International Journal of Production Research,
Vol. 43, No. 10, 15 May 2005, 2113–2134

Taylor & Francis
Taylor & Francis Group

Comparing deterministic, robust and online scheduling using entropy

H.-S. GAN and A. WIRTH†*

†Department of Mechanical and Manufacturing Engineering,
The University of Melbourne, VIC 3010, Australia

(*Received November 2004*)

We conjecture that when the uncertainty of scheduling information increases, one should change from deterministic, through robust to, finally, online scheduling techniques. Previously, extensive mathematical investigations have been carried out on the stability of a deterministic schedule for uncertain operation processing times. In this paper, we will use an empirical approach and an entropy measure to justify the transition between deterministic, robust and online scheduling. The use of an entropy measure in our context can be perceived, in a broader sense, as a pro-active approach to deal with changes in the level of information uncertainty and relative importance of each term in the total schedule execution cost. The level of information uncertainty may change due to the performance deterioration of processors (machines or human) and the replacement of old machines with new ones; and the changes in relative importance of cost elements may be due to changes in shop floor priorities and pressure from partners in the supply chain network. One can decide upon the scheduling strategies to be employed based on the latest entropy value of the information considered and the relative importance of each cost term.

Keywords: Deterministic; Robust; Online; Scheduling; Entropy

1. Introduction

Deterministic, robust and online scheduling techniques can be distinguished in terms of their objective of schedule generation. Deterministic scheduling techniques aim to generate optimal or near-optimal schedules, and in most cases the efficiency of the heuristic or algorithm used is traded off as the performance of the schedules generated moves closer to optimality. Robust scheduling techniques take a different approach since the scheduling information may change during the execution of the schedule. One can choose to produce robust initial off-line schedules but will be faced with the curse of computational intractability due to the large number of possible scenarios associated with the uncertain scheduling information (see work by Daniels and Kouvelis 1995, Kouvelis and Yu 1997, Kouvelis *et al.* 2000, Yang and Yu 2002 and Kuo and Lin 2002). Alternatively, one can select a 'simpler' heuristic to create a 'myopic' initial off-line schedule and repair the schedule when needed. Online scheduling techniques, on the other hand, commit operations over time and

*Corresponding author. Email: wirth@mame.mu.oz.au

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540412331333405

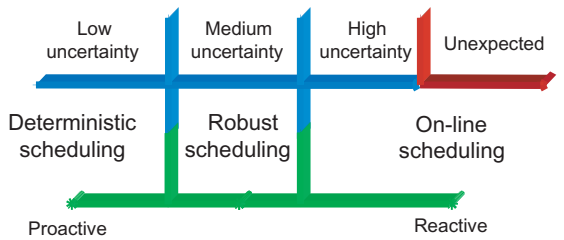

Figure 1. Scheduling information and scheduling techniques.

only aim to produce myopically good partial schedules, since scheduling information may only be revealed over time.

Deterministic, robust and on-line scheduling techniques, can also be perceived as pro-active, pro-active-reactive and reactive methods respectively. Pro-active methods aim to produce initial off-line schedules that are robust to uncertainties during run-time, reactive methods react to uncertainties during run-time. Figure 1 identifies these scheduling techniques with different types of scheduling information and classifies them in terms of proactive and reactive methods.

The type of scheduling technique employed may be dependent on the degree of uncertainty of the scheduling information provided in the planning phase. In general, scheduling information can be classified as follows:

1. Deterministic or almost certain information, where the scheduling information is known with a high level of certainty in the planning horizon. The scheduling information remains unaltered (or may vary by a negligible amount that does not alter the performance of the original schedule) upon deployment and execution of the schedule. Scheduled resource maintenance can be perceived as deterministic scheduling information. Scheduling with this type of information is termed deterministic scheduling.
2. Uncertain or incomplete information, where information is known in the planning phase, but is uncertain. The actual information will only be known at the execution phase. We distinguish between two levels of uncertainty as follows:

   - Moderately uncertain information, where there is uncertainty associated with the scheduling information and the scheduling information used in the planning phase may deviate upon deployment and execution of the schedule. Some of the disruptions that might occur are
     – machine breakdowns;
     – uncertain processing times;
     – uncertain due dates;
     – uncertain arrival dates;
     – uncertain costs;
     – removal of operations;
     – addition of operations.

     To handle the incompleteness of the scheduling information, pro-active and/or reactive scheduling methods must be employed, for example robust scheduling and dynamic or on-line scheduling.

- Highly uncertain information, where the scheduling information is known during the planning phase but is highly uncertain. For this type of information, dynamic or one-line scheduling is used.

3. Unexpected information, where the information is not known *a priori* in the planning phase. Scheduling information is only revealed over time. Disruptions outlined earlier, such as machine breakdowns, may be of this nature. Dynamic or on-line scheduling techniques are usually used.

We will attempt to draw 'boundaries' between deterministic, robust and online scheduling techniques by quantifying the degree of uncertainty of the scheduling information. Sotskov *et al.* (1997, 1998) and Lai *et al.* (1997) have performed extensive mathematical investigations on the stability (note that this is not the rescheduling stability) of a deterministic schedule given uncertain operation processing times. The first of these papers proposes finding a stability radius for processing times of operations in a job-shop and the last for a general shop scenario, such that processing times of operations varying within this radius would not affect the optimality of the current solution. Braun *et al.* (2002) have carried out stability analysis for Johnson's schedule in a two-machine flowshop (this is the schedule generated using Johnson's algorithm (Johnson 1954) for a two-machine flowshop) with limited machine availability, given uncertain operation processing times.

In this paper, we will use an empirical approach and an entropy measure to justify the transition from deterministic to robust scheduling approaches.

There have been limited attempts to justify deterministic, robust and online scheduling techniques in the literature. Most of these attempts have only considered simulations for a small range of uncertainty levels. Some of the papers addressing this issue are outlined below.

The superiority of reactive scheduling (robust or dynamic scheduling) to deterministic scheduling when information is uncertain, has been demonstrated through experiments conducted by Matsuura and Kanezashi (1996). They considered a job-shop problem minimizing makespan subjected to machine breakdowns and rush job arrivals. Scheduling techniques, namely fixed sequencing (deterministic scheduling), resequencing (rescheduling), dynamic dispatching (online scheduling) and switching (this is a technique where operations are scheduled using a dynamic dispatching rule after a certain number of disruptions) were compared in a pairwise manner, based on the proportions of cases where the makespan and the absolute change in makespan of the schedule was less than, equal to or greater than the default scheduling approach. Results show that resequencing is superior to fixed sequencing, and switching is superior to dynamic dispatching. Lawrence and Sewell (1997) have also considered the static and dynamic application of optimal, near-optimal and heuristic solutions to the job-shop problem of minimizing makespan subjected to uncertain processing times. Their results suggest that under dynamic situations algorithmically more sophisticated scheduling policies (which generate solutions that are closer to optimality in static conditions) perform comparatively worse than simple heuristics. Based on these experimental results, they have concluded that deterministic scheduling is not the preferred approach when information is incomplete.

We will begin our discussion by recalling in section 2 the concept of entropy and applying it to our scheduling problem in section 3. A schedule execution cost

function will be introduced in section 4. We will then describe our simulation structure and present our simulation results in the subsequent two sections. Section 7 contains some conclusions and possible future research directions.

2. The entropy concept

The entropy concept (Khinchin 1957) in probability theory was initially introduced in the area of information transmission. We will use this measure to quantify the amount of uncertainty associated with the scheduling information available to the schedule in the planning phase. First, we introduce this measure in a general context.

We say that we have a finite scheme if we are given a complete system of mutually exclusive events $A_1, A_2, \ldots, A_n$ with probabilities $p_1, p_2, \ldots, p_n$, respectively, where $\sum_{i=1}^{n} p_i = 1$ and $p_i \geq 0$ for all i. We define the entropy, a measure of the amount of uncertainty, associated with this finite scheme thus

$$H(p_1, p_2, \ldots, p_n) = -\sum_{k=1}^{n} p_k \log p_k \tag{1}$$

where the logarithms are taken to an arbitrary but fixed base, and we take $p_k \log p_k = 0$ if $p_k = 0$.

For any tow mutually independent finite schemes A and B with n and m events respectively, the probability π_{kl} of the simultaneous occurrence of the events A_k and B_l is $p_k q_l$ where p_k is the probability of event k occurring in scheme A, q_l is the probability of event l occurring in scheme B, and $1 \leq k \leq n$ and $1 \leq l \leq m$. Let the sets of events $A_k B_l$ represent another finite scheme designated by AB. It is easy to see that

$$H(AB) = H(A) + H(B) \tag{2}$$

where $H(AB)$, $H(A)$ and $H(B)$ are the corresponding entropies of the schemes AB, A and B. This expression can be easily extended for an arbitrary number of mutually independent finite schemes. For a scheme consisting of s mutually independent schemes $M = A_1 A_2, \ldots, A_s$, the entropy is given by

$$H(M) = H(A_1 A_2, \ldots, A_s) = \sum_{i=1}^{s} H(A_i) \tag{3}$$

Other properties of this entropy measure, such as those for dependent schemes, can be found, for example, in Khinchin (1957). For the purpose of this work, we will only consider mutually independent schemes.

3. Entropy of scheduling information

We will now reformulate the entropy measure (1) for the case where the operation processing times are uncertain.

Suppose that we have to schedule n operations in an arbitrary scheduling environment. Let $f_i(w_i)$ be the probability density function of the processing time of operation i, such that $\int_{a_i}^{b_i} f_i(w_i)\,\mathrm{d}w_i = 1$, where w_i is the processing time of operation

i and a_i and b_i are the processing time bounds for operation i. Let A_i^k be the event $[a_i + (k-1)\Delta_i] \leq w_i \leq [a_i + k\Delta_i]$ with probability

$$P_i^k = P(a_i + (k-1)\Delta_i \leq w_i \leq a_i + k\Delta_i) = \int_{a_i+(k-1)\Delta_i}^{a_i+k\Delta_i} f_i(w_i)\, \mathrm{d}w_i \tag{4}$$

for $1 \leq k \leq z_i$, $z_i = (b_i - a_i/\Delta_i)$ and the z_i s are integers. Hence, for each operation i, we have a finite scheme consisting of a set of events A_i^k for $1 \leq k \leq z_i$, designated by G_i. The entropy associated with the processing time information of operation i is given by

$$H(G_i) = -\sum_{k=1}^{z_i} P_i^k \log P_i^k \tag{5}$$

Assuming independence of the finite schemes for each operation, G_i, the entropy associated with the processing time information of the set of n operations is given by

$$H_p = H(G_1 G_2, \ldots, G_n) = \sum_{i=1}^{n} H(G_i) \tag{6}$$

from (3).

4. The schedule execution cost function

We will first define the following terms:

Schedule effectiveness: This is the degree of optimality of a schedule (initial or perturbed), for example makespan, flowtime, earliness, tardiness and so on.

Schedule predictability: This is the closeness of the perturbed schedule performance to that of the initial off-line schedule. The higher the predictability, the lower the costs of under-utilization or overtime.

Heuristic efficiency: The efficiency of a heuristic is measured by the computational complexity of the schedule generation/repair method, that is, the number of elementary computer operations needed to schedule a set of operations. This is related to the timeliness of response to perturbation events.

Rescheduling stability: This is the degree of rearrangement of operations (sequence, start-times and so on) after rescheduling. The importance of this measure is justified as follows: when disruptions such as machine breakdowns, order cancellations, arrival of new operations, changes in the operation processing times and so on occur, re-ordering the processing sequence of operations will incur costs due to re-allocation of tools, server machines and operators, rerouting of raw materials and additional schedule regeneration. A more stable rescheduling procedure reduces the costs of replanning and rerouting.

Scheduling nervousness: This is the frequency of H-rescheduling. The lower the frequency, the fewer the number of plan revisions in other parts of the supply chain.

Let

Ω be the set of perturbation times,

$Z(\omega_0)$ be the initial off-line schedule effectiveness,

$Z(\omega_i)$ be the perturbed schedule effectiveness at time $\omega_i \in \Omega$,
$\tilde{Z} = \{Z(\omega_i) : \omega_i \in \Omega\}$,
$\Delta\tilde{Z} = \{|Z(\omega_i) - Z(\omega_0)| : \omega_i \in \Omega\}$,
$E(\omega_i)$ be the rescheduling efficiency at time $\omega_i \in \Omega$,
$\Delta_\rho(\omega_i)$ be rescheduling stability at time $\omega_i \in \Omega$, and
N_H be the scheduling nervousness.

The comparison of deterministic, robust and online scheduling techniques will be based on the cost of executing the schedule for a given entropy value. The total schedule execution cost (TC) may consist of the following:

Perturbed schedule effectiveness cost: This is the cost related to the performance of the perturbed schedules, that is $G(\widetilde{Z})$, where $G(\cdot)$ is a function taken over $\widetilde{Z}$. For example, $G(\widetilde{Z})$ could be an average taken over the set $\widetilde{Z}$.

Schedule predictability cost: This is the cost related to the deviation in performance of the perturbed schedule from the initial off-line schedule, that is $J(\Delta\widetilde{Z})$, where $J(\cdot)$ is a function taken over $\Delta\widetilde{Z}$. This cost term is not applicable to online scheduling techniques, since online scheduling techniques do not predict schedules but commit operations over time.

Heuristic efficiency cost: This is the total cost of heuristic computation due to rescheduling over the schedule execution horizon. We denote the number of elementary computer operations needed to reschedule n_{ω_i} operations at time ω_i by $E(\omega_i)$. For online scheduling techniques, n_{ω_i} represents the number of operations to be committed at time ω_i.

Rescheduling stability cost: This is the cost related to the number of rearrangements needed each time a rescheduling (shift or H-rescheduling) is performed, that is $\Delta_\rho(\omega_i)$. Since the sequence of operations does not change for online scheduling techniques, $\Delta_\rho(\omega_i) = 0$ for all i.

Scheduling nervousness cost: This is the cost related to the number, N_H, of H-reschedulings used over the schedule execution horizon. This quantity can also be expressed as a fraction $\overline{N}_H = (N_H/|\Omega|)$, where $|\Omega|$ is the total number of disruptions at the end of the schedule execution. Generally, we assume that $|\Omega| \leq n$, where n is the total number of operations to be scheduled.

From this point on, Cost $(\cdot)$ denotes the cost of the parameter considered. The total schedule execution cost is expressed as follows:

$$TC_H = \text{Cost}[G(\widetilde{Z})] + \text{Cost}[J(\Delta\widetilde{Z})] + \sum_{i=1}^{|\Omega|} \text{Cost}[E(\omega_i)] + \sum_{i=1}^{|\Omega|} \text{Cost}[\Delta_\rho(\omega_i)] + \text{Cost}(N_H) \tag{7}$$

5. Simulation structure

We will investigate the case of makespan minimization of n non-pre-emptable operations on m identical parallel machines subjected to changes in operation

processing times. We define deterministic, robust and online scheduling techniques as follows:

- A heuristic H is applied in a deterministic sense if all operations are committed to an initial off-line schedule using heuristic H and when a disruption occurs, only a shift is performed.
- A heuristic H is applied in a robust sense if the heuristic is used in an α-lookahead scheme (the appendix for a description of α-lookahead scheme). We will assume $\alpha = 0$ for all experiments. All operations are committed to an initial off-line schedule using heuristic H and when a disruption occurs, one can decide to reschedule using heuristic H or perform a shift (SH) on the operations that have not started their processing. We will use the comparison metric $C'(SH, H)$ to decide whether to reschedule using heuristic H or SH when disruption occurs at time ω_i, where $C'(SII, II)$ is defined as follows,

$$C'(SH,H) = \phi'\left[\frac{Z(\omega_i)_H}{Z(\omega_i)_{SH}}\right] + \beta'\left[\frac{\Delta Z(\omega_i)_H}{\Delta Z(\omega_i)_{SH}}\right] + \gamma'\left[\frac{E(\omega_i)_H}{E(\omega_i)_{SH}}\right] + \nu'\left[\frac{(n^2_{\omega_i}/2) - \Delta_\rho(\omega_i)_H}{(n^2_{\omega_i}/2) - \Delta_\rho(\omega_i)_{SH}}\right] \tag{8}$$

where

$$\phi' = \frac{\phi}{1-\psi},\ \beta' = \frac{\beta}{1-\psi},\ \gamma' = \frac{\gamma}{1-\psi} \quad \text{and} \quad \nu' = \frac{\nu}{1-\psi},\ \phi+\beta+\gamma+\nu+\psi = 1$$

and ϕ, β, γ, ν and ψ are coefficients used in (9) (defined later). We will decide to shift if $C'(SH,H) \geq (1-\psi)$.

- A heuristic H is applied in an online sense, if no initial off-line schedule is created and operations are only committed to the schedule over time according to some specified rule. It should be noted that not all heuristics can be applied in an online sense, for example multifit heuristic.

We will introduce the following terms for ease of describing the simulation structure:

An entropy value, $H_p(i)$: This value is given by (6).

A set of bounds, $B(i, j)$: For a given entropy value $H_p(i)$, J sets of bounds will be generated. Each set of bounds $B(i, j)$ consists of n sets of intervals $(a_x^{(i,j)}, b_x^{(i,j)})$, where n is the total number of operations, $a_x^{(i,j)}$ and $b_x^{(i,j)}$ are the lower and upper bounds of the processing time of operation x, respectively, and $1 \leq j \leq J$.

A set of initial data, $I(i, j, k)$: For a given set of bounds $B(i, j)$, K sets of initial data will be generated. Each set of initial data $I(i, j, k)$ consists of n processing times, where the processing time of an operation x is randomly chosen within $a_x^{(i,j)}$ and $b_x^{(i,j)}$ according to its probability distribution.

An event: An event is described by a time and a set of perturbation parameters. The set of perturbation parameters for a change in processing times is given by a string of n binary numbers, that is $\{0, 1\}$. A value of 0 indicates that the processing time of the corresponding operation will not change (hence, will take the processing

time of the previous event) and 1 otherwise. When the value is 1, processing time of the corresponding operation will be resampled.

A perturbation: This consists of a set of events.

A set of perturbations: For a given set of bounds $B(i, j)$, a set of P perturbations will be generated. For each set of initial data $I(i, j, k)$, these P perturbations will be 'decoded' into actual processing times.

For each set of initial data, we will apply a chosen heuristic to obtain an initial off-line schedule, if the heuristic is applied in a deterministic or robust sense. The initial off-line schedule will be deployed and perturbed with its corresponding perturbation.

The terms in (7) will be normalized to enable a reasonable comparison, in expectation, between heuristics. Suppose we have a set, Γ, of heuristics to be compared, the normalized total schedule execution cost for heuristic $H \in \Gamma$ for a given entropy value $H_p(i)$,

$$\begin{aligned}\overline{TC}_{H,H_p(i)} = {} & \phi\left[\frac{\mathbf{E}(Z(\omega_{|\Omega|})_H)}{\max_{H\in\Gamma}\mathbf{E}(Z(\omega_{|\Omega|})_H)}\right] \\ & + \beta\left[\frac{\mathbf{E}(\Delta Z(\omega_{|\Omega|})_H)}{\max_{H\in\Gamma}\mathbf{E}(\Delta Z(\omega_{|\Omega|})_H)}\right] + \gamma\left[\frac{\mathbf{E}(\Sigma_{i=1}^{|\Omega|}E(\omega_i)_H)}{\max_{H\in\Gamma}\mathbf{E}(\Sigma_{i=1}^{|\Omega|}E(\omega_i)_H)}\right] \\ & + \nu\left[\frac{\mathbf{E}(\Sigma_{i=1}^{|\Omega|}\Delta_\rho(\omega_i)_H)}{\max_{H\in\Gamma}\mathbf{E}(\Sigma_{i=1}^{|\Omega|}\Delta_\rho(\omega_i)_H)}\right] + \psi\left[\frac{\mathbf{E}(\overline{N}_H)}{\max_{H\in\Gamma}\mathbf{E}(\overline{N}_H)}\right]\end{aligned} \tag{9}$$

where $\mathbf{E}(\cdot)$ is the mean over all $J \times K \times P$ values of the parameter considered, for a given entropy $H_p(i)$; $Z(\omega_{|\Omega|})_H$ and $\Delta Z(\omega_{|\Omega|})_H$ are the schedule makespan and predictability after the final perturbation; $E(\omega_i)_H$ and $\Delta_\rho(\omega_i)_H$ are the rescheduling efficiency and rescheduling stability at time ω_i; N_H is the scheduling nervousness, and $\phi + \beta + \gamma + \nu + \psi = 1$.

Recall that an online schedule technique does not create an initial off-line schedule and only commits operations over time. Therefore, for online scheduling technique, schedule predictability is not a relevant measure (predictability has to be evaluated with respect to an initial off-line schedule) and the overall schedule effectiveness can only be evaluated when all operations have been scheduled. For these reasons, $Z(\omega_{|\Omega|})_H$ is used as the measure of schedule effectiveness and $\beta = 0$ for all experiments. For the sake of simplicity we shall assume that $\gamma = 0$ for all experiments. We use a range of values for the other parameters, as set out in table 1. This set of cost coefficients was selected to ensure that the data points are well-distributed across the $\nu - \psi$ plane. This is the plane in which our results will be presented.

The total schedule execution costs of heuristics such as longest processing time (LPT), shortest processing time (SPT) and multifit (MF_k) applied in both a deterministic and robust sense will be compared. Only one heuristic will be applied in an online sense, namely the LPT heuristic. The following gives a brief description of the heuristics mentioned above:

LPT or SPT: The LPT heuristic selects the operation with the largest processing time in the unscheduled operations list and schedules it on the machine which is available first. On the other hand, unlike LPT, the SPT heuristic selects the operation with the smallest processing time in the unscheduled operations list. For LPT applied in an

Table 1. Cost coefficients used.

ϕ	β	γ	ν	Ψ
0	0	0	0	1
0	0	0	0.5	0.5
0	0	0	1	0
0.5	0	0	0	0.5
1	0	0	0	0
0.5	0	0	0.5	0
0.8	0	0	0.1	0.1
0.2	0	0	0.2	0.6
0.2	0	0	0.6	0.2
0.9	0	0	0	0.1
0.8	0	0	0	0.2
0.6	0	0	0.1	0.3
0.8	0	0	0.2	0
0.6	0	0	0.2	0.2
0.4	0	0	0.2	0.4
0.6	0	0	0.3	0.1
0.4	0	0	0.4	0.2
0.4	0	0	0.3	0.3
0.7	0	0	0.0	0.3
0.6	0	0	0.0	0.4

online sense, we will schedule the longest operation from the unscheduled operations list whenever a machine is available. At time $\tau = 0$, we will schedule the m largest operations from the set of initial data. This set of initial data will be updated according to the corresponding perturbation. By default, we will let $\overline{N}_H = 1$ for the online version of LPT, since LPT will be used whenever a machine becomes available. To be exact, LPT will be used n times, since there n operations to be committed over time. Also, we have earlier assumed that $|\Omega| \leq n$, where $|\Omega|$ is the total number of disruptions in the schedule execution horizon. When LPT or SPT is applied in a deterministic or robust sense, all operations are committed to the initial off-line schedule.

Multifit ($\mathbf{MF}_k$): Suppose we have n operations and $m \geq 2$ identical parallel machines. Let $b_j = (UB_j + LB_j/2)$, where LB_j and UB_j are the lower and upper bounds of the machine 'size' at the jth binary cut. Set

$$LB_1 = \max\left(p_{\max}, \frac{\Sigma_i p_i}{m}\right) \text{ and } UB_1 = 2\max\left(p_{\max}, \frac{\Sigma_i p_i}{m}\right),$$

where p_i is the processing time of operation i and $p_{\max} = \max_i(p_i)$. This is valid, since $C^{LS}_{\max} \leq (2 - (1/m))LB_1$ (Graham 1966). At each binary cut j, the 'size' of each machine is b_j. We fit the operations onto the machines of size b_j using FFD, as described in the next item. If the number of machines needed to fit all the operations is $> m$, then $LB_{j+1} = b_j$ and $UB_{j+1} = UB_j$, else $LB_{j+1} = LB_j$ and $UB_{j+1} = b_j$. The above procedures are repeated for $j \leq k$. At $j = k$, if the number of machines is $> m$, then schedule the operations on the machines with 'size' UB_k, else retain the schedule obtained with machines of size b_k. The schedule obtained at this step is the solution by MF_k.

First-fit-decreasing (FFD): Sort the n operations in non-increasing order of processing time and index the machines from 1 to m. Select the largest operation in the list and schedule it onto the machine with the smallest index on which it fits. If there is no such machine, add an extra machine and schedule the operation on it. Remove this operation from the list. Repeat these procedures until all operations are scheduled.

The multifit heuristic, MF_k, has to be slightly modified when it is used to reschedule. This is described as follows:

Rescheduling with multifit heuristic, MF_k: Suppose we have n not-yet-started operations and $m \geq 2$ identical parallel machines. Let $b_j = (UB_j + LB_j/2)$, where LB_j and UB_j are the lower and upper bounds of the machine 'size' at the jth binary cut. Set

$$LB_1 = \max\left(p_{\max} + C_{\min}, \frac{\sum_{i=1}^{n} p_i + \sum_{x=1}^{m} C_x}{m}\right),$$

and

$$UB_1 = 2\max\left(p_{\max} + C_{\min}, \frac{\sum_{i=1}^{n} p_i + \sum_{x=1}^{m} C_x}{m}\right),$$

where p_i is the processing time of operation i, $p_{\max} = \max_i(p_i)$ and $C_{\min} = \min_x(C_x)$. At each binary cut j, the 'size' of any machine x is $(b_j - C_x)$, that is, we assume that there is already an object of size C_x fitted. We fit the n operations onto the machines using FFD-MOD, as described in the next item. If the number of machines needed to fit all the operations is $> m$, then $LB_{j+1} = b_j$ and $UB_{j+1} = UB_j$, else $LB_{j+1} = LB_j$ and $UB_{j+1} = b_j$. The above procedures are repeated for $j \leq k$. At $j = k$, if the number of machines is $> m$, then schedule the operations on the machines with 'size' UB_k, else retain the schedule obtained with machines of size b_k. The schedule obtained at this step is the solution by MF_k.

Modified first-fit-decreasing (FFD-MOD): Sort the n operations in non-increasing order of processing time. Next, sort the machines in non-increasing order of $(b_j - C_x)$ and relabel the machines from 1 to m. Select the largest operation in the list and schedule it onto the machine with the smallest index on which it fits. If there is not such machine, add an extra machine with size $(b_j - \tau)$ and schedule the operation on it. Remove this operation from the list. Repeat these procedures until all operations are scheduled.

Table 2 lists the heuristics to be investigated for different values of entropy. Note that MF in the table represents MF_7.

Table 2. Heuristics to be investigated.

	Heuristics		
	MF	LPT	SPT
Deterministic sense	Yes	Yes	Yes
Robust sense	Yes	Yes	Yes
Online sense	No	Yes	No

Table 3. Bound and corresponding entropy values for the case $b_i - a_i = c$, $\forall i$, where $H_p = n \log (c/\Delta)$, and $\Delta = 0.001$.

$n = 30$		$n = 50$	
c	H_p	C	H_p
0.001	5.000	0.001	8.333
0.003	15.000	0.003	25.000
0.005	20.000	0.005	33.333
0.046	50.000	0.046	83.333
0.200	69.031	0.200	115.051
1.000	90.000	1.000	150.000
5.000	110.969	5.000	184.949
10.000	120.000	10.000	200.000
15.000	125.283	15.000	208.805
20.000	129.031	20.000	215.051
25.000	131.938	25.000	219.897
30.000	134.314	30.000	223.856
35.000	136.322	35.000	227.203
40.000	138.062	40.000	230.103
45.000	139.596	45.000	232.661
50.000	140.969	50.000	234.949
55.000	142.211	55.000	237.018
60.000	143.345	60.000	238.908
65.000	144.387	65.000	240.646
70.000	145.353	70.000	242.255
75.000	146.252	75.000	243.753
80.000	147.093	80.000	245.154
85.000	147.883	85.000	246.471
90.000	148.627	90.000	247.712
95.000	149.332	95.000	248.886
215.443	160.000	215.443	266.667
464.159	170.000	464.159	283.333
1000.000	180.000	1000.000	300.000
2154.435	190.000	2154.435	316.667
4641.589	200.000	4641.589	333.333

We will simulate a total of 30 different values of entropy (see table 3), with $J = 3$ sets of bounds for each entropy value and $K = 3$ sets of initial data for each set of bounds.

We assume all operations i are uniformly distributed between a_i and b_i, such that $f_i(w_i) = (1/b_i - a_i)$ and hence (4) can be rewritten as $P_i^k = (\Delta_i/b_i - a_i)$ for all k. Therefore, for operation i,

$$H(G_i) = -\sum_{k=1}^{z_i} \left(\frac{\Delta_i}{b_i - a_i}\right) \log\left(\frac{\Delta_i}{b_i - a_i}\right) = -\log\left(\frac{\Delta_i}{b_i - a_i}\right) \tag{10}$$

since $z_i = (b_i - a_i/\Delta_i)$. Hence the entropy for the n uniformly distributed processing times is given by

$$H_p = -\sum_{i=1}^{n} \log\left(\frac{\Delta_i}{b_i - a_i}\right) \tag{11}$$

for uniformly distributed processing times. A logarithm of base 10 will be used in our simulation.

For a set of initial data, a set of $P = 10$ perturbations with 10 events each was also generated. The event times are uniformly sampled from $U(0, T_{\max})$, where $T_{\max} = \max(180, (\Sigma c_i/2m))$ and $c_i = b_i - a_i$. This value of $T_{\max}$ is chosen to ensure the events are well distributed across the schedule execution horizon or makespan of the schedule. For large values of c_i, $T_{\max} = (\Sigma c_i/2m)$ is reasonable since on average the processing time of an operation is $(c_i/2)$ assuming $a_i = 0$ and for small values of c_i, we let $T_{\max} = 180$ since on average the lower bound of processing time of any operation is 50 (see (18)). At each event time, an operation's processing time will change with a probability of 0.5 and the processing time which it possesses is randomly selected according to the specified probability distribution within its bound. We will only perturb in-process operations and operations that have not yet started their processing.

6. Simulation results

We will investigate two cases, namely the case when $b_i - a_i = c, \forall i$ and a more general case where $b_i - a_i = c_i, \forall i$. For all cases, we generally assume $\Delta_i = \Delta$. For the case of $b_i - a_i = c$, we can further simplify (11) as follows:

$$H_p = n \log\left(\frac{c}{\Delta}\right) \tag{12}$$

and for the case where $b_i - a_i = c_i$, we rewrite (11) thus:

$$H_p = \sum_{i=1}^{n} \log\left(\frac{c_i}{\Delta}\right) \tag{13}$$

where we use $\Delta = 0.001$ for all simulations. Samples of bounds of operations for both cases are shown in figures 2 and 3.
For the case when $b_i - a_i = c_i$, the value of c_i is chosen such that

$$\log\left(\frac{c_i}{\Delta}\right) = \frac{H_p}{n} + \varepsilon \tag{14}$$

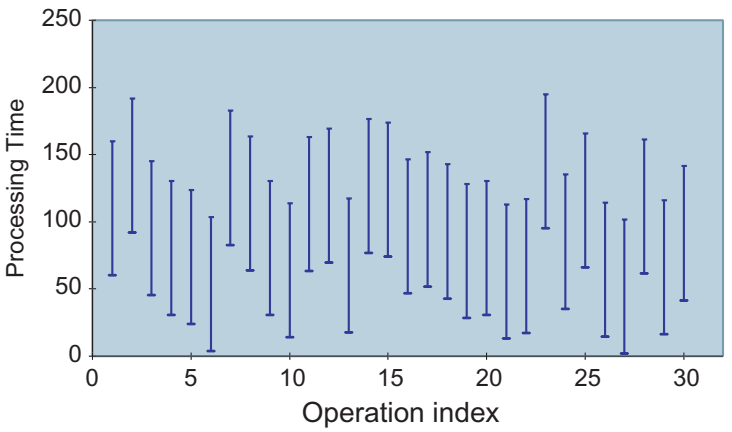

Figure 2. A sample of bounds of $n = 30$ operations when $b_i - a_i = c = 100$, $H_p = 150$ and $\Delta = 0.001$.

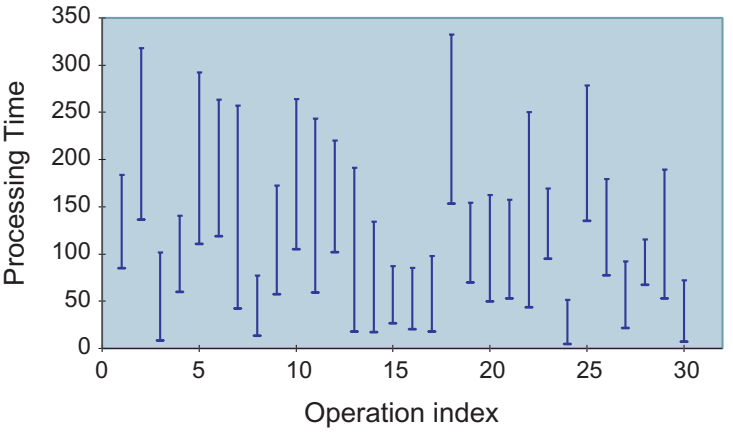

Figure 3. A sample of bounds of $n = 30$ operations when $b_i - a_i = c_i$ and $H_p = 150$ and $\Delta = 0.001$.

or

$$c_i = \Delta K^{((H_p/n)+\varepsilon)} = (\Delta K^{(H_p/n)})(K^{\varepsilon}) \tag{15}$$

where K is set to be 10, H_p is the entropy value and ε is a random number chosen from

$$U\left(-\frac{10}{n}, \frac{10}{n}\right) \text{ and } \mathbf{E}\left[\sum_{i=1}^{n} \log\left(\frac{c_i}{\Delta}\right)\right] = H_p.$$

Alternatively, one can select the value c_i such that

$$\log\left(\frac{c_i}{\Delta}\right) = \mathrm{U}\left(\frac{(2-\lambda)H_p}{n}, \frac{\lambda H_p}{n}\right) \tag{16}$$

or

$$\begin{aligned} c_i = \Delta K^{\mathrm{U}\left(((2-\lambda)H_p)/n), ((\lambda H_p)/n)\right)} &= \Delta K^{\left[(H_p/n)-(H_p/n)+\mathrm{U}\left(((2-\lambda)H_p)/n), (\lambda H_p/n)\right)\right]} \\ &= \Delta K^{\left[(H_p/n)+\mathrm{U}\left(((1-\lambda)H_p)/n), (((\lambda-1)H_p)/n)\right)\right]} \\ &= \left(\Delta K^{H_p/n}\right)\left(\Delta K^{\left[\mathrm{U}\left(((1-\lambda)H_p)/n), (((\lambda-1)H_p)/n)\right)\right]}\right) \\ &= \left(\Delta K^{H_p/n}\right)(K^{\varepsilon'}) \end{aligned} \tag{17}$$

where $\lambda > 1$ and ε is a random number chosen from $\mathrm{U}((1-\lambda)H_p/n, (\lambda-1)H_p/n))$. Both methods yield the same mean value, that is $\mathbf{E}[\Sigma_{i=1}^{n} \log(c_i/\Delta)] = H_p$. However, the value of ε in (17) is dependent on (H_p/n), that is the variability of c_i values for a set of operations increases with H_p/n. For large H_p/n, the bounds for a set of operations will be too diverse. At worst, we will obtain a set of bounds with $(n-1)$ operations with $c_i \approx 0$ and one operation with very large c_i. This is clearly unrealistic. We therefore prefer using (15) to (17).

The lower bound for any operation i is given by

$$a_i \overset{d}{=} \mathrm{U}[1, \max(100, c_i)] \tag{18}$$

We will demonstrate the usefulness of using the term max(100, c_i) via the following examples:

- Suppose we replace max (100, c_i). For very small values of c_i, we will obtain very small a_i values and the a_i values for a set of operations will be too similar, resulting in operations which are stochastically too similar.
- Now suppose we replace max(100, c_i) with 100, or with some other constants. For very large c_i, typically $c_i \gg 100$, the value of 100 would seem insignificant and therefore the a_i values for a set of operations will be too similar.
- Suppose we can find a constant C which is much larger than any possible c_i values, that is $C \gg c_i$, $\forall i$. We replace max(100, c_i) with C. This will then cause large diversity in processing times of a set of operations, since a_i can take values between 1 and C. At worst, the processing time of one operation could be extremely large and the others extremely small. With this set of operations, the performance of any heuristics will be the same, since the performance (makespan) is more likely to be determined by the operation with the largest processing time.

Therefore, in order to, on the one hand, minimize similarity and on the other hand to minimize extreme diversity of a set of operations, we have chosen max(100, c_i). The largest value of c_i is much greater than 100 (see table 3) and therefore minimizes extreme diversity in a set of operations. Also, the max function minimizes the similarity in a set of operations.

Until further notice, we will assume $n = 30$ and $m = 6$ for all our simulations. We will only display the dominant (the technique which has the smallest $\overline{TC}_{H,e}$) scheduling technique (deterministic, robust or online) from our simulations on a $\nu - \psi$ plane, since $\beta = \gamma = 0$ for all cases and $\phi = 1 - \nu - \psi$. The order of preference when two or more techniques possess the same normalized cost $\overline{TC}_{H,e}$ is given by online, followed by deterministic and finally robust, since the online scheduling technique is the simplest.

6.1 *Simulation results for $b_i - a_i = c$, $\forall i$*

Figure 4 shows the dominant scheduling techniques for various entropy values when $b_i - a_i = c$, $\forall i$ and $n = 30$. Note that Cost (Nh) and Cost ($D -$ rho) axes represents ψ and ν axes respectively.

Clearly, for low entropy values, that is for $(H_p/n) \leq 1.7$ or for $c \leq 0.046$, the deterministic technique is preferable. Our results show that the deterministic version of MF_7 dominates in the region of low entropy values. This is not surprising, since MF_7 dominates in the region of low entropy values. This is not surprising, since MF_7 is designed to produce more effective (lower makespan) solutions than either LPT or SPT when applied in a deterministic sense.

For high entropy values, that is for $(H_p/n) \geq 4.8$ or for $c \geq 65$, there exists a clear region distinguishing deterministic and online techniques. Typically, deterministic techniques (MF_7) dominate when ψ is large and ν is small and online techniques dominate when ψ is small and ν is large, since deterministic techniques do not require any rescheduling and the sequence of operations for online techniques does not change.

Robust techniques only dominate when $\nu = 0$ and ψ is small. This is because, at high ψ it is too expensive to reschedule using a heuristic when a disruption occurs,

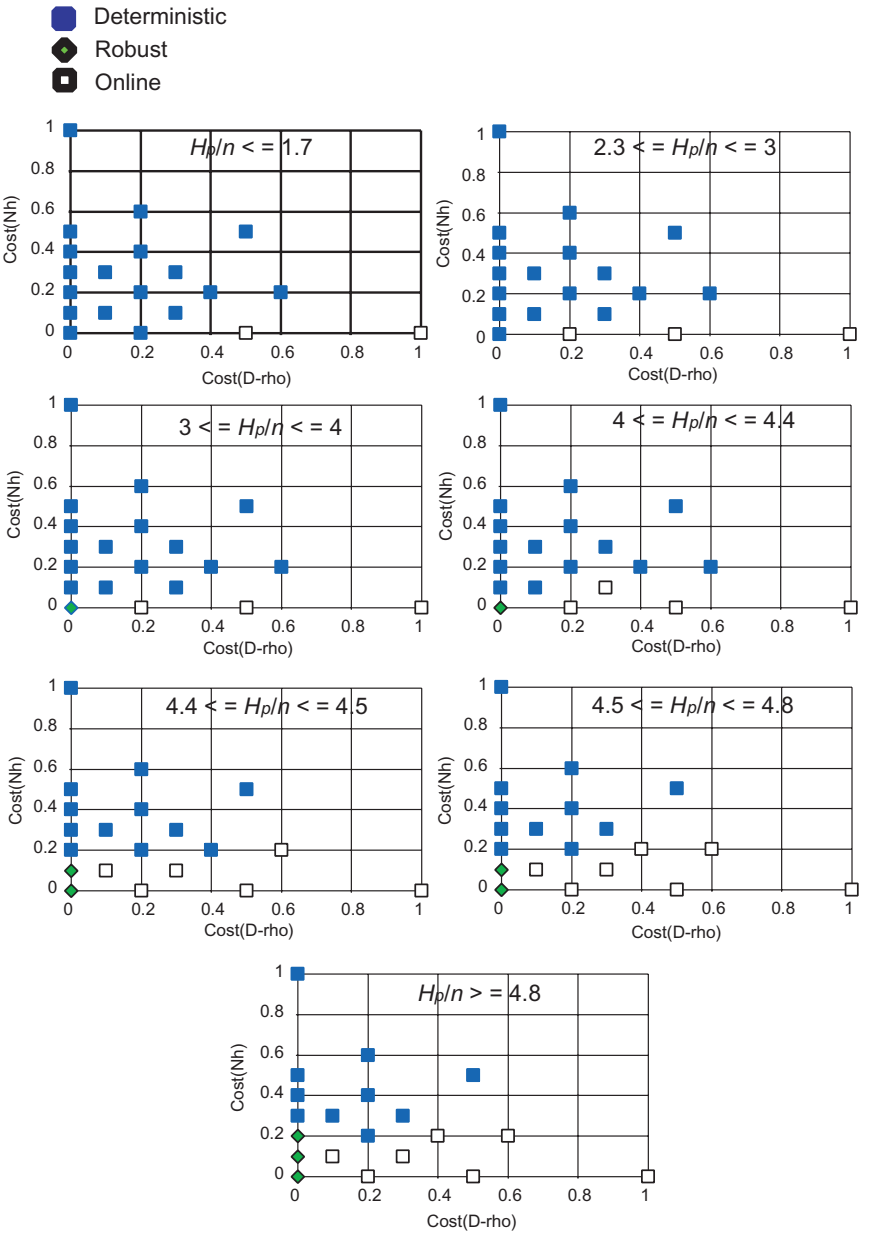

Figure 4. Simulation results for various entropy values when $b_i - a_i = c$, $\forall i$ and $n = 30$.

and hence a robust technique will behave like a deterministic one for high ψ. SPT applied in the robust sense dominates for this combination of cost coefficients.

6.2 *Simulation results for* $b_i - a_i = c_i$, $\forall i$

Figure 5 shows the dominant scheduling techniques for various entropy values when $b_i - a_i = c_i$, $\forall i$ and $n = 30$. The dominance trends for both low ($(H_p/n) \leq 2$) and high

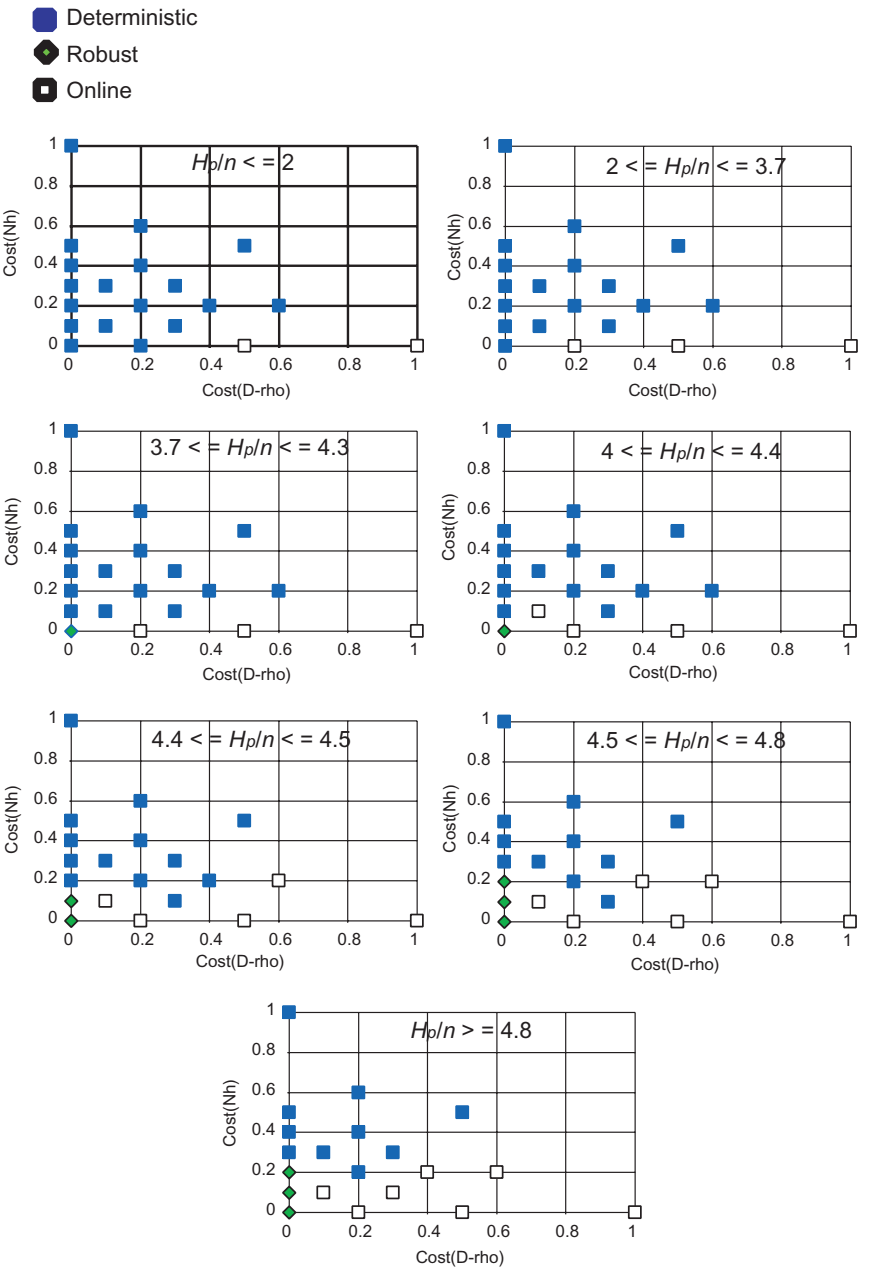

Figure 5. Simulation results for various entropy values when $b_i - a_i = c$, $\forall i$ and $n = 30$.

$((H_p/n) \leq 4.8)$ entropy values are similar to that observed for the case when $b_i - a_i = c$, $\forall i$. For low entropy values, MF_7 applied in a deterministic sense dominates, as observed previously.

For high entropy values, unlike the previous case where the robust version of SPT dominates, when $v = 0$ and ψ is small, the robust version of LPT dominates for this case. This observation demands further investigation into the effect of bound

types (uniform and non-uniform) on the selection of heuristics applied in a robust sense. Despite this difference, we are still able to conclude that a robust technique should be applied for this combination of cost coefficients for high entropy values.

Simulation results suggest that the 'boundaries' between deterministic, robust and online techniques can be drawn without much loss in accuracy, by simply assuming that $c_i = c$, $\forall i$.

6.3 *Scalability of the entropy measure*

Figure 6 shows the dominant scheduling techniques for various entropy values when $b_i - a_i = c_i$, $\forall i$ and $n = 50$. Earlier simulations for $n = 30$ when $b_i - a_i = c_i$, $\forall i$ yield similar trends (see figure 4), that is deterministic techniques dominate for low entropy values ($(H_p/n) \leq 1.7$) and the dominance regions of all three techniques (deterministic, robust and online) are visible for high entropy values ($(H_p/n) \geq 4.8$). The crossover values† and dominance transition patterns‡ for both $n = 30$ and $n = 50$ are similar. Also, the robust version of SPT outperforms other heuristics in regions where a robust technique dominates, as observed for $n = 30$.

This result suggests a possible scalability property of the entropy measure, namely a smaller sample of operations can be used to anticipate the dominance patterns and crossover points. This implies that it is possible to anticipate the sensitivity of scheduling techniques to cost coefficients with a shorter simulation runtime.

7. Conclusions

We have proposed the use of entropy measure (6) to justify deterministic, robust and online scheduling techniques. Simulations have been performed on the problem of scheduling n operations on m identical parallel machines when the processing times of operations are uncertain, and are uniformly distributed. Our simulation results suggest two important conjectures regarding the properties of this entropy measure, namely

1. The dominance transition patterns can be predicted by making the simplifying assumption, that is $b_i - a_i = c$, $\forall i$, instead of $b_i - a_i = c_i$, $\forall i$, without much loss in accuracy.
2. The entropy measure is scalable with respect to the number of operations and hence gaining improvement in simulation runtime, since smaller problem sizes can be simulated to obtain similar crossover values and dominance transition patterns to those for larger problem sizes.

†A crossover value is the value of (H_p/n) for which a change in dominance of scheduling technique occur. For example, in figure 6, $(H_p/n) = 1.7$ is a crossover value for $\psi = 0$ and $\nu = 0.2$, since for values < 1.7, deterministic technique dominates and for values > 1.7, online technique dominates.

‡Dominance transition patterns are transition patterns observed in the neighbourhood of a crossover point. For example, figures 4 and 6 show similar transition pattern in the neighbourhood of $(H_p/n) = 1.7$, since the dominant technique changes from deterministic to online for $\psi = 0$ and $\nu = 0.2$ and the dominance behaviour for other combinations of cost coefficients remains unchanged.

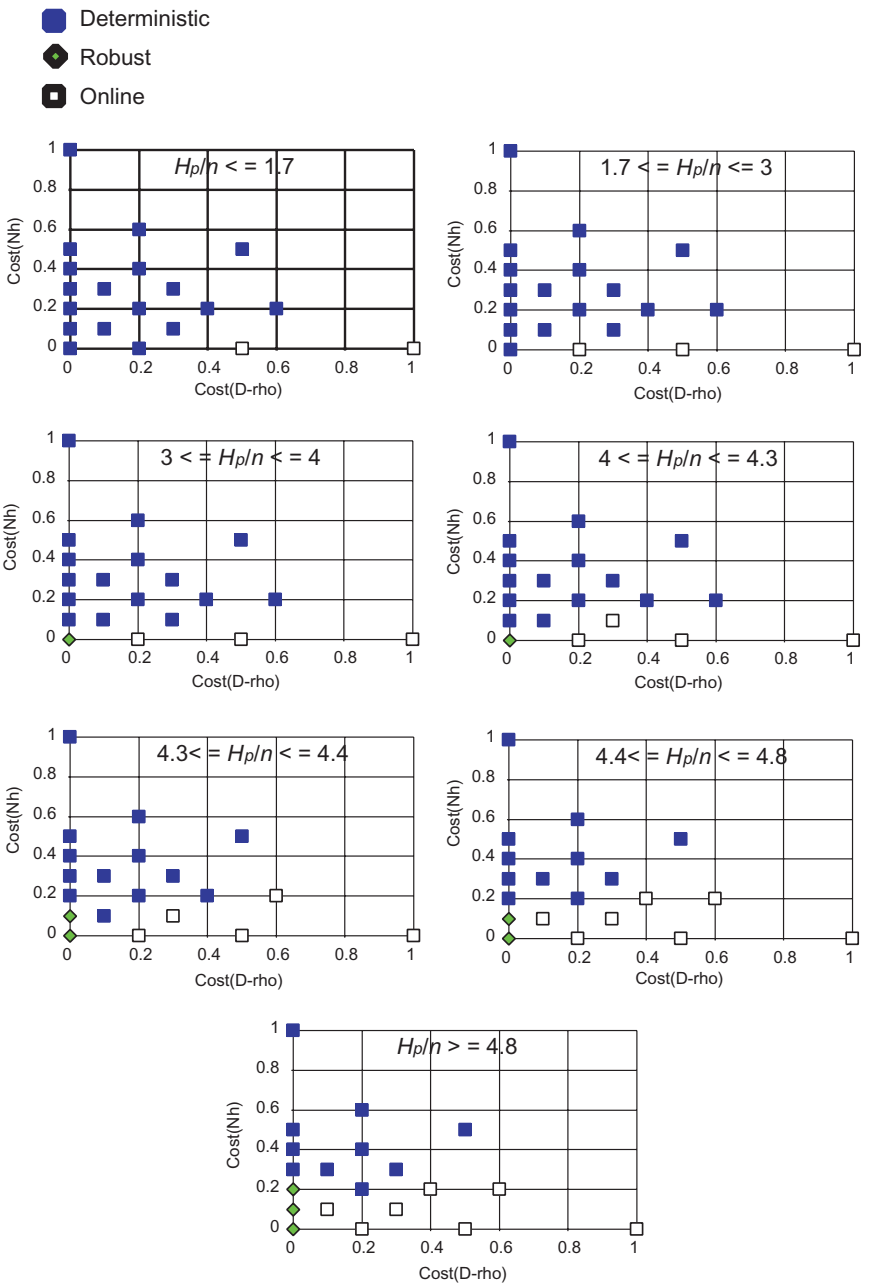

Figure 6. Simulation results for various entropy values when $b_i - a_i = c$, $\forall i$ and $n = 50$.

However, a detailed analysis and simulations should be carried out before the above conjectures can be confirmed. Furthermore, our simulations results are restricted to identical parallel machines and to uniformly distributed operation processing times. Therefore, further investigations on other shop floor configurations and probability distributions of processing times should be performed.

Table 4. Performance ranking of different versions of LPT, where 1 represents the best and 3 the worst performance.

	Effectiveness	Stability	Efficiency	Nervousness
Deterministic	3	3	1	1
Robust	1	2	2	2
Online	2	1	3	3

It may be argued that figures 4, 5 and 6 suggest that a robust scheduling technique is of marginal importance. We believe such a conclusion is not warranted, for the following reasons:

- In all experiments, we have only used the α-lookahead scheme, where $\alpha = 0$. In previous experiments, we found that the performance of an α-lookahead scheme for $\alpha > 0$ is significantly better than that of $\alpha = 0$. Also, the robust scheduling scheme proposed for ρ-heuristics generally outperforms the α-lookahead scheme. Therefore, the importance of a robust scheduling technique will be amplified if a better scheme is used.
- Recall that we have only considered three measures of performance in all experiments, namely schedule effectiveness, rescheduling stability and scheduling nervousness. The dominance of a robust scheduling technique will be significant if the measure of heuristic efficiency is considered in the total schedule execution cost. Consider the heuristic efficiencies of different versions of LPT for the case of $n \gg m$, where n is the total number of operations to be scheduled and m is the number of machines. The efficiency of the deterministic, robust and online versions of LPT are given by $O(n|\Omega|)$, $O(n|\Omega| \log n)$ and $O(n^2 \log n)$,† respectively, where $|\Omega|$ is the total number of perturbations in the schedule execution horizon and we generally assume that $|\Omega| \leq n$. Table 4 shows the relative performance of different versions of LPT for various performance measures.‡ From this table, we conjecture that a robust scheduling technique will be important if all four measures are important.

Apart from uncertain processing times, uncertainty associated with disruptions such as machine breakdowns, arrival of new operations and removal of operations can also be described by the entropy measure. For example, machine breakdowns can be described by their mean time to failure ($MTTF$) and breakdown duration (D). Both $MTTF$ and D can be expressed in terms of some probability distribution (see (4)), and if independence is assumed, we can add the entropy values of these two quantities. Another possible extension of our work is to investigate the dominance transition patterns for dependent schemes.

†Recall that an online scheduling technique commits operations over time. Each time an operation is committed, the unscheduled list of operations has to be sorted again, since there may be changes in processing time. The sorting algorithm, which runs in $O(n \log n)$ time, will be used n times.

‡These rankings are obtained by observation of the results obtained from other experiments carried out by the authors. Details of these experiments have been omitted from this paper due to their length.

In fact, the use of an entropy measure in our context can be perceived, in a broader sense, as a proactive approach to deal with changes in the level of information uncertainty and relative importance of schedule execution cost elements. The level of information uncertainty may change due to the performance deterioration of processors (machines or human) and the replacement of old machines with new ones; and the changes in relative importance of cost elements may be due to changes in shop floor priorities and pressure from partners in the supply chain network. One can decide upon the scheduling strategies to be employed based on the latest entropy value of the information considered and the relative importance of cost elements.

Appendix: The α-lookahead scheme

We will describe the α-lookahead approach for a one heuristic robust scheduling framework, that is, we will decide to use a heuristic, H, to reschedule or perform a shift, SH, when a disruption occurs. We will use heuristic H to generate the initial off-line schedule for this scheme.

Let $H(\omega_i)$ is the rescheduling decision made when a disruption occurs at time ω_i, that is, $H(\omega_i) = H$ when heuristic H is used to reschedule and $H(\omega_i) = SH$ when a shift is performed. Let $E(\omega_i)$ be the actual perturbation event that occurs at time ω_i and $E_\lambda(\omega_j^\lambda)$ be the sampled perturbation event at time ω_j^λ under scenario $\lambda \in \Lambda$. For instance, when considering uncertain operation processing times, a perturbation event, $E(\omega_i)$ or $E_\lambda(\omega_j^\lambda)$ can be described by a string of operation processing times to be updated at time ω_i or ω_j^λ respectively. It is interesting to note that $|\Lambda| = p^N$, where p is the number perturbation samples and N is the depth of tree in Leon *et al.*'s (1994) game-theoretic controller. The sequence of disruptions on any one path in the DEDS tree of the controller corresponds to one scenario in Λ.

Suppose that the cureent time is ω_k and that an event $E(\omega_k)$ has occurred. A preliminary procedure for the α-lookahead approach is to sample the possible events from ω_{k+1}^λ to $\omega_{k+\alpha}^\lambda$ for all $\lambda \in \Lambda$, except for $\alpha = 0$ where no sampling is required. For a scenario λ, the α-lookahead approach can be represented by a tree consisting of H and SH nodes as shown in figure 7. The arc connecting any two nodes represents two consecutive rescheduling decisions. Now let $\theta_i^\lambda \in \boldsymbol{\Theta}^\lambda$ be a path branching out from $H(\omega_k) = H$ node and $\pi_i^\lambda \in \boldsymbol{\Pi}^\lambda$ be a path branching out from the $H(\omega_k) = SH$ node, and both θ_i^λ and π_i^λ are the ith element of Θ^λ and Π^λ respectively. There are 2^α possible paths in each set of Θ^λ and Π^λ for any scenario λ. Also, let $obj(\cdot)$ be the objective function of the path considered.

Given any scenario λ, we wish to find

$$\mathrm{Cost}_\theta^\lambda = \underset{\theta_i^\lambda \in \boldsymbol{\Lambda}^\lambda}{G} [obj(\theta_i^\lambda)] \tag{19}$$

and

$$\mathrm{Cost}_\pi^\lambda = \underset{\pi_i^\lambda \in \boldsymbol{\Pi}^\lambda}{G} [obj(\pi_i^\lambda)] \tag{20}$$

where $G(\cdot)$ is a function of the set of paths considered. For example, if the 'max' function is used, our objective function would at worst be Z_θ^λ, if heuristic H is used

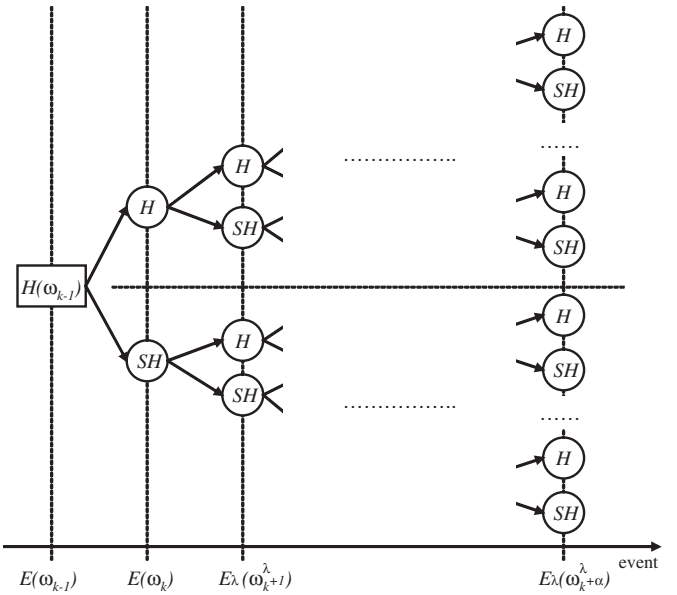

Figure 7. Tree representation of the α-lookahead approach for scenario λ.

to reschedule at time ω_k when scenario λ is actually realized. Thus, the objective function of the α-lookahead procedure is given by

$$OBJ = \min\left[\underset{\lambda\in\Lambda}{F}(\mathrm{Cost}_{\theta}^{\lambda}) \quad \underset{\lambda\in\Lambda}{F}(\mathrm{Cost}_{\pi}^{\lambda}) \right] \tag{21}$$

where $F(\cdot)$ is a function of the set of scenarios considered. We will decide to use H at time ω_k if $\lambda \in \Lambda(\mathrm{Cost}_{\theta}^{\lambda}) < \lambda \in (\mathrm{Cost}_{\pi}^{\lambda})$. Therefore, the rescheduling policy is a cost-driver type. The cost-evaluation procedure will be invoked when a disruption occurs.

The α-lookahead procedure for one heuristic robust scheduling framework can be perceived as a 'special case' of Leon *et al.*'s (1994) game-theoretic controller where $N=2$. However, the evaluation of the α-lookahead tree is a more general and straightforward procedure than that proposed by Leon *et al.* (1994).

References

Braun, O., Lai, T.C., Schmidt, G. and Sotskov, Y.N., Stability of Johnson's schedule with respect to limited machine availability. *Int. J. Prod. Res.*, 2002, **40**, 4381–4400.

Daniels, R.L. and Kouvelis, P., Robust scheduling to hedge against processing time uncertainty in single-stage. *Manage. Sci.*, 1995, **41**, 363–376.

Graham, R.L., Bounds for certain multiprocessing anomalies. *Bell Syst. Tech. J.*, 1966, **45**, 1563–1581.

Johnson, S.M., Optimal two and three-stage production schedules with setup times included. *Naval Res. Logist. Quart. 1*, 1954, 61–68.

Khinchin, A.I., *Mathematical Foundations of Information Theory*, translated by R.A. Silverman and M.D. Friedman, 1957 (Dover Publications Inc.: New York).

Kouvelis, P., Daniels, R.L. and Vairaktarakis, G., Robust scheduling of a two-machine flow-shop with uncertain processing times. *IIE Trans.*, 2000, **32**, 421–432.

Kouvelis, P. and Yu, G., *Robust Discrete Optimisation and Its Application* (Vol. 14 of *Nonconvex Optimisation and Its Applications*), 1997 (Dordrecht: Kluwer Academic Publishers).

Kuo, C.Y. and Lin, F.J., Relative robustness for single-machine scheduling problem with processing time uncertainty. *J. Chinese Inst. Indust. Eng.*, 2002, **19**, 59–67.

Lai, T.C., Sotskov, Y.N., Sotskova, N. and Werner, F., Optimal makespan scheduling with given bounds of processing times. *Math. Comput. Model.*, 1997, **26**, 67–86.

Lawrence, S.R. and Sewell, E.C., Heuristic, optimal, static and dynamic schedules when processing times are uncertain. *J. Oper. Manage.*, 1997, **15**, 71–82.

Leon, V.J., Wu, S.D. and Storer, R.H., A game-theoretic control approach for job shops in the presence of disruptions. *Int. J. Prod. Res.*, **32**, 1451–1476.

Matsuura, H. and Kanezashi, M., Makespan comparison between resequencing and switching in a dynamic manufacturing environment. *Int. J. Prod. Econ.*, 1996, **44**, 137–149.

Sotskov, Y.N., Sotskova, N.Y. and Werner, F., Stability of an optimal schedule in a job shop. *Omega, Int. J. Manage. Sci.*, 1997, **25**, 397–414.

Sotskov, Y.N., Wagelmans, A.P.M. and Werner, F., On the calculation of the stability radius of an optimal or an approximate schedule. *Annals Oper. Res.*, 1998, **83**, 213–252.

Yang, J. and Yu, G., On the robust single machine scheduling problem. *J. Combin. Optimiz.*, 2002, **6**, 17–33.