Job Shop Scheduling with Probabilistic Durations[1]

J. Christopher Beck and **Nic Wilson**[2]

Abstract. Proactive approaches to scheduling take into account information about the execution time uncertainty in forming a schedule. In this paper, we investigate proactive approaches for the job shop scheduling problem where activity durations are random variables. The main contributions are (i) the introduction of the problem of finding probabilistic execution guarantees for difficult scheduling problems; (ii) a method for generating a lower bound on the minimal makespan; (iii) the development of the Monte Carlo approach for evaluating solutions; and (iv) the design and empirical analysis of three solution techniques: an approximately complete technique, found to be computationally feasible only for very small problems, and two techniques based on finding good solutions to a deterministic scheduling problem, which scale to much larger problems.

1 INTRODUCTION

Proactive scheduling techniques seek to produce an off-line schedule that incorporates a model of uncertainty, with the goal of building robust schedules. In this paper, we address the problem of job shop scheduling when the durations of the activities are random variables. Our main objective is to find a solution which has a probability of execution of at least $1 - \alpha$ (with e.g., $\alpha = 0.05$) and which has a good (ideally, minimal) probabilistic makespan. This is a challenging problem as simply *evaluating* a solution appears to be hard. To search for such a solution, we define a Monte Carlo branch-and-bound algorithm. At each node, the search is pruned if we can be almost certain (based on the Monte Carlo simulation) that the partial solution cannot be extended to a solution better than our current best solution. Our empirical results indicate that such a technique is only practical for very small problems. The other methods we introduce make use of the highly developed solution methods for the deterministic problem to generate promising solutions based on deterministic assumptions. These solutions are then evaluated using Monte Carlo simulation. We investigate both constructive search and tabu search generation techniques and our experimental results indicate that these perform almost as well as the branch-and-bound technique for small problems and greatly out-perform it on larger problem instances.

A secondary objective is to find a lower bound for the minimal probabilistic makespan. We show that such a lower bound can be generated from a lower bound for a particular deterministic problem.

In the next section we define probabilistic job shop scheduling problems and discuss related work. In Section 3, we present a lower bound on the minimal makespan. Section 4 describes the use of Monte Carlo simulation for evaluating solutions. In the following section, the Monte Carlo branch-and-bound and two heuristic algorithms based on transforming a probabilistic problem into a standard problem are introduced. Section 6 presents our empirical studies.

[1] This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075 and ILOG, SA.

[2] Cork Constraint Computation Centre, Department of Computer Science, University College Cork Cork, Ireland. {c.beck,n.wilson}@4c.ucc.ie

2 BACKGROUND

The job shop scheduling problem with probabilistic durations is a natural extension of the standard (deterministic) job shop scheduling problem (JSP). A JSP involves a set $\mathcal{A}$ of activities, where each $A_i \in \mathcal{A}$ has a positive duration d_i. $\mathcal{A}$ is partitioned into *jobs*, and with each job is associated a total ordering on that set of activities. No activities that require the same resource can overlap in their execution and once an activity is started it must be executed for its entire duration. We represent this formally by another partition of $\mathcal{A}$: into *resource sets*. A *solution* consists of a total ordering on each resource set, which does not conflict with the jobs ordering, i.e., the union of the resource orderings and job orderings is an acyclic relation on $\mathcal{A}$. Thus if A_i and A_j are in the same resource set, a solution either orders A_i before A_j, or A_i after A_j. A *partial solution* consists of a partial ordering on each resource set which can be extended to a solution.

Let s be a solution. A *path in* s (or an s*-path*) is a sequence of activities such that if A_i immediately precedes A_j in the sequence, either (i) A_i and A_j are in the same job, and A_i immediately precedes A_j in that job, or (ii) A_i and A_j are in the same resource set and s orders A_i before A_j. The length $\mathrm{len}(\pi)$ of a path π (of a solution), is equal to the sum of the durations of the activities in the path, i.e., $\sum_{A_i \in \pi} d_i$. The *makespan* $\mathrm{make}(s)$ of a solution s is defined to be the length of the longest s-path. An s-path π is said to be a *critical* s*-path* if the length of π is equal to the makespan of the solution. The *minimum makespan* of a job shop scheduling problem is defined to be the minimum makespan $\mathrm{make}(s)$ over all solutions s.

An *independent probabilistic job shop scheduling problem* is the same as the JSP, except that the duration $\mathbf{d}_i$ associated with an activity $A_i \in \mathcal{A}$ is a random variable; $\mathbf{d}_i$ has distribution P_i, expected value $\mu_i = \mathrm{E}[\mathbf{d}_i]$ and variance $\sigma_i^2 = \mathrm{Var}[\mathbf{d}_i]$. These random variables are fully independent. The length of a path π of a solution s is now a random variable, which we write as $\mathbf{len}(\pi)$. The makespan $\mathbf{make}(s)$ of solution s (the length of the longest path in s) is therefore also a random variable.

In the *probabilistic job shop scheduling problem* we have a joint probability measure P over the durations vectors. (The intention is that this will be such that we can efficiently sample with the joint density function.) Here, for activity A_i, distribution P_i is defined to be the appropriate marginal, with expected value μ_i and variance σ_i^2.

We fix a value $\alpha \in (0, 0.5]$; our aim is to find as small a value of D as possible such that there exists a solution s with $\Pr(\mathbf{make}(s) \leq D) \geq 1 - \alpha$, i.e., $\Pr(\mathbf{make}(s) > D) \leq \alpha$. Call $D_\alpha(s) = \inf\{D : \Pr(\mathbf{make}(s) \leq D) \geq 1 - \alpha\}$ the α-makespan of s, and let D_α (*the* α*-minimum makespan*) be the minimum value of $D_\alpha(s)$ over all solutions s. It can be shown that there exists a

solution s with $\Pr(\mathbf{make}(s) \leq D_\alpha) \geq 1-\alpha$ (in the case of a continuous distribution we have equality: $\Pr(\mathbf{make}(s) \leq D_\alpha) = 1-\alpha$). Furthermore, for any $D' < D_\alpha$, and all solutions s, there is more than α chance that the random makespan is more than D': $\Pr(\mathbf{make}(s) > D') > \alpha$.

Evaluating a solution in the deterministic case, i.e., finding the associated makespan, can be achieved in low degree polynomial time using a longest path algorithm. The disjunctions of resource constraints turn this very easy problem into the NP-complete JSP [6]. PERT networks, on the other hand, generalize this simple longest-path problem by allowing durations to be independent random variables, leading to a #P-complete problem [7]. The probabilistic JSP makes both these generalizations. Consequently, finding the optimal solutions of a probabilistic JSP looks to be very hard, and so we focus on methods for finding good solutions.

As well as PERT networks, probabilistic JSPs have some similarity with the problem studied in Daniels & Carrillo [3] which considers a one-machine scheduling problem with probabilistic durations. This is shown to be NP-hard even though the underlying deterministic problem can be solved in polynomial time. In [4], informal proactive techniques for difficult problems are presented but to our knowledge no one has sought probabilistic guarantees for difficult underlying scheduling problems.

3 A LOWER BOUND FOR α-MINIMUM MAKESPAN

In this section we show that a lower bound for the α-minimum makespan D_α can be found by solving a particular deterministic JSP.

The duration of activity A_i in the deterministic problem is defined to be $\mu_i + q\sigma_i$, where q is a fixed non-negative value. Let s be a solution, and let π be an s-path. The deterministic length $\mathrm{len}_q(\pi)$ of π is equal to $\sum_{A_i \in \pi}(\mu_i + q\sigma_i) = \sum_{A_i \in \pi}\mu_i + q\sum_{A_i \in \pi}\sigma_i$. Let make_q be the minimum deterministic makespan over all solutions. We say that q is α-sufficient if for any solution s there exists a (deterministic) critical s-path π with $\Pr(\mathbf{len}(\pi) > \mathrm{len}_q(\pi)) > \alpha$, i.e., there is more than α chance that the random path length is greater than the deterministic length.

Proposition 1 *For a probabilistic JSP, suppose q is α-sufficient, and let $D' \leq \mathrm{make}_q$ be a lower bound for the deterministic minimum makespan. Then for any solution s, $\Pr(\mathbf{make}(s) > D') > \alpha$. Therefore D' is a lower bound for the α-minimum makespan D_α.*

Finding α-sufficient q-values

In the rest of this section, we assume an independent probabilistic JSP. Let μ_π be $\mathrm{E}[\mathbf{len}(\pi)]$, the expected value of the length of π, which is equal to $\sum_{A_i \in \pi}\mu_i$. Let $\sigma_\pi^2 = \mathrm{Var}[\mathbf{len}(\pi)]$, the variance of the length of π, which is equal to $\sum_{A_i \in \pi}\sigma_i^2$, since we are assuming that the durations are independent. For $B \geq 0$, write $\theta_B(\pi)$ for $\mu_\pi + B\sigma_\pi = \sum_{A_i \in \pi}\mu_i + B\sqrt{\sum_{A_i \in \pi}\sigma_i^2}$. We say that B is α-sufficient if for any critical path π of any solution, $\Pr(\mathbf{len}(\pi) > \theta_B(\pi)) > \alpha$, i.e., there is more than α chance that π is more than B standard deviations longer than its expected length.

If each duration is normally distributed, then $\mathbf{len}(\pi)$ will be normally distributed, since it is the sum of independent distributions. Even if the durations are not normally distributed, $\mathbf{len}(\pi)$ will often be close to being normally distributed, by the central limit theorem. So $\Pr(\mathbf{len}(\pi) > \theta_B(\pi))$ will then be approximately $1 - \Phi(B)$, where Φ is the unit normal distribution. A B value of slightly less than $\Phi^{-1}(1-\alpha)$ will be α-sufficient, given approximate normality.

We say that q is B-sufficient if for any critical path π of any solution, $\mathrm{len}_q(\pi) \leq \theta_B(\pi)$. Clearly: *a sufficient condition for q to be α-sufficient, is if q is B-sufficient for some B which is α-sufficient.* A q-value q is B-sufficient if and only if for any critical path π of any solution $\sum_{A_i \in \pi}\mu_i + q\sum_{A_i \in \pi}\sigma_i \leq \sum_{A_i \in \pi}\mu_i + B\sqrt{\sum_{A_i \in \pi}\sigma_i^2}$, that is, $q \leq \frac{B}{\sqrt{L_\pi}}\frac{\sqrt{\text{Average}_{A_i \in \pi}\sigma_i^2}}{\text{Average}_{A_i \in \pi}\sigma_i}$, where L_π is the number of activities in path π.

A crude sufficient condition for this is $q \leq \frac{B}{\sqrt{L}}$ where L is an upper bound for the number of activities in any critical path π for any solution. In particular, we could choose $q = \frac{B}{\sqrt{L}}$. So, given that the lengths of the paths are approximately normally distributed, a q-value of a little less than $\frac{\Phi^{-1}(1-\alpha)}{\sqrt{L}}$ will be α-sufficient.

The lower bound we have derived can be seen to be quite conservative; we approximate the random variable $\mathbf{make}(s)$ by the random length of particular path and $\mathrm{len}_q(\pi)$ will tend to be a crude lower bound for $\theta_B(\pi)$. The strength of our lower bound method, however, is that it is computationally feasible for reasonably large problems as it uses existing well-developed JSP methods.

4 EVALUATING A SOLUTION USING MONTE CARLO SIMULATION

For a given D we want to assess if there exists a solution for which there is a chance at most α that the random makespan is greater than D. Our methods will all involve generating (partial) solutions, and testing this condition.

Evaluating a solution amounts to solving a PERT problem with uncertain durations, a #P-complete problem. As in other #P-complete problems such as the computation of Dempster-Shafer Belief [14], a natural approach to take is Monte Carlo simulation [2]; we do not try to make an exact computation but instead choose an accuracy level δ and require that with a high chance our random estimate is within δ of the true value. The algorithm then has optimal complexity (low-degree polynomial) but with a potentially high constant factor corresponding to the number of trials required for the given accuracy.

To evaluate a (partial) solution s using a Monte Carlo simulation: we perform a (large) number N of independent trials assigning values to each random variable. This generates a deterministic problem, and we can check very efficiently if the corresponding makespan is greater than D; if so, we say the the trial succeeds. The proportion of trials that succeed is then an estimate of $\Pr(\mathbf{make}(s) > D)$. For the case of independent probabilistic JSPs we can pick the random durations vector by picking, using distribution P_i, a value for $\mathbf{d}_i$ for each activity A_i. For the general case, picking random durations vector will still be efficient in many situations; for example, if the distribution is represented by a Bayesian network.

Let T be the proportion of trials that succeed, which is an estimate of $p = \Pr(\mathbf{make}(s) > D)$, the chance that a randomly generated durations vector leads to a makespan (for solution s) greater than D. The expected value of T is equal to p. The standard deviation of T is $\sqrt{\frac{p(1-p)}{N}}$. The random variable NT is binomially distributed, and so (because of the deMoivre-Laplace limit theorem) we can use a normal distribution to approximate T.

When is the solution good enough?

A solution is good enough when we can say with confidence that, for our fixed value α, $\Pr(\mathbf{make}(s) > D) \leq \alpha$. Based on the above, we therefore need the observed T to be at least a little smaller than α.

We shall use a confidence interval-style approach. Let $K \geq 0$. Recall that p is an unknown quantity that we want to find information about. We say that "*$p \geq \alpha$ is K-implausible given the result T*" if the following condition holds: *$p \geq \alpha$ implies that T is at least K standard deviations below the expected value*, i.e., $T \leq p - \frac{K}{\sqrt{N}}\sqrt{p(1-p)}$.

If it were the case that $p \geq \alpha$, and "$p \geq \alpha$ is K-implausible given T", then an unlikely event would have happened. For example, with $K = 2$, (given the normal approximation), such an event will only happen about once every 45 experiments; if $K = 4$ such an event will only happen about once every 32,000 experiments.

If $\Pr(\mathbf{make}(s) > D) \geq \alpha$ is K-implausible given the result T then we can be confident that $\Pr(\mathbf{make}(s) > D) < \alpha$, so that D is an upper bound of $D_\alpha(s)$ and hence of the α-minimum makespan D_α. The confidence level, based on a normal approximation of the binomial, is $\Phi(K)$, where Φ is the unit normal distribution.

Similarly, for any α between 0 and 0.5, we say that $p \leq \alpha$ is K-implausible given the result T if the following condition holds: *$p \leq \alpha$ implies that T is at least K standard deviations above the expected value, i.e.,* $T \geq p + \frac{K}{\sqrt{N}}\sqrt{p(1-p)}$.

K-implausibility cannot be tested directly using the definition since p is unknown. Fortunately we have the following result.

Proposition 2 *With the above definitions:*

(i) *$p \geq \alpha$ is K-implausible given T iff* $T \leq \alpha - \frac{K}{\sqrt{N}}\sqrt{\alpha(1-\alpha)}$.

(ii) *$p \leq \alpha$ is K-implausible given T iff* $T \geq \alpha + \frac{K}{\sqrt{N}}\sqrt{\alpha(1-\alpha)}$.

Part (i) of this result shows us how to evaluate a solution s with respect to a bound D: if we generate T using a Monte Carlo simulation which is at least $\frac{K}{\sqrt{N}}\sqrt{\alpha(1-\alpha)}$ less than α then we can have confidence that D is an upper bound for $D_\alpha(s)$.

Finding D from a given solution

The Monte Carlo simulation can be adapted to generate a just-satisfactory D from a solution s. We simulate the values of $\mathbf{make}(s)$ and record the distribution. We decide on a value of K, corresponding to the desired confidence, and choose D minimal such that the associated T value satisfies $T \leq \alpha - \frac{K}{\sqrt{N}}\sqrt{\alpha(1-\alpha)}$. Then by Proposition 2(i), $\Pr(\mathbf{make}(s) > D) \geq \alpha$ is K-implausible given T. We can be confident that $\Pr(\mathbf{make}(s) > D) \leq \alpha$, i.e., that $D_\alpha(s) \leq D$.

5 SEARCHING FOR SOLUTIONS

Recall that our aims in solving the probabilistic JSP are to find an α-minimum makespan and to find a lower bound on the α-minimum makespan. Ideally, we would like these two values to be identical. In practice, due to limits on computational resources this may not be possible and so we aim to make these two values as close as possible. As shown in Section 3, a crude lower bound can be found by solving a deterministic JSP with the appropriate choice of activity durations. In this section, we present how the Monte Carlo simulation results presented in the previous section are used in search.

An Approximately Complete Branch-and-Bound Algorithm

Given the ability to establish a new D value from a complete sequence of activities in each resource set, and the ability to test if partial sequences will not lead to lower D values, the obvious choice for search is a branch-and-bound (B&B). If we are able to cover the entire search space, such an approach is approximately complete with confidence related to our choice of K.

The B&B tree is a (rooted) binary tree. Associated with each node e in the tree is a partial solution s_e, which is a solution if the node is a leaf node. The empty partial solution is associated with the root node. Also associated with each non-leaf node e is an activity, A_i, which has not been ranked in its resource set in partial solution s_e. The two nodes below e extend s_e: one ranks A_i as the next activity in its resource set, the other postpones A_i to be ranked later [9]. This branching scheme searches for non-delay schedules [5].

A value of global variable D^* is always such that we have confidence (corresponding to the choice of K) that there exists a solution s with $D_\alpha(s)$ at most D^*. Whenever we reach a leaf node e we find the D associated with a solution s_e by Monte Carlo simulation. We set $D^* := \min(D^*, D)$. Variable D^* is initialized to some high value.

At non-leaf nodes e we make a check to see if it is worth exploring the subtree below e. We perform a Monte Carlo simulation for partial solution s_e using the current value of D^*; this generates a result T. We use proposition 2(ii) to determine if $\Pr(\mathbf{make}(s_e) > D^*) \leq \alpha$ is K-implausible given T; if it is then we backtrack, since we can be confident that there exists no solution extending the partial solution s_e that improves our current best solution. If K is chosen sufficiently large then we can be confident that we will not miss a good solution.[3]

We refer to this algorithm as the *MCBB* algorithm.

Heuristic Algorithms

The main computational weakness of the MCBB algorithm is that the pruning condition depends on simulation at each node. The pruning techniques developed in constraint-based scheduling are ineffective as they assume deterministic durations and therefore the implicit lower bounds calculated using, for example, the mean durations, are too weak. To reduce this dependency, we present two heuristic algorithms that make use of deterministic scheduling problems to produce candidate solutions that are then evaluated using simulation.

A simple constructive search algorithm can be developed by first finding a good initial solution to a deterministic problem. Subsequently, deterministic solutions, s, as good or better than the initial solution are enumerated and each one is simulated to determine $D_\alpha(s)$. We use the same constructive algorithm for both phases. This algorithm searches only for non-delay schedules by using the same branching scheme as MCBB but also strong constraint propagation [12, 8]. For small problems (size 10×10 or smaller), an optimal deterministic solution is easily found and proved in the first phase. The second phase therefore enumerates the optimal, non-delay deterministic schedules. For larger problems, a time limit is placed on the first phase, and the simulated solutions are those which are as good or better than the best solution found in phase 1. We denote this technique as *NDf* as the solutions to be simulated are filtered to be good,

[3] Because we are doing a very large number of tests, we need much higher confidence than for a usual confidence interval; fortunately, the confidence associated with K is (based on the normal approximation of a binomial, and the approximation of a tail of a normal distribution) approximately $1 - \frac{1}{K\sqrt{2\pi}}e^{-\frac{1}{2}K^2}$, and so tends to 1 extremely fast as K increases.

non-delay deterministic solutions.

We can also use local search to generate a sequence of deterministic solutions that are then simulated. Using an implementation of Nowicki & Smutnicki's TSAB tabu search algorithm [11], whenever a solution is found with a deterministic makespan as good or better than the best makespan found so far, we simulate it. As with the NDf algorithm, the solution, s, with the lowest $D_\alpha(s)$ is retained. We denote this technique as *Tabuf*.

For these techniques, it is necessary to assign fixed durations to each activity. A standard approach is to use the mean duration. However, in such cases there is no representation of the uncertainty surrounding that duration, and this does not take into account that we want a high probability $(1 - \alpha)$ of execution. A more general approach is to heuristically use the formulation for the lower bound on α-minimum makespans presented in Section 3: the duration of activity A_i is defined to be $\mu_i + q\sigma_i$, where q is a fixed non-negative value. Since we are no longer limited to producing a lower bound, we have flexibility in selecting q. Intuitively, we want a q-value that leads to a situation where good deterministic solutions also have low values of $D_\alpha(s)$. We experiment with a number of q-values based on the analysis in Section 3.

6 EMPIRICAL INVESTIGATIONS

Our empirical investigations examine five sets of probabilistic JSPs of size $\{3 \times 3, 4 \times 4, 6 \times 6, 10 \times 10, 20 \times 20\}$ with three uncertainty levels $u_j \in \{0.1, 0.5, 1\}$. A deterministic problem is generated using an existing generator [13] with integer durations drawn uniformly from the interval [1, 99]. Three probabilistic instances at different levels of uncertainty are then produced by setting the mean durations μ_i to be the deterministic durations and by uniformly drawing the standard deviation σ_i of the duration of an activity, A_i, from the interval [0, $u_j\mu_i$]. The distribution of each duration is approximately normal. For each problem size, we generate 10 deterministic problems which are transformed into 30 probabilistic instances.

The smaller problem sizes were chosen to demonstrate the behavior at the apparent size limit of the MCBB algorithm: MCBB is unable to completely solve problems within 600 CPU seconds of size larger than 3×3. The 20×20 problems were chosen as deterministic problems of that size cannot be generally solved to optimality using current technology within the 600 CPU seconds. The hardware used for the experiments is a 1.8GHz Pentium 4 with 512 M of main memory running Linux RedHat 9.

Given the stochastic nature of the simulation, each algorithm is run 10 times on each problem instance with different random seeds. Each run has an overall time limit of 600 CPU seconds. Each Monte Carlo simulation uses $N = 1000$. We report the mean stochastic makespan that each algorithm found over each run on each instance.

We experimented with four q values displayed in Table 1. In all cases, we set $B = 1.645$ corresponding $\alpha = 0.05$. Value q_3 was found for each instance by Monte Carlo simulation: simulating 100000 paths of n activities.

q_0	q_1	q_2	q_3
0	$\frac{B}{\sqrt{2n}}$	$\frac{q_1+q_3}{2}$	$\frac{B}{\sqrt{n}} \frac{\sqrt{\text{Average}_{A_i \in \pi} \sigma_i^2}}{\text{Average}_{A_i \in \pi} \sigma_i}$

Table 1. The q-values used in the experiments.

A lower bound for each instance was found using q_1, a simple, very plausibly α-sufficient q-value (see Section 3). We solved each instance using constraint-based tree search incorporating the global constraint propagation used above and using a texture-based search heuristic [1]. A maximum time of 600 CPU seconds was given. All problems smaller than 20×20 were easily solved to optimality. None of the 20×20 problems were solved to optimality. Therefore, the lower bound for those problems is approximate.

An impression of the overall results can be gained by looking at the bold entries in Table 2. MCBB dominates on small problems, the two heuristic methods are strong for the medium size problems, and, for the largest problems, Tabuf is superior. In terms of the q-values, there is little difference on small problems but for 6×6 and larger there is a clear disadvantage for q_0. This demonstrates that incorporating uncertainty information into the deterministic durations for the heuristic algorithms has benefits.

For the 3×3 problems, MCBB terminates without timing out. Therefore, the $D_\alpha(s)$ solutions are approximately optimal. The quality of the lower bound formulation can therefore be assessed by looking at the optimality gap for these problems. This is the difference between the best mean makespan found and the mean lower bound expressed as a percentage of the mean lower bound. For low uncertainty, the gap for the 3×3 problems is only 2.3% but increases to over 10% for more uncertain problems. We expect the quality of the lower bound calculation to also worsen somewhat with increased problem size. Therefore, given that the gap for uncertainty level 0.1 across all problem sizes grows only slightly, we can expect that the solutions found for low uncertainty across all these problem sizes are reasonably close to optimal.

In comparing the two heuristic algorithms, when the NDf can enumerate all optimal non-delay solutions (i.e., up to 10×10) within the time limit, it is competitive, at least at some q-values with the Tabuf algorithm. On the 20×20 problems, Tabuf is able to find solutions with much lower (deterministic) makespans than NDf. We believe that the performance difference in terms of $D_\alpha(s)$ is due to better underlying deterministic makespans found by Tabuf.

Each cell in Table 2 is the mean value over 10 independent runs of 10 problems. The observed mean standard deviations for each cell are very small: the maximum over all cells is less than 2.2% of the corresponding mean value with the mean over all cells being less than 0.7% of the corresponding mean makespan.

7 FUTURE WORK

There are two directions for future work on the algorithms presented in this paper. First, MCBB could be improved to make more use of deterministic techniques and/or to incorporate probabilistic reasoning into existing deterministic techniques. For example, a number of deterministic lower bound formulations for PERT networks exist in the operations research literature [10] that may be used to evaluate partial solutions. Another approach to improving the MCBB performance is to incorporate explicit reasoning about probability distributions into standard constraint propagation techniques. Techniques like the longest path calculations and edge-finding make inferences based on the propagation of minimum and maximum values for temporal variables. We believe that many of these techniques can be adapted to reason about probabilistic intervals.

A second direction for future work is the improvement of the heuristic algorithms. The key advantage of these algorithms is that they make use of deterministic techniques for scheduling. By transforming probabilistic problems to deterministic problems, we can bring a significant set of existing tools to bear on the problem. Further

Problem Size	Unc. Level	Lower Bound (q_1)	MCBB	q_0		q_1		q_2		q_3		Gap (%)
				NDf	Tabuf	NDf	Tabuf	NDf	Tabuf	NDf	Tabuf	
3×3	0.1	258	**264***	**265**	266	**265**	267	**265**	267	**265**	267	2.3
	0.5	301	**312***	318	318	317	317	316	316	316	316	3.7
	1	375	**414***	433	432	425	426	425	424	425	423	10.7
4×4	0.1	322	**332**	**331**	**331**	**331**	**331**	**331**	**331**	**331**	**331**	2.8
	0.5	375	**396**	401	401	401	399	402	399	402	400	5.6
	1	433	**492**	515	510	504	501	505	502	507	501	13.6
6×6	0.1	502	546	**516**	**516**	**515**	**515**	**515**	**516**	**515**	**516**	2.6
	0.5	569	643	**619**	**620**	**622**	**619**	**620**	624	**620**	**620**	8.8
	1	642	794	783	781	**765**	**761**	**763**	**764**	**764**	**763**	18.5
10×10	0.1	827	961	873	**853**	863	**856**	860	**854**	862	**855**	3.1
	0.5	913	1119	1038	1027	**1025**	**1022**	**1024**	**1023**	**1020**	**1025**	11.7
	1	1026	1367	1291	1283	**1265**	**1270**	**1265**	**1264**	**1269**	1271	23.2
20×20	0.1	1672†	1942	1857	1732	1833	**1723**	1841	**1723**	1798	**1723**	3.1
	0.5	1789†	2235	2169	2053	2142	**2037**	2147	**2027**	2134	**2032**	13.3
	1	1943†	2759	2690	2584	2635	**2538**	2629	**2532**	2602	**2528**	30.1

Table 2. The lower bounds, mean makespans for each algorithm, and the percentage difference between the best solution found and the lower bound. '*' indicates a set of runs for which we have, with high confidence, found close-to-optimal makespans. '†' indicates approximate lower bounds. The mean makespans within 0.5% of the best mean makespans are shown in bold font.

study is needed to find additional ways to do this transformation. For example, we can envision a search that adaptively changes q-values during the search in order to find those that lead to solutions with better $D_\alpha(s)$ values. A deeper understanding of the relationship between good deterministic solutions and good probabilistic solutions is necessary to pursue this work in a principled fashion.

8 CONCLUSION

In this paper, we addressed job shop scheduling when the durations of the activities are random variables. The objectives are to find a solution which has a high probability of having a good makespan, and to find a lower bound for the minimal probabilistic makespan. We introduced three approaches to the problem of generating solutions: a branch-and-bound technique using Monte Carlo simulation to evaluate partial solutions, and two heuristic approaches that transform the probabilistic problem into a deterministic problem which is then used to generate potential solutions. The quality of these solutions is evaluated with Monte Carlo simulation. We also showed how a lower bound for the minimal probabilistic makespan can be generated from a lower bound for a particular deterministic problem.

Our empirical evaluation demonstrated that the branch-and-bound is only able to find approximately optimal solutions for very small problem instances. For larger instances, the heuristic techniques perform much more strongly while providing no optimality guarantees. Based on comparisons with the lower bound, our results indicate that for large problems with low uncertainty, good quality solutions are found. At higher levels of uncertainty, the gap between the lower bounds and the best solutions found is greater, which we suspect is due to both poorer quality solutions and a weaker lower bound.

Proactive scheduling techniques seek to incorporate models of uncertainty into an off-line, predictive schedule. The goal of such techniques is to increase the robustness of the schedules produced. This is important because a schedule is not typically generated or executed in isolation. Other decisions such as when to deliver raw materials and how to schedule up- and down-stream factories are all affected by an individual schedule. Differences between a predictive schedule and its execution can be a significant source of disruption leading to cascading delays across widely separated entities. The ability, therefore, to develop schedules that are robust to uncertainty is very important. This paper represents a step in that direction.

REFERENCES

[1] J. C. Beck and M. S. Fox, 'Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics', *Artificial Intelligence*, **117**(1), 31–81, (2000).

[2] J. M. Burt and M. B. Garman, 'Monte carlo techniques for stochastic network analysis', in *Proceedings of the fourth annual conference on applications of simulation*, pp. 146–153, (1970).

[3] R.L. Daniels and J.E. Carrillo, 'β-robust scheduling for single-machine systems with uncertain processing times', *IIE Transactions*, **29**, 977–985, (1997).

[4] A.J. Davenport, C. Gefflot, and J.C. Beck, 'Slack-based techniques for robust schedules', in *Proceedings of the Sixth European Conference on Planning (ECP-2001)*, (2001).

[5] H-L. Fang, *Genetic Algorithms in Timetabling and Scheduling*, Ph.D. dissertation, Department of Artificial Intelligence, University of Edinburgh, 1994.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[7] J. N. Hagstrom, 'Computational complexity of PERT problems', *Networks*, **18**, 139–147, (1988).

[8] P. Laborie, 'Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results', *Artificial Intelligence*, **143**, 151–188, (January 2003).

[9] C. Le Pape, P. Couronné, D. Vergamini, and V. Gosselin, 'Time-versus-capacity compromises in project scheduling', in *Proceedings of the Thirteenth Workshop of the UK Planning Special Interest Group*, (1994).

[10] A. Ludwig, R.H. Möhring, and F. Stork, 'A computational study on bounding the makespan distribution in stochastic project networks', *Annals of Operations Research*, **102**, 49–64, (2001).

[11] E. Nowicki and C. Smutnicki, 'A fast taboo search algorithm for the job shop problem', *Management Science*, **42**(6), 797–813, (1996).

[12] W. P. M. Nuijten, *Time and resource constrained scheduling: a constraint satisfaction approach*, Ph.D. dissertation, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.

[13] J.-P. Watson, L. Barbulescu, L.D. Whitley, and A.E. Howe, 'Constrasting structured and random permutation flow-shop scheduling problems: search-space topology and algorithm performance', *INFORMS Journal on Computing*, **14**(1), (2002).

[14] N. Wilson, *Algorithms for Dempster-Shafer Theory*, in: Kohlas, J., Moral, S., (eds.) Algorithms for Uncertainty and Defeasible Reasoning, Volume 5, Handbook of Defeasible Reasoning, Kluwer Academic Publishers, 2000.