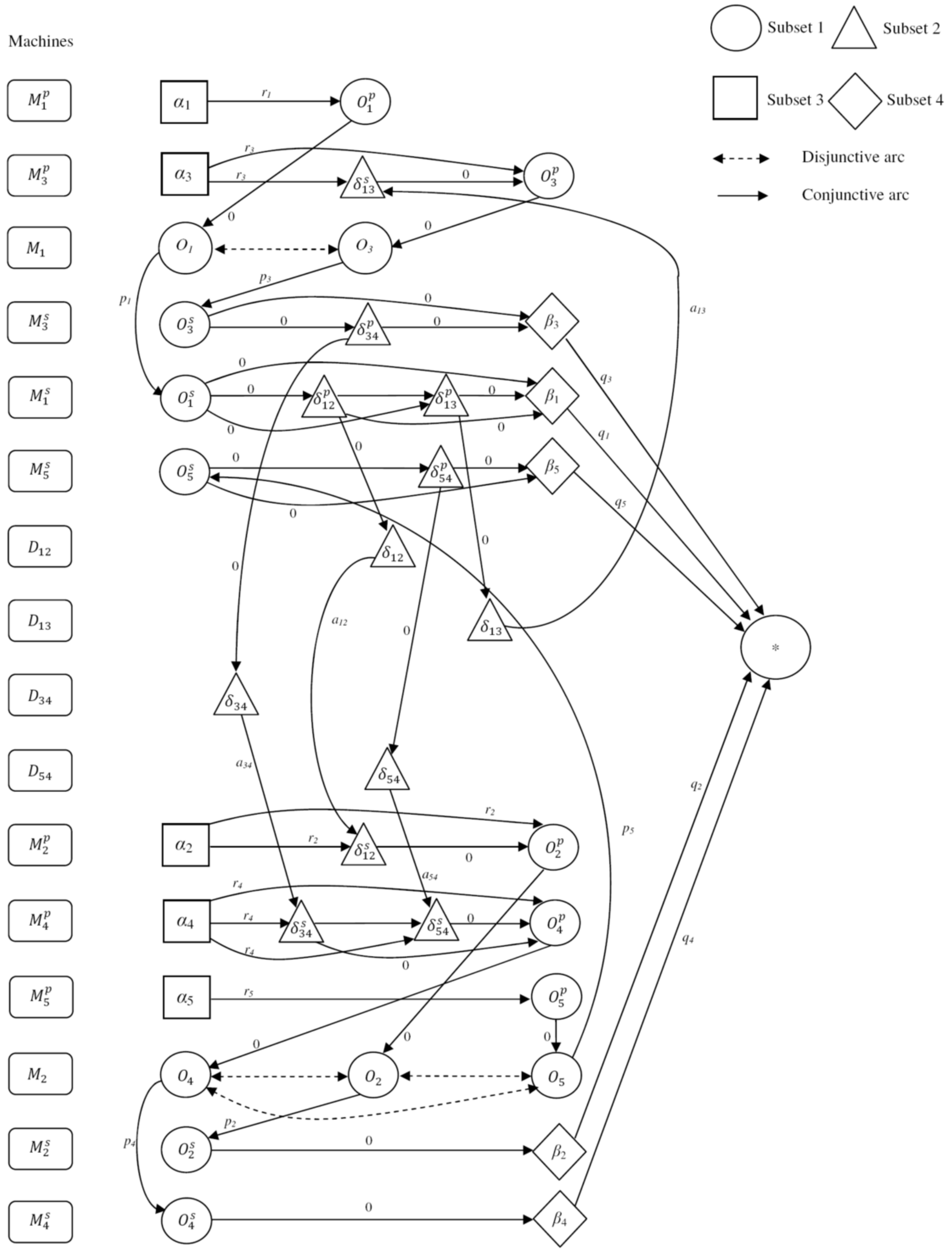


Figure 1. The $J|prec|C_{max}$ disjunctive graph of Example 1.

In our implementation, $LB_K$ ($K \geq 2$) is computed as follows: Let $C$ denote a trial value on the optimal makespan of the $J||C_{\max}$. Initially, the value of $C$ is set to $LB_{K-1}$. For each value of $C$, a deadline $d_{ij} = C - q_{ij}$ is set for each operation $O_{ij}$ and the adjustment procedure described in [1] is applied in order to identify precedence constraints. Then, the branch-and-bound algorithm of Brucker et al. [1] is run for each $\pi_h$ (starting from $h = 1$) after setting at the root node an artificial upper bound $UB = C + 1$. The branch-and-bound algorithm is stopped as soon as a feasible schedule with makespan less than $UB$ has been found. In that case, the computation of $C^*(\pi_h)$ will not be useful in finding a lower bound better than $C$, and $\pi_h$ need not to be solved for larger values of $C$. Otherwise, we will have $C^*(\pi_h) = UB$ i.e. there is no feasible schedule of $\pi_h$ with makespan less than or equal to $C$. Therefore, $C$ is incremented by one unit and the procedure is restarted from $\pi_h$. The procedure is stopped, yielding $LB_K = C$, when a feasible schedule with makespan less than or equal to $C$ has been found for $\pi_{\binom{K}{m}}$.

## 4 Preliminary computational results

The effectiveness/efficiency of the implemented multiple machine-based lower bounds (namely $LB_2$, $LB_3$ and $LB_4$) with respect to the best single machine-based one (namely $LB_1$) is evaluated on a set of selected benchmark instances [5] for which $LB_1$ is not equal to the optimal makespan. The results are depicted in Table 1 where we provide, for each lower bound $LB_K \in \{LB_2, LB_3, LB_4\}$:

- $Gap_{red}$ : the percentage by which the gap between $LB_1$ and the optimal/best found makespan ($C_{best}$) has been reduced i.e.

  $$Gap_{red} = 100(LB_K - LB_1)/(C_{best} - LB_1)$$

  The bold values denote the fact that $LB_K > LB_{K-1}$.
- $Time$ : the total CPU time on a Quad 2.8 GHz Personal Computer with 4 GB RAM (including the CPU time of $LB_{K-1}$). The value between parentheses denotes the average CPU time that is required for solving a single $J_K|r_j, q_j, del - prec|C_{\max}$ subproblem.

Table 1 provides evidence that the multiple machine-based lower bounds consistently outperform the single machine-based one, while requiring a (surprisingly) moderate CPU time. For the sake of illustration, we observe that, for all the largest instances (20 jobs and 15 machines), the four machine-based lower bound is computed in about one minute, although it requires

| | $LB_2$ | | $LB_3$ | | $LB_4$ | |
|---|---|---|---|---|---|---|
| ***Instance (nxm)*** | $Gap_{Red}$ | *Time* | $Gap_{Red}$ | *Time* | $Gap_{Red}$ | *Time* |
| ***ABZ5*** (10x10) | **48.06** | 0.18 (-) | **51.46** | 0.48 (-) | **70.87** | 1.69 (-) |
| ***ABZ6*** (10x10) | **50.93** | 0.14 (-) | **54.63** | 0.44 (-) | **74.07** | 1.33 (-) |
| ***ABZ7*** (20x15) | 0.00 | 0.36 (-) | 0.00 | 2.96 (-) | 0.00 | 22.31 (0.01) |
| ***ABZ8*** (20x15) | **25.00** | 0.74 (-) | **27.94** | 4.78 (-) | **33.82** | 69.06 (0.04) |
| ***ABZ9*** (20x15) | **22.22** | 0.42 (-) | 22.22 | 4.02 (-) | **30.16** | 51.26 (0.03) |
| ***FT6*** (6x6) | **100.00** | 0.02 (-) | 100.00 | 0.05 (-) | 100.00 | 0.07 (-) |
| ***FT10*** (10x10) | **49.18** | 0.14 (-) | **54.10** | 0.47 (-) | **72.95** | 2.26 (-) |
| ***La4*** (10x5) | **69.57** | 0.03 (-) | 69.57 | 0.06 (-) | **91.30** | 0.14 (-) |
| ***La16*** (10x10) | **48.57** | 0.11 (-) | **65.71** | 0.50 (-) | **80.00** | 1.25 (-) |
| ***La17*** (10x10) | **81.08** | 0.11 (-) | 81.08 | 0.37 (-) | 81.08 | 1.02 (-) |
| ***La18*** (10x10) | **69.23** | 0.17 (-) | 69.23 | 0.37 (-) | 69.23 | 0.91 (-) |
| ***La19*** (10x10) | **36.84** | 0.13 (-) | **44.36** | 0.48 (-) | **57.14** | 1.49 (-) |
| ***La20*** (10x10) | **46.32** | 0.12 (-) | **49.47** | 0.43 (-) | **54.74** | 1.32 (-) |
| ***La21*** (15x10) | **74.51** | 0.15 (-) | 74.51 | 1.10 (-) | 74.51 | 4.15 (0.01) |
| ***La22*** (15x10) | 0.00 | 0.09 (-) | 0.00 | 0.45 (-) | **57.14** | 5.26 (0.02) |
| ***La24*** (15x10) | **14.81** | 0.13 (-) | **16.67** | 0.78 (-) | **25.93** | 3.75 (0.01) |
| ***La25*** (15x10) | **43.37** | 0.28 (-) | **49.40** | 1.02 (-) | **61.45** | 11.30 (0.04) |
| ***La29*** (20x10) | **13.16** | 0.13 (-) | 13.16 | 1.25 (-) | 13.16 | 95.49 (0.44) |
| ***La36*** (15x15) | **20.45** | 0.27 (-) | **52.27** | 2.25 (-) | **59.09** | 11.24 (-) |
| ***La38*** (15x15) | **25.21** | 0.35 (-) | **35.29** | 2.63 (-) | **42.02** | 16.87 (0.01) |
| ***La39*** (15x15) | 0.00 | 0.28 (-) | 0.00 | 2.43 (-) | 0.00 | 17.21 (0.01) |
| ***La40*** (15x15) | **42.31** | 0.38 (-) | 42.31 | 2.86 (-) | **53.85** | 21.52 (0.01) |
| ***Average*** | **40.04** | 0.21 (-) | **44.24** | 1.37 (-) | **54.66** | 15.50 (0.03) |

(-) means that the CPU time is less than 0.01 sec.

Table 1. Performance of the multiple machine-based lower bounds

solving more than 1365 $J_4|r_j, q_j, del - prec|C_{\max}$ subproblems. Finally, it is worth noting that the $J_K|r_j, q_j, del-prec|C_{\max}$ subproblems are solved within an extremely short average CPU time, frequently less than 0.01 seconds and scarcely exceeding 0.03 seconds.

## References


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

[2] Carlier, J. (1982). The one-machine sequencing problem. European Journal of Operational Research, 11, 42-47.

[3] Dauzère-Pérès, S. (1995). A procedure for the one-machine sequencing problem with dependent jobs. European Journal of Operational Research, 81, 579-589.

[4] Gharbi A., Labidi M. (2010). Jackson's Semi-Preemptive Scheduling on a Single Machine, to appear in Computers & Operations Research.

[5] A.S. Jain, S. Meeran (1999). Deterministic job-shop scheduling: Past, present and future, European Journal of Operational Research, 113, 390-434.