Computers & Operations Research 39 (2012) 1652–1660

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Two branch-and-bound algorithms for the robust parallel machine scheduling problem

Mohammad Ranjbar [a,*], Morteza Davari [a], Roel Leus [b]

[a] Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Research group ORSTAT, Faculty of Business and Economics, KULeuven, Belgium

ARTICLE INFO

Available online 29 September 2011

Keywords:
Robust scheduling
Identical parallel machines
Stochastic processing times

ABSTRACT

Uncertainty is an inevitable element in many practical production planning and scheduling environments. When a due date is predetermined for performing a set of jobs for a customer, production managers are often concerned with establishing a schedule with the highest possible confidence of meeting the due date. In this paper, we study the problem of scheduling a given number of jobs on a specified number of identical parallel machines when the processing time of each job is stochastic. Our goal is to find a robust schedule that maximizes the customer service level, which is the probability of the makespan not exceeding the due date. We develop two branch-and-bound algorithms for finding an optimal solution; the two algorithms differ mainly in their branching scheme. We generate a set of benchmark instances and compare the performance of the algorithms based on this dataset.

1. Introduction

The literature on parallel machine scheduling has developed over multiple decades, and contains many useful models and algorithms (see, for instance, [1–3]). One of the most frequently studied objectives in scheduling identical parallel machines is to minimize the makespan, i.e. the maximum completion time of the jobs. This problem is denoted as $Pm\|C_{max}$ in the notation of Graham et al. [4], and has been shown to be NP-hard [5]. Several heuristic and exact procedures have been proposed to construct workable baseline schedules that solve this problem [6]. One drawback, however, is that $Pm\|C_{max}$ is inherently deterministic: all problem parameters (among which, the processing times) are supposed to be known with certainty.

However, the usefulness of a scheduling algorithm depends on the system parameters of the shop floor. In many practical production planning and scheduling environments, uncertainty is an inevitable element: ex-ante developed production plans will be confronted with multiple disturbances during implementation, and the system's parameters can change in real time. If such changes are significant and not considered, the proposed "optimal" schedules may turn out to be quite poor in practice. The most significant changing parameters are new unscheduled arrivals of jobs, resource breakdowns and variability in the task durations. The unavoidable time lag between the development of a schedule and its implementation only reinforces the reduction of the effectiveness of deterministic schedules. Most of these phenomena can be modeled by uncertainty in job processing times, since the scheduling objective function is usually a function of the processing times.

In this paper, we address an identical parallel machine scheduling problem with stochastic processing times and a predetermined due date for the makespan that is imposed by the customer. Our goal is to find a *robust* schedule that maximizes the customer service level, which is the probability of the makespan not exceeding the due date. In the remainder of this text, we refer to this problem as the *robust parallel machine scheduling problem* (RPMSP). The intuition underlying the problem statement is that many production managers and schedulers should be concerned with the construction of a baseline schedule with maximum "trust" that relevant constraints will not be violated. In order to obtain an optimal solution to the RPMSP, we develop two branch-and-bound algorithms (B&B) that differ in their branching schemes.

To the best of our knowledge, the RPMSP is a new problem, which has not yet been studied in the literature. There are two areas in the scheduling literature, however, that contain related material: deterministic identical parallel machine scheduling and stochastic scheduling.

Many studies have been published on identical parallel machine problems, but most of them pertain to a static environment, in which all parameters are known in advance [7,8]. Over the last two decades, several exact and heuristic procedures have been proposed for $Pm\|C_{max}$, which is the most classic and frequently studied

* Correspondence to: Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, P.O. Box 91775-1111, Iran. Tel.: +98 511 8805092.
E-mail addresses: m_ranjbar@um.ac.ir (M. Ranjbar), mo_da26@stu-mail.um.ac.ir (M. Davari), Roel.Leus@econ.kuleuven.be (R. Leus).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.09.019

problem in this field. Dell'Amico and Martello [9] present a fast B&B algorithm for this problem, which tends to find optimal solutions in a very short time. Dell'Amico and Martello exploit the relationship between $Pm\|C_{max}$ and the bin packing problem to develop efficient bounds. Mokotoff [10] formulates $Pm\|C_{max}$ as a mixed integer program (MIP) and solves it using a cutting-plane scheme. Computational experiments by Dell'Amico and Martello [11] demonstrate that the latter MIP-based approach is consistently outperformed by their earlier-developed branch-and-bound algorithm (in [9]). Haouari and Jemmali [12] propose a B&B algorithm with tight bounds for $Pm\|C_{max}$ in which the machines are sequentially loaded whereas in the Dell'Amico and Martello's algorithm, the machines are loaded simultaneously. Van Den Akker et al. [13] apply column generation to solve $Pm\|C_{max}$; a decomposition algorithm based on column generation is also proposed for a just-in-time parallel machine scheduling problem by Cheng and Powell [14]. In addition to exact algorithms, several heuristic procedures have been developed for $Pm\|C_{max}$; the best performing ones are surveyed in Dell'Amico et al. [6]. Since B&B outperforms other exact algorithms in the literature for $Pm\|C_{max}$, in this text we choose this optimization framework as the basis for a solution procedure for RPMSP.

Multiple articles have been published on the stochastic parallel machine scheduling problem with objectives such as minimization of makespan, minimization of total completion time and due date related objectives (see Chapter 12 of Pinedo [15]). *Robust* or *proactive* scheduling can be defined as scheduling with a degree of anticipation of variability during the schedule's execution [16]. In this article, we operationalize this anticipation of variability by maximizing the customer service level. We are aware of a few papers that look into robust scheduling of a single machine, but the literature on robust parallel machine scheduling is almost void. The only related work stems from Anglani et al. [17], who investigate robust scheduling of parallel machines with sequence-dependent set-up costs. In contrast to our paper, in which the uncertainty is captured by stochastic variables, they represent the uncertainty by means of fuzzy numbers. The objective of Anglani et al. [17] is the minimization of the set-up costs. The work proposed in the current article is an extension of the research papers of Daniels and Carrillo [18] and Wu et al. [19], in which a so-called "β-robust" scheduling model is developed for a single machine. Both these references search for a schedule that minimizes the risk that the flow time exceeds a given threshold.

The contributions of this article are threefold: (1) we provide the first description of the RPMSP and present a non-linear formulation; (2) we develop two exact depth-first B&B algorithms, which differ mainly in their branching strategy; and (3) we derive upper and lower bounds as well as dominance rules to accelerate the B&B procedures.

The remainder of this paper is organized as follows. In Section 2, we provide a formal description of the problem and state a non-linear binary formulation. Section 3 describes the branching schemes, while Sections 4 and 5 are devoted to the bounds and dominance rules, respectively. Computational experiments are reported in Section 6 and our main findings are discussed in Section 7. Finally, conclusions and suggestions for future work are presented in Section 8.

2. Problem description and formulation

We define the problem RPMSP as follows: there are n independent jobs gathered in set $J=\{J_1, J_2, \ldots, J_n\}$ and a set of m identical machines $M=\{M_1, M_2, \ldots, M_m\}$. Each job j (also written as J_j, for short) has a stochastic processing time p_j with normal distribution function $f(p_j)$, with expectation $E(p_j)=\mu_j$ and variance $Var(p_j)=\sigma_j^2$ $(j=1,2,\ldots,n)$. We assume that neither release dates nor due dates are imposed for any individual job but that a due date δ is put forward by the customer for the completion of the entire job set. The machines are always available and the processing of each job can be done by any of the machines. We define the customer service level P as the probability that the makespan does not exceed the due date δ and we aim to find a schedule with maximum customer service level.

Clearly, inserting idle time between jobs on any machine never improves the objective function, so we only consider solutions in which each machine starts its work from time zero and processes its subset of jobs without interruption. The order in which the jobs assigned to the same machine are processed does not influence the objective, and therefore we can equate a solution with a partition of the job set into m subsets (one subset per machine); we represent an (ordered) subset of jobs assigned to M_i by $\mathbf{a}_i$. The total processing time Y_i on M_i has a normal distribution: $Y_i \sim N[\mu_{M_i}, \sigma^2_{M_i}]$, with $\mu_{M_i}=\sum_{j\in \mathbf{a}_i}\mu_j$ and $\sigma^2_{M_i}=\sum_{j\in \mathbf{a}_i}\sigma^2_j$. The customer service level on M_i is written as $\pi_i=P(Y_i\le\delta)$. An arbitrary normal distribution can be converted to a standard normal distribution by changing variables to $Z_i=(y_i-\mu_{M_i})/\sigma_{M_i}$, so the service level for M_i becomes $\pi_i=P(Z_i\le z_i)=\phi(z_i)$, where $Z_i=(\delta-\mu_{M_i})/\sigma_{M_i}$ and ϕ represents the cumulative standard normal distribution function.

We introduce the following decision variables:

$$X_{ij}=\begin{cases}1 & \text{if } J_j \text{ is assigned to } M_i\\ 0 & \text{otherwise}\end{cases} \quad i=1,\ldots n;\ j=1,\ldots,m$$

RPMSP can now be formulated as the following non-linear binary model:

$$Max\ \mathrm{P}=\prod_{i=1}^{m}\phi\left(\frac{\delta-\mu_{M_i}}{\sigma_{M_i}}\right) \tag{1}$$

subject to

$$\sum_{i=1}^{m}X_{ij}=1 \quad j=1,\ldots,n \tag{2}$$

$$\mu_{M_i}=\sum_{j=1}^{n}\mu_j X_{ij} \quad i=1,\ldots,m \tag{3}$$

$$\sigma^2_{M_i}=\sum_{j=1}^{n}\sigma^2_j X_{ij} \quad i=1,\ldots,m \tag{4}$$

$$X_{ij}\in\{0,1\} \quad i=1,\ldots,m;\ j=1,\ldots,n \tag{5}$$

We have $\mathrm{P}=P(\max\{Y_1,\ldots,Y_m\}\le\delta)$ and since $Y_1,\ldots,Y_m$ are independent variables, we can rewrite P as $\mathrm{P}=P(Y_1\le\delta,\ldots, Y_m\le\delta)=P(Y_1\le\delta)\ldots P(Y_m\le\delta)$. Consequently, we can write the objective function (1) as $\prod_{i=1}^{m}\pi_i$, which maximizes the customer service level. Constraint (2) assures that each job is assigned to exactly one machine. Constraints (3) and (4) compute the mean and the variance of total processing time on each machine, which are used to determine P. Constraint (5) states that all the decision variables are binary.

3. Solution representation and branching schemes

We develop two depth-first B&B algorithms to find an optimal solution to the RPMSP. These two B&B algorithms are both implemented with a backtracking strategy but they differ in their branching schemes. We identify the two algorithms as B&B1 and B&B2. In B&B1, the machines are sequentially loaded while in B&B2, the machines are simultaneously loaded. For $Pm\|C_{max}$, branching schemes following the overall logic of B&B1 and

B&B2 were implemented by Dell'Amico and Martello [7] and Haouari and Jemmali [12], respectively.

3.1. Example

Existing algorithms for $Pm\|C_{max}$ can obviously not be applied "as is" for solving the RPMSP: an optimal solution to $Pm\|C_{max}$ may not be optimal for the RPMSP. The following example illustrates this issue, where the mean processing time of jobs in the RPMSP is set as the fixed processing time in $Pm\|C_{max}$. In this example, $n=10$, $m=4$ and $\delta=59$; the remaining job characteristics are shown in Table 1. The parameters were chosen such that the sets of optimal solutions to $Pm\|C_{max}$ and RPMSP are not the same. This instance will also be used in Sections 3 and 4 for further illustrations.

Table 1
Mean and variance of the processing times.

Job number	1	2	3	4	5	6	7	8	9	10
Mean (min)	24	22	23	20	21	21	20	19	19	20
Variance $(\text{min})^2$	23	32	9	27	16	14	14	15	13	4

An optimal solution to the RPMSP instance with these parameters has $\mathbf{a}_1=\{J_1,J_6\}$, $\mathbf{a}_2=\{J_2,J_3\}$, $\mathbf{a}_3=\{J_4,J_5,J_7\}$ and $\mathbf{a}_4=\{J_8,J_9,J_{10}\}$; the attained customer service level is

$$P^* = \pi_1^*\pi_2^*\pi_3^*\pi_4^* = \phi((59-45)/\sqrt{37})\phi((59-45)/\sqrt{41})$$
$$\phi((59-61)/\sqrt{57})\phi((59-58)/\sqrt{32}) = 0.220$$

where the symbols P^* and π_i^* represent the optimal value of P and the corresponding values for π_i, $i=1,2,3,4$, respectively. Machine M_3 has the highest mean processing time in this solution, namely $\mu_{M_3}=61$. Next, we consider the following job-machine assignment: $\mathbf{a}_1=\{J_1,J_2\}$, $\mathbf{a}_2=\{J_3,J_{10}\}$, $\mathbf{a}_3=\{J_4,J_5,J_8\}$ and $\mathbf{a}_4=\{J_6,J_7,J_9\}$. This solution is an optimum solution for $Pm\|C_{max}$ with $C_{max}^*=60$, but the customer service level is lower than optimal in RPMSP:$P=\pi_1\pi_2\pi_3\pi_4=\phi((59-46)/\sqrt{55})\ \phi((59-43)/\sqrt{13})\ \phi((59-60)/\sqrt{58})\ \phi((59-60)/\sqrt{41})=0.188$.

3.2. Solution representation

Haouari and Jemmali [12] represent a feasible schedule for $Pm\|C_{max}$ as a permutation of n jobs; we modify this representation for our algorithms. Our solution encoding is based on the concept of symmetry breaking developed by Sherali and Smith [20]. Consider an arbitrary schedule for the RPMSP. We can obtain $m!$ equivalent solutions by exchanging the assigned jobs of each pair of machines. Also, for each M_i, if $\mathbf{a}_i$ is not an ordered subset then the number of equivalent solutions becomes $m!*\prod_{i=1}^{m}|\mathbf{a}_i|!$, where $|\mathbf{a}_i|$ is the number of jobs assigned to M_i. Thus, a B&B procedure can become captured in a time consuming and largely redundant search process because it risks needing to generate and evaluate numerous equivalent solutions. In order to avoid this phenomenon, we represent each set of alternative symmetric solutions by a unique permutation of the n jobs, based on the ideas developed by Haouari and Jemmali [12]. In this representation, a permutation $\mathbf{a}_i=(a_i^1,a_i^2,\ldots,a_i^{|\mathbf{a}_i|})$ is associated with each machine ($i=1,\ldots,m$), where a_i^k is the kth job assigned to M_i, and each solution S is represented by a permutation $\mathbf{A}(S)=(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_m)$. Such a permutation is valid if it satisfies the following conditions:

$C1$) Each ordered subset $\mathbf{a}_i$ is an increasing list of the job indices, so $a_i^k<a_i^{k+1}$ for $k=1,\ldots,|\mathbf{a}_i|-1$ and $i=1,\ldots,m$. We assume that the jobs j are indexed in non-increasing order of the values $\theta_j=\mu_j+\sigma_j$, with ties broken arbitrary. This property also holds for the example instance presented at the start of Section 3.

$C2$) The machines are indexed in increasing order of the index of their first job. Thus, $a_i^1<a_{i+1}^1$ for $i=1,\ldots,m-1$.

3.3. B&B1

In our first algorithm, the machines are loaded sequentially and M_1, having J_1 as its first job, is loaded first. The root node N_0 (at level zero) of the search tree corresponds to the empty permutation and each node N_l^r at level $l\geq 1$ of the search tree corresponds to a partial valid permutation of l jobs on r machines, represented as $\mathbf{A}(N_l^r)=(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_r)$, where $1\leq r\leq m$ and $|\mathbf{a}_1|+|\mathbf{a}_2|+\ldots+|\mathbf{a}_r|=l$. In this permutation, the job assignments to machines $1,\ldots,r-1$ are completed and no more jobs will be added to these machines. As a result, each child node of N_l^r corresponds to appending a job from the set of unscheduled jobs $\bar{J}(N_l^r)=J\backslash(\mathbf{a}_1\cup\mathbf{a}_2\cup\ldots\cup\mathbf{a}_r)$ to either M_r or to M_{r+1}, unless $r=m$, in which case all members of $\bar{J}$ must be assigned to M_m. Nodes with the same parent are created in order of increasing job index of the new job to be assigned.

We represent the set of children of node N_l^r by $C_{N_l^r}$, and we partition this set into two subsets $C_{N_l^r}^1$ and $C_{N_l^r}^2$, where $C_{N_l^r}^1$, resp. $C_{N_l^r}^2$, represents the child nodes in which the next job will be assigned to M_r, resp. to M_{r+1}. Based on ($C1$), in the generation of $C_{N_l^r}^1$ we only consider jobs with an index greater than the previously loaded job. Moreover, from ($C1$) and ($C2$), $C_{N_l^r}^2$ includes only one job, with the lowest index among the unscheduled jobs. When backtracking, we save the best value of $\prod_{i=r+1}^{m}\pi_i$ in node N_l^r and denote it by $\pi^*_{C_{N_l^r}}$.

The branching scheme of procedure B&B1 is illustrated in Fig. 1 for the example instance introduced in Section 3.1. The figure represents the different branching choices at level $l=2$ when previously J_1 and J_6 were assigned to M_1 at levels 1 and 2, respectively. These two assignments were established in the two nodes 1 and 2 (the number written inside each node is the node number). At level 3, nodes 3 to 6 belong to $C_{N_l^r}^1$ while node 7 belongs to $C_{N_l^r}^2$.

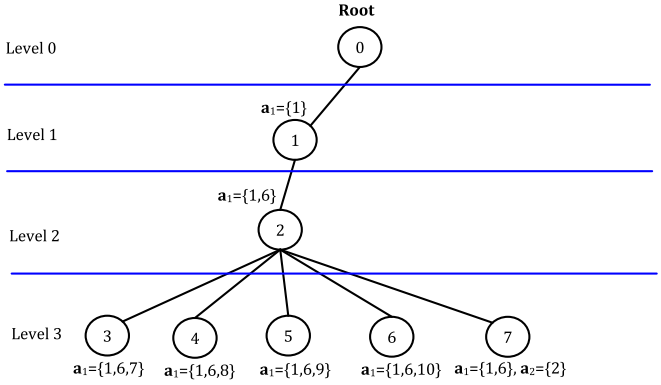

Fig. 1. Illustration of B&B1.

3.4. B&B2

In the second algorithm, the jobs are assigned to machines by increasing index and machines are loaded simultaneously. At level l of the search tree, each node N_l has $\overline{m}\leq m$ child nodes, obtained by assigning job $l+1$ to M_i ($i=1,\ldots,m$), to be scanned in increasing order of i. Node N_l corresponds to a partial valid permutation of jobs $1,2,\ldots,l$, represented as $\mathbf{A}(N_l)=(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_m)$, where $|\mathbf{a}_1|+|\mathbf{a}_2|+\ldots+|\mathbf{a}_m|=l$.

In B&B2, since the jobs are assigned to machines by increasing index, (*C*1) automatically holds. In each job assignment, we divide the set of machines into two subsets, containing all empty (with no assigned job) and all non-empty (with at least one job) machines, respectively. Based on (*C*2), each unscheduled job should be assigned either to one of the non-empty machines or to the empty machine with the smallest index.

The branching scheme of procedure B&B2 is illustrated in Fig. 2, for the same instance as before. We consider the situation where $J_1$is assigned to M_1 at level 1 and J_2 is assigned to M_2 at level 2. The children of node 2 correspond to the three possible alternatives for assignment of J_3, i.e. assignment of J_3 to M_1, M_2 or M_3.

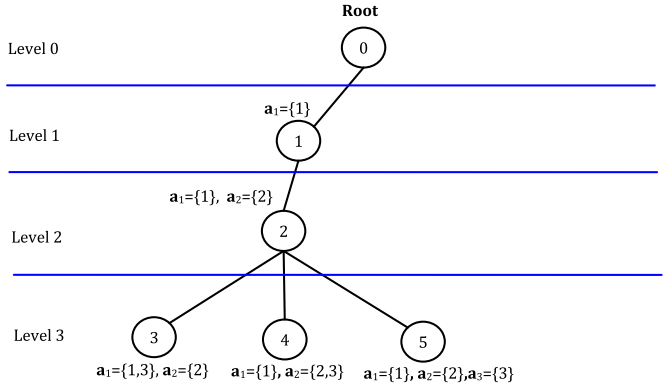

Fig. 2. Illustration of B&B2.

4. Bounds

In this section, we present a lower bound and an upper bound for the optimal objective value of the RPMSP, as well as a lower bound for the number of jobs assigned to each machine. Also, we develop procedures for improving the upper bound of the optimal solution and the lower bound of the number of jobs assigned to each machine.

4.1. Lower bound for the optimal objective value

We develop a heuristic procedure to create an initial solution, leading to a lower bound (*LB*) that can be computed in short running time. This initial solution produces a lower bound that can be used in both B&B1 and B&B2. Whenever a better solution is obtained during the search process, the *LB* will be updated. The procedure tries to balance the π values across the machines. The heuristic procedure is described in pseudo-code in Fig. 3 and runs in $O(mn)$ time.

If we apply this procedure to the example instance, we obtain $\mathbf{a}_1=\{J_1,J_8\}$, $\mathbf{a}_2=\{J_2,J_7,J_{10}\}$, $\mathbf{a}_3=\{J_3,J_6\}$ and $\mathbf{a}_4=\{J_4,J_5,J_9\}$. The customer service level for this solution is P=0.15.

4.2. Upper bound for the optimal objective value

The upper bound developed in this section is used at all levels of B&B1 and at the first level of B&B2. For each node N_l^r of B&B1, this upper bound is valid only for the child nodes corresponding to subset $C^2_{N_l^r}$ and not for $C^1_{N_l^r}$. Each node in the search tree of both procedures corresponds to a partial schedule. Thus, there is a set of scheduled jobs and a set of unscheduled jobs in each node. Consider a node in which there are $g \le n$ unscheduled jobs and $h \le m$ empty machines; these unscheduled jobs and empty machines in isolation constitute a smaller instance of the RPMSP. We seek to derive an upper bound $UB_1^{g,h}$ on the best objective value for this smaller instance, which is referred to as "sub-problem" below. Obviously, $UB_1^{n,m}$ is an upper bound for the original instance.

Let $\tilde{v}$ be a lower bound on the maximum number of jobs assigned to an empty machine. For now, we simply consider $\tilde{v}=g/h$ but $\tilde{v}$ may be improved by the lower bound presented in Section 4.3. Each job has two characteristics, namely the mean and variance of its processing time, and we will distinguish two sets of jobs in the computation of $UB_1^{g,h}$: the set of $\tilde{v}$ jobs with lowest means and the set of $\tilde{v}$ jobs with the lowest variances. Let $J_{[j]}$ be the job with jth lowest mean processing time and $J^{[j]}$ the job with jth lowest variance. In other words, $\mu_{J_{[1]}} \le \mu_{J_{[2]}} \le \ldots \le \mu_{J_{[n]}}$ and $\sigma^2_{J^{[1]}} \le \sigma^2_{J^{[2]}} \le \ldots \le \sigma^2_{J^{[n]}}$. Note that we only use information on the *number* of remaining jobs and machines in this upper bound, since sorting jobs based on both mean and variance in each node of B&B1 would be quite time consuming and lead to an inefficient algorithm. The construction of $UB_1^{g,h}$ is based on Proposition 1.

Proposition 1. Consider a node in the search tree with g unscheduled jobs and h empty machines and let $\tilde{v}=[g/h]$. If $\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}>0$, let $UB_1^{g,h}=\phi(\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}/\sqrt{\sum_{k=1}^{\tilde{v}}\sigma^2_{J^{[k]}}})$, otherwise $UB_1^{g,h}=\phi(\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}/\sqrt{\sum_{k=n-\tilde{v}+1}^{n}\sigma^2_{J^{[k]}}})$. The value $UB_1^{g,h}$ is an upper bound on the best objective value for the sub-problem corresponding to the node under consideration.

Proof. For each combination of $\tilde{v}$ unscheduled jobs, the sum of means of processing times is not less than $\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}$ and the sum of variances is not less than $\sum_{k=1}^{\tilde{v}}\sigma^2_{J^{[k]}}$. Thus, if $\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}>0$, $UB_1^{g,h}=\phi(\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}/\sqrt{\sum_{k=1}^{\tilde{v}}\sigma^2_{J^{[k]}}})$ is a bound. In the case $\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}<0$, the maximum of the function $\phi(\cdot)$ is obtained when the $\tilde{v}$ largest variances are considered in the denominator of the argument, so $UB_1^{g,h}=\phi(\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}/\sqrt{\sum_{k=n-\tilde{v}+1}^{n}\sigma^2_{J^{[k]}}})$. Obviously, when $\delta-\sum_{k=1}^{\tilde{v}}\mu_{J_{[k]}}=0$, $UB_1^{g,h}=0.5$.□

Intuitively, the calculation of $UB_1^{g,h}$ is based on the assignment of a maximum number of unscheduled jobs with shortest processing times to an empty single machine. For the example instance

1. Assign J_i to M_i for $i=1,\ldots,m$. Let $k=m$ and go to step 5.
2. Let $\mathbf{A}=\mathbf{a}_1\cup\mathbf{a}_2\ldots\cup\mathbf{a}_m$ be the set of assigned jobs and compute the average load per machine $\bar{\mu}=\frac{\sum_{j\in\mathbf{A}}\mu_j}{m}$.
3. For each M_i, let $\rho_i=\frac{\mu_{M_i}-\bar{\mu}}{\sigma_{M_i}}$.
4. Select J_k and assign it to the machine with the smallest value of ρ.
5. If $k\ge n$, then stop; otherwise, let $k=k+1$ and go to step 2.

Fig. 3. Pseudo-code of the initial solution.

and with $g=n=10$ and $h=m=4$, we obtain $\tilde{v}=[10/4]=3$ and $UB_1^{10,4}=\phi((59-(19+19+20))/(\sqrt{4+9+13}))=\phi(0.196)=0.577$.

$UB_1^{g,h}$ is computed using the workload of one machine and is very workable. We will additionally present an improved version of this upper bound in the next section by considering the minimum workload of $h-1$ other machines.

4.3. Lower bound for the number of jobs assigned to each machine

The result in this section is inspired by Proposition 3 of Dell'Amico and Martello [9] and entails the determination of a lower bound v^{min} on the number of jobs assigned to each machine in any optimal schedule.

Proposition 2. If $UB_1^{n-v,m-1}<LB$ then $v^{min}\geq v+1$ for any integer $\{1,2,\ldots,[n/m]\}$.

Proof. Consider any solution in which a machine, say M_1, has v or less tasks. The remaining $n-v$ jobs should then be assigned to the other $m-1$ machines and will achieve a service level of at most $UB_1^{n-v,m-1}$, and so the overall service level can never be equal to LB or more. □

If $v^{min}>\tilde{v}$ then we can update $\tilde{v}$ in Section 4.2 as $\tilde{v}=v^{min}$. Also, whenever the LB is updated, the value v^{min} may be updated, which may in turn lead to updating $\tilde{v}$ and the upper bounds.

For the example instance, since $UB_1^{9,3}=\phi((59-(19+19+20))/(\sqrt{4+9+13}))=0.577\not<0.24=LB$, we let $v^{min}=1$.

4.4. An improved upper bound on the optimal objective value

The upper bound developed in this section is an improved version of the upper bound described in Section 4.2. Similar to previous upper bound, for each node N_l^r of B&B1, this upper bound is valid only for subset $C_{N_l^r}^2$ and not for $C_{N_l^r}^1$. A sub-problem of the RPMSP is again considered, with g unscheduled jobs and h empty machines. It was explained in Section 4.2 that $UB_1^{g,h}$ is constructed based on the workload of a single machine, but if the minimum workload of the other $h-1$ empty machines is also taken into account, the bound may be improved. If $\tilde{v}$ jobs are assigned to a single machine then $g'=g-\tilde{v}$ jobs should be assigned to $h'=h-1$ machines, with $g'>h'$. Without loss of generality, we denote the unscheduled jobs by $J_1,\ldots,J_{g'}$ and the empty machines by $M_1,\ldots,M_{h'}$. For each M_i ($i=1,\ldots,h'$), let $\hat{\pi}_i$ be an upper bound on π_i^*, where $\hat{\pi}_i$ is computed based on the minimum workload of M_i. The minimum workload of each machine is the direct outcome of assigning the minimum possible number of jobs (v^{min}) to that machine. The improved upper bound is written as $UB_2^{g,h}$ and is computed as $UB_2^{g,h}=UB_1^{g,h}\prod_{i=1}^{h'}\hat{\pi}_i$.

The question that remains is which v^{min} jobs to assign to each machine. Intuitively, in an optimal solution one expects "large" jobs (with high mean or variance) to be assigned to machines with "small" jobs, and these jobs will be our target. We consider two cases. In both cases, we first assign one of the largest jobs to each machine. In the first case, we assign each of the jobs $J_{[g']}J_{[g'-1]},\ldots J_{[g'-h'+1]}$ to one of the h' empty machines, while in the second case, each of the jobs $J^{[g']},J^{[g'-1]},\ldots,J^{[g'-h'+1]}$ is allocated to one of the h' empty machines. Subsequently, in both cases, if $v^{min}>1$ then we consider $v^{min}-1$ remaining jobs for each machine, as described in Proposition 1.

In the first case, if we assume that job $J_{[g'-i+1]}$ with mean processing time $\mu_{J_{[g'-i+1]}}$ and variance $\sigma^2_{J_{[g'-i+1]}}$ is assigned to M_i, $i=1,\ldots,h'$, the minimum mean processing time on M_i is obtained when $v^{min}-1$ other jobs with lowest means are assigned to M_i. Since the calculations would otherwise become overly time consuming, these $v^{min}-1$jobs are not updated in each node based on the set of unscheduled jobs, but they are predetermined at the start of B&B1. As a result, we determine

$$\hat{\pi}_i=\phi\left(\frac{\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}}{\sqrt{\sigma^2_{J_{[g'-i+1]}}+\alpha}}\right),$$

where α depends on whether $\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}>0$ or $\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}\leq 0$. Similarly to Proposition 1, if $\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}>0$ then $\alpha=\sum_{k=1}^{v^{min}-1}\sigma^2_{J^{[k]}}$ whereas if $\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}\leq 0$ then $\alpha=\sum_{k=n-v^{min}+2}^{n}\sigma^2_{J^{[k]}}$. Consequently, we define ϕ_i^1 as follows:

$$\phi_i^1=\begin{cases}\phi\left(\dfrac{\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}}{\sqrt{\sigma^2_{J_{[g'-i+1]}}+\sum_{k=1}^{v^{min}-1}\sigma^2_{J[k]}}}\right); & \delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}>0\\ \phi\left(\dfrac{\delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}}{\sqrt{\sigma^2_{J_{[g'-i+1]}}+\sum_{k=n-v^{min}+2}^{n}\sigma^2_{J[k]}}}\right); & \delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}\leq 0.\end{cases}$$

In the second case, again assuming that job $J^{[g'-i+1]}$ with mean processing time $\mu_{J[g'-i+1]}$ and variance $\sigma^2_{J^{[g'-i+1]}}$ is assigned to M_i, $i=1,\ldots,h'$, ϕ_i^2 is defined as follows:

$$\phi_i^2=\begin{cases}\phi\left(\dfrac{\delta-\mu_{J[g'-i+1]}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}}{\sqrt{\sigma^2_{J[g'-i+1]}+\sum_{k=1}^{v^{min}-1}\sigma^2_{J[k]}}}\right); & \delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}>0\\ \phi\left(\dfrac{\delta-\mu_{J[g'-i+1]}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}}{\sqrt{\sigma^2_{J[g'-i+1]}+\sum_{k=n-v^{min}+2}^{n}\sigma^2_{J[k]}}}\right); & \delta-\mu_{J_{[g'-i+1]}}-\sum_{k=1}^{v^{min}-1}\mu_{J_{[k]}}\leq 0.\end{cases}$$

Finally, using the following proposition, we can find $\hat{\pi}_i$, $i=1,2,\ldots,h-1$.

Proposition 3. $\hat{\pi}_i=min\{\phi_i^1,\phi_i^2\}$.

Proof. Straightforward. □

For the example instance, if M_1 is the machine with the maximum number of jobs (resulting in $UB_1^{10,4}=0.577$), we have two cases for M_2,M_3 and M_4. In the first case, in which we consider the jobs with largest mean processing time, we assign J_1 to M_2, J_3 to M_3 and J_2 to M_4. In the second case, considering the jobs with largest variances, we assign J_2 to M_2, J_4 to M_3 and J_1 to M_4. For this example, we actually have $v^{min}=2$ (see Appendix) and therefore in both cases, we consider a (dummy) job with minimum mean and minimum variance of processing time as the second job of these three machines. Further calculations for this example are:

$$\phi_2^1=\phi\left(\frac{59-24-19}{\sqrt{23+4}}\right)=\phi(3.07)=0.999\quad \phi_2^2=\phi\left(\frac{59-22-19}{\sqrt{32+4}}\right)=\phi(3.00)=0.998$$

$$\phi_3^1=\phi\left(\frac{59-23-19}{\sqrt{9+4}}\right)=\phi(4.71)\approx 1\quad \phi_3^2=\phi\left(\frac{59-20-19}{\sqrt{27+4}}\right)=\phi(3.59)\approx 1$$

$$\phi_4^1=\phi\left(\frac{59-22-19}{\sqrt{32+4}}\right)=\phi(3.00)=0.998\quad \phi_4^2=\phi\left(\frac{59-24-19}{\sqrt{23+4}}\right)=\phi(3.07)=0.999$$

$\hat{\pi}_2=0.998$, $\hat{\pi}_3=1$, $\hat{\pi}_4=0.998$ and $UB_2^{8,3}=0.577*0.998*1*0.998=0.574$.

5. Dominance rules

5.1. Dominance rules for B&B1

Dominance rule 1. If in a node of the search tree, the customer service level is less than the *LB*, then the node can be fathomed.

Proof. Straightforward.□

This rule explains why we index jobs in non-increasing order of their θ values: since $\mathbf{a}_i$ is an ordered subset for each M_i, jobs with higher means and variances are assigned first, and so dominance rule 1 will fathom dominated solutions earlier.

For the example instance, consider a node N_4^1 where $\mathbf{a}_1=\{J_1, J_2, J_4, J_5\}$ and $LB=0.15$. For this partial solution, $\pi_1=\phi((59-87)/\sqrt{98})=0.002<0.150$, and so the node can be fathomed.

Dominance rule 2. At node N_l^r, if there is another node $N_l'^r$ such that $N_l'^r<N_l^r$ ($N_l'^r$ is explored before N_l^r), $\bar{J}(N_l^r)=\bar{J}(N_l'^r)$ and $\pi^*_{C_{N_l'^r}}\prod_{i=1}^r\pi_i<LB$, then the child node that corresponds to $C^2_{N_l^r}$ can be fathomed.

Proof. This dominance rule is applied whenever two nodes N_l^r and $N_l'^r$ have identical unscheduled jobs. Since $N_l'^r<N_l^r$ and our algorithm is a depth-first B&B, node $N_l'^r$ is visited first and the value of $\pi^*_{C_{N_l'^r}}$ is calculated. Afterwards, since we know the value of $\prod_{i=1}^r\pi_i$ at N_l^r and $\pi^*_{C_{N_l^r}}=\pi^*_{C_{N_l'^r}}$ (because $\bar{J}(N_l^r)=\bar{J}(N_l'^r)$), $\pi^*_{C_{N_l'^r}}\prod_{i=1}^r\pi_i$ is an upper bound for the objective value of all solutions obtainable from the child node corresponding to $C^2_{N_l^r}$. As a result, if this upper bound is less than the *LB*, then the child node can be fathomed.□

Using this rule, we can also update the *LB* in each node N_l^r when $\pi^*_{C_{N_l'^r}}\prod_{i=1}^r\pi_i>LB$.

For the example instance, consider node N_6^2 with $\mathbf{a}_1=\{J_1,J_5,J_8\}$, $\mathbf{a}_2=\{J_2,J_4\}$, $\bar{J}(N_6^2)=\{3,6,7,9,10\}$ and $\pi^*_{C_{N_6'^2}}=0.499$, and $N_6'^2$ with $\mathbf{a}_1=\{J_1,J_2\}$, $\mathbf{a}_2=\{J_4,\ J_5J_8\}$ and $\bar{J}(N_6'^2)=\{3,6,7,9,10\}$; currently $LB=0.216$. For the node N_6^2, we have $\pi_1=0.986$ and $\pi_2=0.248$. Since $\bar{J}(N_6^2)=\bar{J}(N_6'^2)$ and $\pi_1*\pi_2*\pi^*_{C_{N_6'^2}}=0.122<0.216$, the node pertaining to $C^2_{N_6^2}$ is fathomed.

Dominance rule 3. If in node N_l^r, $UB_2^{n-l,m-r}\prod_{i=1}^r\pi_i<LB$, then the child node corresponding to $C^2_{N_l^r}$ is fathomed.

Proof. Straightforward.□

For the example instance, consider $C^2_{N_3^1}$ with $\mathbf{a}_1=\{J_1,J_2,J_4\}$, $\pi_1=0.22$and $LB=0.216$. Because $v^{min}=2$ and $\tilde{v}=[7/3]=3$, we have $UB_1^{7,3}=\phi((59-(19+19+20))/(\sqrt{4+9+13}))=0.578$ (an upper bound for π_2),

$$\hat{\pi}_3=\min\left\{\phi\left(\frac{59-(24+19)}{\sqrt{23+4}}\right),\phi\left(\frac{59-(22+19)}{\sqrt{32+4}}\right)\right\}=0.998$$

$$\hat{\pi}_4=\min\left\{\phi\left(\frac{59-(23+19)}{\sqrt{9+4}}\right),\phi\left(\frac{59-(20+19)}{\sqrt{27+4}}\right)\right\}=1$$

and $UB_2^{7,3}=UB_1^{7,3}\prod_{i=3}^4\hat{\pi}_i=0.575$. As $UB_2^{7,3}\pi_1=0.127<LB$, the node is fathomed.

It should be noted that in our implementation, we first verify in each node the condition $UB_1^{n-l,m-r}\prod_{i=1}^r\pi_i<LB$ and if the node is not cut by this first condition then $UB_2^{n-l,m-r}$ is computed and the condition $UB_2^{n-l,m-r}\prod_{i=1}^r\pi_i<LB$ is tested.

Dominance rule 4. In each node N_l^r, let J^* be a dummy job with $\mu_{J^*}=\min_{J_j\in\bar{J}(N_l^r)}\{\mu_{J_j}\}$ and $\sigma^2_{J^*}=\min_{J_j\in\bar{J}(N_l^r)}\{\sigma^2_{J_j}\}$, and consider adding J^* to M_r at the next level of the search tree. If $\prod_{i=1}^r\pi_i<LB$, then all the child nodes corresponding to $C^1_{N_l^r}$ can be fathomed.

Proof. If we cannot add the dummy job to M_r due to dominance rule 1 then for any job from $\bar{J}(N_l^r)$ added to M_r, we will have $\prod_{i=1}^r\pi_i<LB$.□

For the example, consider a node at level 3 with $\mathbf{a}_1=\{J_1,J_2,J_4\}$ and $LB=0.15$. In the set $\bar{J}$, the lowest mean is 19 and the lowest variance is 4. If we add the corresponding dummy job to M_1, then $\pi_1=\phi((59-85)/\sqrt{86})=0.003<0.15$ and so we fathom the node.

5.2. Dominance rules for B&B2

Dominance rule 1. In each node N_l, if $\sum_{i\in\{1,\ldots,m\}:|\mathbf{a}_i|<v^{min}}(v^{min}-|\mathbf{a}_i|)>n-l$, then N_l is dominated.

Proof. For each M_i with $|\mathbf{a}_i|<v^{min}$, at least $v^{min}-|\mathbf{a}_i|$ jobs need to be added, otherwise the resulting solutions are dominated by *LB* through Proposition 2, and the number of remaining jobs is $(n-l)$.□

This dominance rule does not allow to fathom parts of the search tree in the example instance, but it can achieve efficiency gains for instances with more jobs and machines.

Dominance rule 2. Dominance rule 1 of B&B1.

Proof. Straightforward.□

In each node of the search tree of both B&B1 and B&B2, the dominance rules are applied in order of their numbering.

6. Computational results

6.1. Computational setup

All the procedures were coded in Visual C++.Net 2008; all computational experiments were performed on a laptop computer with Pentium IV 2.00 GHz processor, 3.00 GB of internal memory and a 32-bit operating system. In order to evaluate the performance of the algorithms B&B1 and B&B2, we consider five values for the number of jobs, $n=12,14,16,18$ and 20, and three values for the number of machines, $m=3,4$ and 5. For each job j, the value of μ_j is drawn from the normal distribution $N[20,9]$ and the value of σ_j^2 is drawn from the uniform distribution on interval $(0,0.1\mu_j^2]$. The uniform distribution for σ_j^2 is chosen so as to obtain non-negative processing times in virtually all cases. To evaluate the impact of the variance on the CPU time, we also consider $\sigma_j^2\in(0,0.1\eta\mu_j^2]$, for the three values 0.25, 0.5 and 0.75 for η. Three instances are generated for each combination of n, m and η, leading to 5*3*3*3=135 test instances in total. The due date is chosen as $\delta=\overline{\mu}+\sqrt{m}\overline{\sigma}$, where $\overline{\mu}=\sum_{j=1}^n\mu_j/m$ and $\overline{\sigma}=\sqrt{\sum_{j=1}^n\sigma_j^2/m}$.

The reasoning behind our choices for the experimental settings is the following: first and foremost, we test different values of n and m. The choice of the variance was made so as to obtain non-negative processing times in virtually all cases. Other values for the due date have been tested, but for all of those the optimal objective frequently went to one or zero and (almost) none of the dominance rules worked properly. Finally, the value of the expected duration is somewhat specific, but from our experiments, we find that other choices for the expectation do not significantly influence the interpretation of the results.

6.2. Overall findings

In this section, the total run time of an algorithm is referred to as T_{Total} while the time spent before finding an optimal solution is referred to as T_{Best}. All times are expressed in seconds (s). The total number of nodes explored in the search tree is referred to as n_{TNodes} while the number of nodes explored until an optimal solution is found, is referred to as n_{BNodes}. Tables 2 and 3 show the results obtained for B&B1 and B&B2, respectively. In detail, each cell in Tables 2 and 3 is (unless mentioned otherwise) the average of nine values (three replications for each of three values of η). We naturally observe an increase in T_{Total}, T_{Best}, n_{TNodes} and n_{BNodes} in both B&B1 and B&B2 when the number of jobs is increased.

A similar observation is made when the number of machines increases, with two exceptions, namely the case of $n=14$, $m=5$ and $n=16$, $m=5$ in B&B1. Although these exceptions seem to have a random character, sometimes they occur when n is divisible by m. The empty cells in Table 3 show that B&B2 could not be run to completion or that optimal solutions could not be found within a reasonable time (3 h). Based on Tables 2 and 3, we conclude that B&B1 is more efficient than B&B2.

Table 2
Summary of the results for B&B1.

n	m	T_{Total}	T_{Best}	n_{TNodes}	n_{BNodes}
12	3	0.03	0.01	5498	2101
	4	0.04	0.02	6506	2597
	5	0.05	0.03	7494	2618
14	3	0.30	0.10	76,408	24,971
	4	0.36	0.13	119,969	44,201
	5	0.19	0.02	52,503	4681
16	3	1.73	0.71	558,521	230,007
	4	4.56	0.89	1,899,443	372,308
	5	4.31	1.41	1,839,663	605,891
18	3	23.38	5.04	7,766,514	1,674,670
	4	91.13	33.02	35,729,885	13,000,505
	5	146.66	58.55	61,227,780	24,542,967
20	3	243.6	37.84	78,124,171	12,124,558
	4	1918.00	333.55	686,099,627	119,415,891
	5	3652.03	560.01	1,400,604,653	220,726,079

Table 3
Summary of the results for B&B2.

n	m	T_{Total}	T_{Best}	n_{TNodes}	n_{BNodes}
12	3	0.05	0.01	26,543	1780
	4	0.31	0.02	134,420	6022
	5	1.38	0.08	504,717	17,788
14	3	0.40	0.01	256,507	1720
	4	5.34	0.13	2,605,489	53643
	5	24.94	0.64	10,032,771	243,993
16	3	2.47	0.07	1,442,141	39,834
	4	41.29	1.15	18,483,122	503,668
	5	327.40	5.17	121,799,224	1,878,149
18	3	22.35	0.38	13,790,248	227,487
	4	745.17	24.52	342,970,665	11,125,236
	5	7800.81	163.11	2,694,756,455	59,041,267
20	3	198.10	1.20	117,590,873	704,354
	4	–	135.44	–	68,666,059
	5	–	–	–	–

7. Discussion of the results

7.1. Detailed comparison

In order to compare the performance of the two exact algorithms, we take the average T_{Total} as the comparison criterion. For a fair comparison, we exclude all test instances with $n=20$ because B&B2 is unable to obtain the corresponding results. Table 4 shows the average T_{Total} for B&B1 and B&B2 for different numbers of jobs and machines and different values of η. In Table 4, T_{Total} for B&B2 is 2871% greater than for B&B1, averaged over all values of n. B&B2 is slightly faster than B&B1 for $m=3$ machines, but for large numbers of machines B&B1 is more efficient. B&B2 has average T_{Total} around 2005% larger than B&B1, across the different values for m. Similarly, for all values of η, B&B2 has average T_{Total} around 3202% larger than B&B1. Overall, the average T_{Total} for B&B2 is almost 33 times larger than the average T_{Total} for B&B1. This ratio will increase when larger values are considered for the number of jobs.

We run each algorithm for 1, 5, 10, 30, 60 and 120 s in order to evaluate B&B1 and B&B2 with limited CPU times; the results are shown in Table 5. Each cell in Table 5 contains two numbers: the average percentage deviation from the optimal objective value and the number of optimal solutions found out of 135 test instances. For time limits under 10 s, the deviation from optimal for B&B2 is less than for B&B1 while the contrary holds for higher time limits. Furthermore, in terms of the number of optimal solutions found, B&B2 outperforms B&B1 for limits less than or equal to 30 s while for higher thresholds, B&B1 is better than B&B2.

In order to evaluate and compare the ability of B&B1 and B&B2 to solve large test instances, we report the average deviation from the optimal objective value for the best solutions found within

Table 5
Average percentage deviation from the optimal objective value and number of optimal solutions found (out of 135; between parentheses) within the CPU time limit.

Time limit (s)	1	5	10	30	60	120
B&B1	4.64(68)	3.24(88)	1.16(94)	0.52(106)	0.37(116)	0.16(122)
B&B2	4.08(87)	2.35(101)	2.26(106)	0.90(111)	0.88(115)	0.81(119)

Table 4
Average T_{Total} for different numbers of jobs and machines and different values of η.

	n				m			η		
	12	14	16	18	3	4	5	0.25	0.5	0.75
B&B1	0.04	0.28	3.53	87.06	6.36	24.02	37.80	18.69	23.12	26.29
B&B2	0.58	10.22	123.72	2856.11	6.32	198.03	2038.63	648.00	771.17	816.32

10 s. The results are shown in Table 6 for different numbers of jobs and machines. From the table, we conclude that B&B2 is generally better than B&B1 in generating heuristic solutions for large test instances with limited running times when m is low; for high values of m, on the other hand, B&B1 is preferable.

Table 6
Average percentage deviation from the optimal objective value within 10 s run time.

		n				
		12	14	16	18	20
B&B1						
m	3	0.00	0.00	0.00	0.00	0.44
	4	0.00	0.00	0.00	0.39	3.90
	5	0.00	0.00	0.00	6.09	6.64
B&B2						
m	3	0.00	0.00	0.00	0.00	0.32
	4	0.00	0.00	0.00	0.26	0.55
	5	0.00	0.00	0.50	10.19	22.12

7.2. Performance of the initial solution

The performance of the initial solution is evaluated in Table 7. Each cell of this table contains the average optimality gap of the solutions produced by the initial solution procedure. Averaged over all test instances, the deviation from optimal is almost 25.3%. This deviation will tend to increase with increasing number of jobs and machines, although Table 7 contains some exceptions to this general observation.

Table 7
Average optimality gap for the initial solution.

	n				
	12	14	16	18	20
m					
3	5.96	28.75	21.04	4.38	18.62
4	11.04	46.49	10.73	36.33	5.45
5	56.45	45.07	32.76	47.10	8.12

7.3. Impact of the dominance rules

Table 8 reports the average percentage deviation from optimal for different variants of B&B1: in the table, setting "B&B1" refers to B&B1 with all of its dominance rules, while "B&B1-(i)" pertains to B&B1 without dominance rule i ($i=1,2,3,4$). By removing each of the proposed dominance rules in turn, we can evaluate the impact of the rules on the algorithm's performance. Each variant is run with a 10-s time limit. Table 8 indicates that dominance rule 2 is the most effective, followed by rules 1, 3 and 4, in that order.

Similarly, Table 9 evaluates the dominance rules of B&B2. We see that dominance rule 1 has a rather minor impact while dominance rule 2 seems to be very effective.

Table 8
Impact of the dominance rules for B&B1.

B&B1	B&B1-(1)	B&B1-(2)	B&B1-(3)	B&B1-(4)
1.16	2.52	2.93	1.94	1.45

Table 9
Impact of the dominance rules for B&B2.

B&B2	B&B2-(1)	B&B2-(2)
2.26	2.37	14.93

8. Conclusions and outlook on future work

In this paper, the robust parallel machine scheduling problem RPMSP is studied, which aims to maximize customer service level in a parallel machine environment. A solution representation is proposed that prevents repetitive enumeration of equivalent solutions in a search tree. Furthermore, two branch-and-bound algorithms (B&B1 and B&B2) are developed for finding optimal solutions. These two algorithms differ in their branching schemes: B&B1 loads machines sequentially while B&B2 does this simultaneously. We also describe an upper-bounding procedure and a heuristic algorithm that produces an initial lower bound. In order to accelerate the B&B algorithms, four dominance rules for B&B1 and two dominance rules for B&B2 are incorporated. Experimental results demonstrate that B&B1 is more efficient overall but that for short run times (less than 10 s), B&B2 regularly achieves a lower average deviation from the optimal objective value.

To the best of our knowledge, this paper is the first to study robust scheduling of parallel machines with stochastic job durations, and as such is a step in the direction of the development of robust plans for practical large-scale scheduling environments.

Nevertheless, the restriction to a parallel machine environment is still an important limitation of this work. As a future research opportunity, we distinguish robust scheduling in more complicated scheduling environments (e.g. flow shop, job shop, open shop); this would constitute an important step also towards the possibility of practical implementation of the models. The development of other exact, heuristic or meta-heuristic procedures for the RPMSP may also be interesting research topics. Another limitation of this paper is the fact that all processing times are assumed to be normally distributed, and so a direct relevant generalization of our work is the consideration of other probability distributions and of distribution-independent procedures; this will probably require approximation methods for computing customer service level. Also, the optimality gap of our initial heuristic is still rather large for a number of the instances tested; future work should focus on constructing a better starting heuristic, which may lead to a significant speed-up of the search procedures. Finally, the question whether a performance guarantee can be established for the initial heuristic also remains open.

Acknowledgements

This work was supported in part by: Ferdowsi University of Mashhad's Research Council, under grant #17579.2 (dated 26-04-2011).

Appendix. improvement of v^{min}

For some RPMSP instances, such as for the example instance used in this paper, the value obtained for v^{min} is rather low and does not achieve a significant speed-up of the algorithms. In this appendix, we describe an improvement procedure for v^{min}. In view of the extensive running times, however, we have not included this procedure in the implementations that were tested

in Sections 6 and 7. The intuition of the procedure is illustrated on the example instance. If we assume $v^{min}=1$, nine jobs should be assigned to three machines M_2, M_3 and M_4. If we suppose these nine jobs are equal (later on, we remove this assumption), there are seven ways for assigning these nine jobs to three machines such that each machine has at least v^{min} jobs, as follows: {(1,1,7),(1,2,6)(1,3,5)(1,4,4)(2,2,5)(2,3,4)(3,3,3)}, where each triple shows $(|\mathbf{a}_2|,|\mathbf{a}_3|,|\mathbf{a}_4|)$. In order to compute an easy upper bound for each case, we consider the condition of Proposition 1. Consider the case (3,3,3), for instance, implying $|\mathbf{a}_1|=1$, $|\mathbf{a}_2|=3$, $|\mathbf{a}_3|=3$ and $|\mathbf{a}_4|=3$. The upper bound in this case is $\phi((59-19)/\sqrt{4})(\phi((59-(19+19+20))/\sqrt{4+9+13}))^3=0.192$. Next, we select the case having the maximum upper bound and compare it to the current lower bound. If the upper bound is less than the lower bound, we conclude that v^{min} should be increased. For the example, the case (3,3,3) has the maximum upper bound among the seven cases. In B&B1, at the beginning of the procedure we have $LB=0.15$, which does not allow us to increase v^{min}. Consider now the node corresponding to the following solution: $\mathbf{a}_1=\{J_1,J_2\}$, $\mathbf{a}_2=\{J_3,J_6\}$, $\mathbf{a}_3=\{J_4,J_5,J_7\}$ and $\mathbf{a}_4=\{J_8,J_9,J_{10}\}$. The customer service level of this solution is 0.216 and since $0.192<0.216$, we update v^{min} to the value 2 for the rest of search.

References

[1] Biskup D, Herrmann J, Gupta JND. Scheduling identical parallel machines to minimize total tardiness. International Journal of Production Economics 2008;115:134–42.

[2] Shim SO, Kim YD. A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. Computers & Operations Research 2008;35:863–75.

[3] Mellouli R, Sadfi C, Chu C, Kacem I. Identical parallel machine scheduling under availability constraints to minimize the sum of completion times. European Journal of Operational Research 2009;197:1150–65.

[4] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. Annals of Discrete Mathematics 1979;5:287–326.

[5] Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys D. Sequencing and scheduling: algorithms and complexity. In: Graves SS, Rinnooy Kan AHG, Zipkin P, editors. Handbooks in Operations Research and Management Science, vol. 4; 1993. p. 445–522.

[6] Dell'Amico M, Iori M, Martello S, Monaci M. Heuristic and exact algorithms for the identical parallel machine scheduling problem. INFORMS Journal on Computing 2008;20:333–44.

[7] Shim SO, Kim YD. Scheduling on parallel identical machines to minimize total tardiness. European Journal of Operational Research 2007;177:135–46.

[8] Dunstalland S, Wirth A. A comparison of branch-and-bound algorithms for a family scheduling problem with identical parallel machines. European Journal of Operational Research 2005;167:283–96.

[9] Dell'Amico M, Martello S. Optimal scheduling of tasks on identical parallel processors. ORSA Journal on Computing 1995;7:191–200.

[10] Mokotoff E. An exact algorithm for the identical parallel machine scheduling problem. European Journal of Operational Research 2004;152:758–69.

[11] Dell'Amico M, Martello S. A note on exact algorithms for the identical parallel machine scheduling problem. European Journal of Operational Research 2005;160:576–8.

[12] Haouari M, Jemmali M. Tight bounds for the identical parallel machine-scheduling problem: Part II. International Transactions in Operational Research 2008;15:19–34.

[13] Van Den Akker JM, Hoogeveen JA, Van De Velde SL. Parallel machine scheduling by column generation. Operations Research 1999;47:862–72.

[14] Cheng ZL, Powell WBA. Column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. European Journal of Operational Research 1999;116:220–32.

[15] Pinedo M. Scheduling. Theory, Algorithms, and Systems. 3rd ed.Prentice Hall; 2008.

[16] Herroelen W, Leus R. Project scheduling under uncertainty: survey and research potentials. European Journal of Operational Research 2005;165(2): 289–306.

[17] Anglani A, Grieco A, Guerriero E, Musmanno R. Robust scheduling of parallel machines with sequence-dependent set-up costs. European Journal of Operational Research 2005;161:704–20.

[18] Daniels RL, Carrillo JE. β-Robust scheduling for single-machine systems with uncertain processing times. IIE Transactions 1997;29:977–85.

[19] Wu CW, Brown KN, Beck CJ. Scheduling with uncertain durations: modeling β-robust scheduling with constraints. Computers & Operations Research 2009;36:2348–56.

[20] Sherali HD, Smith JC. Improving discrete model representations via symmetry considerations. Management Science 2001;47:1396–407.