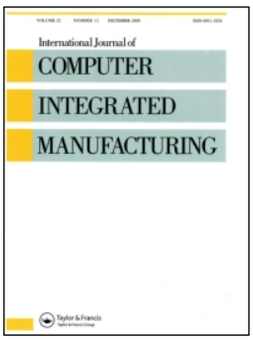

International Journal of Computer Integrated Manufacturing

ISSN: 0951-192X (Print) 1362-3052 (Online) Journal homepage: http://www.tandfonline.com/loi/tcim20

# Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research

I. Sabuncuoglu & S. Goren

**To cite this article:** I. Sabuncuoglu & S. Goren (2009) Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research, International Journal of Computer Integrated Manufacturing, 22:2, 138-157, DOI: 10.1080/09511920802209033

**To link to this article:** https://doi.org/10.1080/09511920802209033

Published online: 28 Jan 2009.

*International Journal of Computer Integrated Manufacturing*
Vol. 22, No. 2, February 2009, 138–157

Taylor & Francis
Taylor & Francis Group

# Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research

I. Sabuncuoglu* and S. Goren

*Department of Industrial Engineering, Bilkent University, Ankara, Turkey*

(*Received 11 February 2007; final version received 22 March 2008*)

Scheduling is a decision-making process that is concerned with the allocation of limited resources to competing tasks (operations of jobs) over a time period with the goal of optimising one or more objectives. In theory, the objective is usually to optimise some classical system performance measures such as makespan, tardiness/earliness and flowtime under deterministic and static assumptions. In practice, however, scheduling systems operate in dynamic and stochastic environments. Hence, there is a need to incorporate both uncertainty and dynamic elements into the scheduling process. In this paper, the major issues involved in scheduling decisions are discussed and the basic approaches to tackle these problems in manufacturing environments are analysed. Proactive scheduling is then focused on and several robustness and stability measures are presented. Previous research on scheduling robustness and stability is also reviewed and further research directions are suggested.

**Keywords:** scheduling; proactive; robustness; stability; uncertainty

## 1. Introduction

Scheduling plays an important role in achieving timely and cost-effective production, which is becoming increasingly important in today's highly competitive manufacturing environments. In an industrial setting, scheduling systems usually operate in highly dynamic and uncertain environments in which several interruptions (mostly random in nature) prevent the execution of production schedules exactly as they are developed. Examples of such disruptions are machine breakdowns, rush orders, order cancellations, due-date changes, scraps and waste owing to machine malfunctions, etc. Variation in processing times and other stochastic events further increase variability in the system.

Even though real problems are dynamic and stochastic in nature, most of the solutions in the literature use static and deterministic models. In theory, most scheduling problems, even those with deterministic and static assumptions are NP-hard or mathematically intractable. For that reason, heuristic procedures are generally recommended for practical applications.

Practitioners often view the ignorance of uncertainty and the dynamic elements of the scheduling process as the major source of the gap between scheduling theory and practice. In the last two decades researchers have closed this gap by proposing several scheduling systems under various names: on-line scheduling, dynamic scheduling, real-time scheduling, etc. (Sabuncuoglu and Kizilisik 2003). Some of these systems have been developed using artificial intelligence and expert system tools from computer science as well as OR tools.

Recently, two new approaches have emerged as an alternative way to cope with uncertainty in a scheduling environment: *reactive* and *proactive* scheduling. These scheduling policies will be discussed in detail in subsequent sections.

The purpose of this paper is to develop a tool to understand the philosophy of proactive and reactive scheduling better on a high level (in terms of what major decisions are made during a scheduling process and how these decisions lead to different scheduling policies). There are other papers in the literature which review the scheduling process under uncertainty. Herroelen and Leus (1995) and Davenport and Beck (2000), for example, review the literature on a low level (as a collection of techniques that are used to generate and revise schedules). Aytug *et al*. (2005) also includes a taxonomy on uncertainty, but the scheduling process itself is just broadly classified into proactive and reactive. The work of Vieira *et al*. (2003) might be considered closer to the approach taken in this paper. Their review is more general in the sense that the authors cover the entire rescheduling process whereas the emphasis in this paper is specifically on robustness and stability concepts. The emphasis in this

*Corresponding author. Email: sabun@bilkent.edu.tr

ISSN 0951-192X print/ISSN 1362-3052 online

DOI: 10.1080/09511920802209033
http://www.informaworld.com

paper is not particularly on the classification into proactive and reactive scheduling (which is well known in the literature) but on what major decisions lead to these two policies and on how robustness and stability are used together with them to cope with uncertainty.

This paper also analyses the proactive scheduling problem with the help of a scenario planning approach. This approach provides a link to decision theory, which not only enables one to understand robustness and stability measures that are used in the literature better, but also presents a way to define several new robustness and stability measures analogous to some criteria from decision theory.

The rest of the paper is organised as follows. In Section 2 two major decisions (when-to-schedule and how-to-schedule) in scheduling and appropriate policies to deal with the real life problems are discussed. In Section 3, reactive and proactive scheduling policies are focused on. In Section 4, the proactive scheduling problem is analysed under a scenario planning approach. Several robustness and stability measures that are used in the literature are presented. New robustness and stability measures are formulated by making use of the similarity between the developed scenario planning framework and decision theory. In Section 5, the existing studies in the literature are reviewed. Finally, concluding remarks are made and future research directions are recommended in Section 6.

## 2. Two major issues of scheduling

In highly dynamic and stochastic production environments, shop-floor schedules cannot be used as they are developed for a long time because of unexpected disruptions and random events that alter the state of the system. Thus, it is often necessary to revise the schedules at some points in time. In this context, two immediate questions arise: when to revise (when-to-schedule) and how to revise (how-to-schedule)?

### *2.1. When-to-schedule*

'When-to-schedule' has to do with the timing of scheduling decisions and determines the system responsiveness to various kinds of effects from the environment. As scheduling frequency increases, the system responsiveness also increases. There are several alternative ways to decide on timing of scheduling decisions. The first one, called *periodic scheduling*, schedules the system periodically; the period length can be constant or variable. In the constant case which is often used in practice, revisions are made at the beginning of each fixed-time interval. However, according to the variable-time interval method, scheduling decisions are made after a certain amount of schedule is realised (Sabuncuoglu and Karabuk 1999).

Another alternative could be to revise the schedule following a certain number of random events. For example, the schedule can be updated after each major machine breakdown, or a new important job arrival. This method of scheduling is called *continuous scheduling* (Raman *et al*. 1989). Another method is *adaptive scheduling* (Sabuncuoglu and Karabuk 1999), which may also be called *controlled response*. According to this policy, a scheduling decision is triggered following a predetermined amount of deviation from the original schedule. For example, a revision is made when the total difference in completion times between the initial and realised schedules exceeds some threshold value, say 30 min, or some percentage of the makespan. Similarly, schedules can be revised after a certain amount of deviation from the planned throughput, flowtime, or tardiness is observed.

In addition, several hybrid methods can also be considered. For example, Yamamoto and Nof (1985) propose a scheduling policy, called *event-driven scheduling*, where revisions are not only made at the end of each fixed time interval (i.e., periodic scheduling), but also in response to important events that change the system state (i.e., continuous scheduling). See Church and Uzsoy (1992) for comparison of periodic and continuous scheduling policies for dynamic shops.

Even though the when-to-schedule decisions are analysed to some extent by Sabuncuoglu and Kizilisik (2003), the subject needs further research. Specifically, it would be interesting to know the conditions under which a particular policy is better than others

### *2.2. How-to-schedule*

'How-to-schedule' determines the ways in which schedules are generated and updated. There are four related issues. The first one has to do with the scheduling scheme. This can be off-line, on-line, or a combination of the two (i.e., hybrid). *Off-line scheduling* refers to scheduling all operations of available jobs for the entire scheduling period, before executing the schedule; in *on-line scheduling*, decisions are made one at a time, during the execution of schedule (Sabuncuoglu and Hommertzheim 1992). A good example for on-line scheduling is the implementation of dispatching rules in a dynamic environment. Between these two extremes, another alternative could be *quasi-online scheduling*, in which a subset of the operations of the job set are scheduled and the rest are left for future time periods (Sabuncuoglu and Karabuk 1999, Wu *et al*. 1999).

On-line approach accommodates considerable flexibility in the schedule to compensate for unforeseen

system disturbances but lacks the global perspective provided by an off-line approach. Therefore, analysing both approaches' strengths and weaknesses and identifying the circumstances under which one performs better than the other is a valuable research topic. It is known that off-line scheduling is superior to on-line scheduling in a static and deterministic environment. In a static and dynamic environment, off-line scheduling is still better, but the difference between the performances of these scheduling schemes is not as large as in the static case (Sabuncuoglu and Karabuk 1999). However, a further analysis of off-line scheduling and on-line scheduling methods is needed in a dynamic and stochastic environment that includes the consideration of robustness and stability.

The second issue is the amount of data used during the schedule generation process. Kutanoglu and Sabuncuoglu (2001) define the forecasting horizon (*FH*) as the time span of job-release data. It represents the maximum time period for which schedulers have enough information to generate a schedule. Look-ahead window (*LW*) is defined as the portion of *FH* for which a new schedule is generated or a revision is made. It can be smaller than *FH* or equal to *FH*. If it is smaller than *FH*, only a part of the available information is used. This is generally attributable to low confidence about the accuracy of the far-future information. In this case only near-future information is used. If *LW* is equal to *FH*, all available information is used and this is called *full scheduling*. When $LW < FH$, it is called *partial scheduling*. Partial scheduling is illustrated in Figure 1, where horizontal trail represents the time and vertical arrows mark the scheduling points. In the figure, successive forecasting horizons are depicted with dashed curves while solid curves represent successive look-ahead windows. Note that if full scheduling is employed with periodic review and if *FH* is equal to the period length, this policy corresponds to doing nothing (i.e., leaving the system alone and letting it recover from disruptions).

The existing studies indicate that full scheduling is superior to partial scheduling in a static environment because of its global perspective and avoidance of myopic decision-making (Sabuncuoglu and Bayiz 2000). On the other hand, the relative performance of full scheduling and partial scheduling in a dynamic and stochastic environment is needed for both robustness and stability measures.

The third and fourth issues are *type of response* and *performance metrics* to use, respectively, which are discussed now in detail.

### *1. Type of response/nature of revision*

One can identify at least two cases: 1) rescheduling the operations of all the remaining jobs from scratch and 2) taking no corrective action and letting the system recover itself from the negative effects of disruptions. Between these two extremes, it is also possible to repair the schedules. One possible repair method could be to generate a match-up schedule, where at some point in the future, the new schedule and the original one become the same/converge. (Bean *et al*. 1991, Akturk and Gorgulu 1999). Another method is right/left shifting of all the remaining jobs (altogether) in the time horizon so that the disruption length is accommodated but the sequence of jobs remains unchanged (Abumaizar and Svestka 1997). Another minor revision could be to change only a small number of jobs' positions in the schedule. The type of response issue

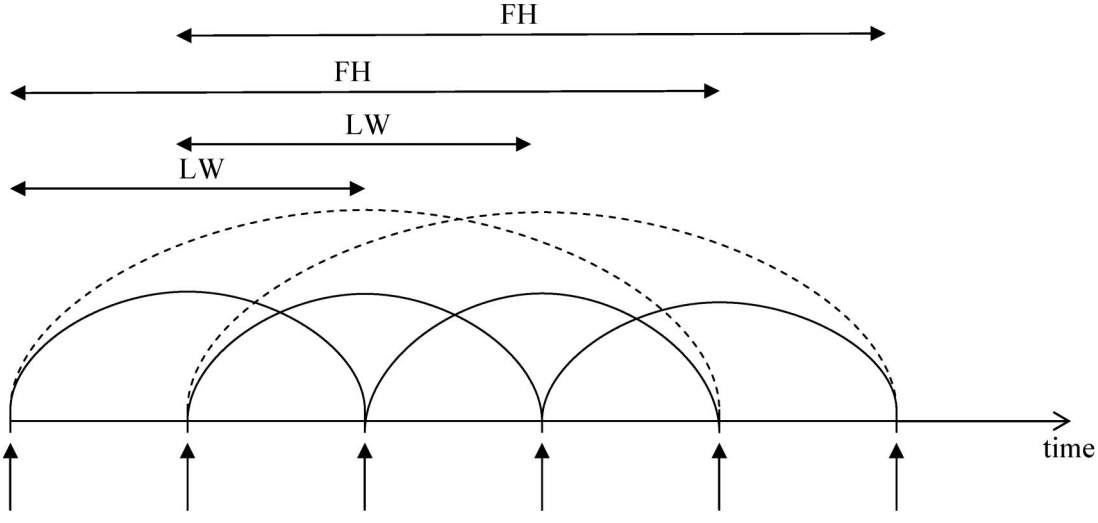


Figure 1. Partial scheduling.

has not been thoroughly studied. Even though there are some studies that analyse rescheduling frequency (Church and Uzsoy 1992, Sabuncuoglu and Karabuk 1999), the relative weaknesses and strengths of different response types are not systematically studied.

*2. Performance metrics*

Finally, the scheduler should decide on which performance metric to use. Classical performance measures such as makespan, flowtime, earliness, or tardiness are often preferred in practice. In the recent literature, two new measures have also emerged: *robustness and stability*. These are used particularly in the environments where uncertainty is a major issue.

In practice, a typical scheduling process is as follows. An *initial* schedule is generated to guide future shop floor activities. In the face of random disruptions such as machine breakdowns, order cancellations, due-date changes, and scraps and waste, etc or variations in processing times, this schedule needs to be partly or completely revised to maintain its feasibility. The schedule which is actually executed on the shop floor is called the *realised schedule*. This schedule may substantially differ from the initial schedule, depending on the level of disruptions and changes in the environment. Robustness and stability are related with this difference.

Robustness is concerned with the difference in terms of objective function value. It refers to the insensitivity of scheduling performance to the disruptions. A schedule whose performance does not deteriorate much in the face of disruptions is called *robust*. In general, the performance of the realised schedule is the main concern of practitioners rather than the planned or estimated performance of the initial schedule. This is because the former is reality while the latter is just an anticipated course of actions.

On the other hand, stability is concerned with the difference between initial and realised schedules themselves, rather than between their performances. A schedule whose realisation does not deviate much from the initial schedule in the face of disruptions is called *stable*. A schedule also serves as the plan for other production activities, such as determining delivery dates, releasing times, planning requirements for secondary resources such as tools, fixtures, etc (Wu *et al*. 1993). Any deviation from the initial schedule can disrupt the plans for such activities and increase *system nervousness*. Thus, stability is an important performance metric in practice.

Even though existing studies hint that robustness and stability are conflicting objectives (e.g. Wu *et al*. 1993) the nature of trade-off between these performance metrics requires further research. Similarly, it would be interesting to know the circumstances under which one metric should be given more priority than the other.

Sotskov *et al*. (1997) discuss another perception of stability. They handle the uncertainty in a job shop environment by an *a posteriori* analysis, in which an optimal schedule has already been constructed and the question is to determine the maximum variation in the processing time of the operations such that the optimal schedule at hand remains still optimal. Such a maximum variation is called 'the stability radius' of the schedule. This notion of stability, obtained by sensitivity analysis, can be considered as a measure of 'solution robustness' in terms of Herroelen and Leus (2005). Although this type of post optimality analysis may provide some valuable insights about the impacts of the uncertainty, it is also associated with some problems. If the 'stability radius' of the optimal schedule is large enough to accommodate all possible changes in the processing times the optimal schedule at hand can safely be used, but if it is not that large, the question of what course of actions to take still remains to be answered.

Several robustness and stability measures are defined later in Section 4.2. When-to-schedule and how-to-schedule (scheduling scheme, amount of data, type of response and performance metric) decisions are entitled as *parameters* of a scheduling system, as in setting parameters of a scheduling system will yield *scheduling policies*, which are explained in Section 3. Scheduling parameters are summarised in Figure 2.

### *2.3. Reacting to disruptions*

In the previous section, 'type of response' was briefly mentioned when discussing 'how-to-schedule'. In this section, response types are further analysed in detail. Table 1 shows a list of disruptions and the possible responses that can be used to cope with these unexpected events. The first column is the list of disruptions. Responses are classified in three categories: 1) *Do nothing* – taking no corrective action, 2) *Reschedule* – rescheduling all operations of available jobs from scratch, and 3) *Repair* – making minor modifications to the existing schedule. Note that 'reschedule' column in Table 1 corresponds to the response of rescheduling all available jobs from scratch. That is, although different disruptions have responses with different names in that column, they all refer to the same approach in essence. Their names are different because the responses are named after disruption types for which they are developed.

In Table 2, the response types for each disruption are also proposed. The 'X' entry in the cell means that this combination of when-to-schedule and response

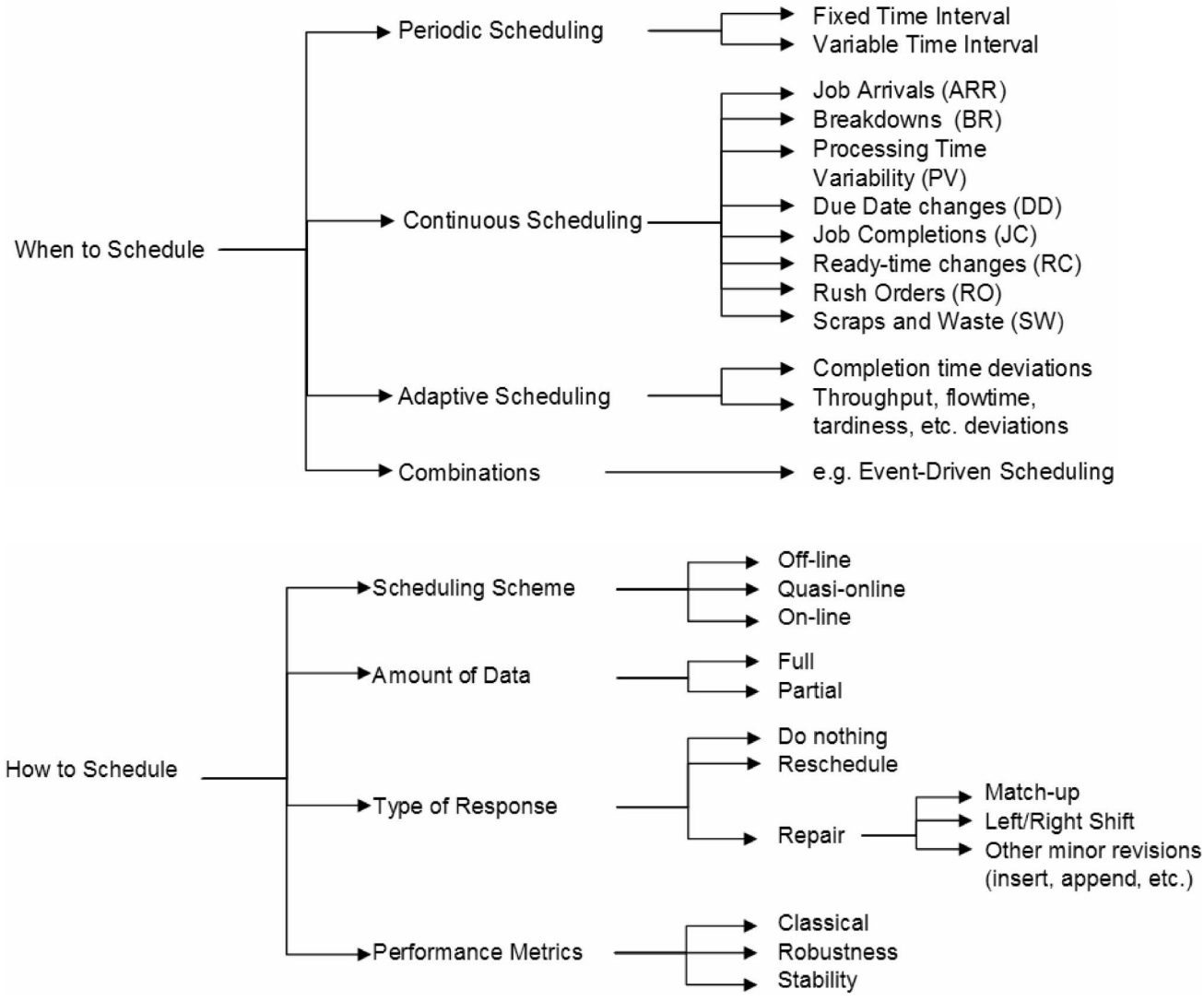


Figure 2. Scheduling parameters.

type can be used together, otherwise the cell is left empty. For example, according to periodic scheduling, revisions are made only after a fixed or variable time period after the previous scheduling point. Thus, if some disruption occurs in between, no corrective action is taken and the next scheduling point is awaited. Therefore, only the ARR1 (Do nothing) response is applicable if the when-to-schedule decision is periodic scheduling. The corresponding cell has the 'X' entry and other cells related to unexpected arrivals (ARR2, ARR3 and ARR4) are left empty. In another example, RO2 inserts the rush order in the first available position in the existing schedule. If the when-to-schedule policy is periodic scheduling, no action is taken and the next scheduling point is awaited. Hence, RO2 cannot be used with periodic scheduling and the corresponding cell is left empty. However, if the when-to-schedule policy is continuous or adaptive scheduling (which reacts after a certain amount of deviation from the planned schedule), RO2 can be used to react to rush order(s). Hence, the corresponding cells are marked with an 'X'.

In summary, periodic scheduling is suitable with only 'do nothing' type of response, whereas adaptive scheduling can be used with all types. Similarly, for a given kind of disruption, all types of response for that kind can be used with the associated continuous scheduling scheme.

Recall that some hybrid schemes of when-to-schedule are considered in the literature (e.g. Yamamato and Nof 1985). Similarly, each type of response listed in Table 2 can be used together with a combination of the suitable (the ones marked with an 'X') when-to-schedule methods.

The information in these two tables can be very useful for both practitioners and researchers.

Table 1. Disruptions and possible responses.

| Disruptions | Do nothing | Repair | Reschedule |
|---|---|---|---|
| Unexpected Arrivals (ARR) | Wait until the next scheduling point (ARR1) | • Append at the end of existing schedule (ARR2)<br>• Insert in a suitable position in the existing schedule (ARR3) | Reschedule all the available jobs from scratch (ARR4) |
| Machine Breakdowns (BR) | Right Shift (BR1) | • Match-up schedule (BR2) | Reschedule all the available jobs from scratch (BR3) |
| Processing time variability (PV) | Right Shift/Left-Shift (PV1) | | Reschedule all the available jobs from scratch (PV2) |
| Due-Date changes (DD) | Wait until the next scheduling point (DD1) | • Find a suitable position for the job whose due-date has been changed (DD2) | Reschedule all the available jobs from scratch (DD3) |
| Job cancellations (JC) | Make to stock (JC1) | • Delete the job that is cancelled in the existing schedule, left shift the portion after the deleted job (JC2) | Reschedule all the available jobs from scratch (JC3) |
| Ready-time changes (RC) | Left or Right Shift (RC1) | • Find a suitable position whose ready time has been changed in the existing schedule (RC2) | Reschedule all the available jobs from scratch (RC3) |
| Rush orders (RO) | | • Add the rush order to the beginning of the existing schedule (RO1)<br>• Insert the rush order in the first available position in the existing schedule (RO2) | Reschedule all the available jobs from scratch (RO3) |
| Scraps and waste (SW) | Wait until the next scheduling point, consider a new job in place of the scrap/waste then (SW1) | • Append a new job in place of the scrap/waste at the end of the existing schedule (SW2)<br>• Insert a new job for the scrap/waste in the existing schedule (SW3) | Reschedule all the available jobs from scratch (SW4) |

Table 2. Cross table for type of response/when to schedule.

| | When To | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Type of Response | | | Continuous | | | | | | | |
| How To | Periodic | Adaptive | ARR | BR | PV | DD | JC | RC | RO | SW |
| ARR1 | X | X | X | X | X | X | X | X | X | X |
| ARR2 | | X | X | | | | | | | |
| ARR3 | | X | X | | | | | | | |
| ARR4 | | X | X | | | | | | | |
| BR1 | X | X | X | X | X | X | X | X | X | X |
| BR2 | | X | | X | | | | | | |
| BR3 | | X | | X | | | | | | |
| PV1 | X | X | X | X | X | X | X | X | X | X |
| PV2 | | X | | | X | | | | | |
| DD1 | X | X | X | X | X | X | X | X | X | X |
| DD2 | | X | | | | X | | | | |
| DD3 | | X | | | | X | | | | |
| JC1 | X | X | X | X | X | X | X | X | X | X |
| JC2 | | X | | | | | X | | | |
| JC3 | | X | | | | | X | | | |
| RC1 | X | X | X | X | X | X | X | X | X | X |
| RC2 | | X | | | | | | X | | |
| RC3 | | X | | | | | | X | | |
| RO1 | X | X | X | X | X | X | X | X | X | X |
| RO2 | | X | | | | | | | X | |
| RO3 | | X | | | | | | | X | |
| SW1 | X | X | X | X | X | X | X | X | X | X |
| SW2 | | X | | | | | | | | X |
| SW3 | | X | | | | | | | | X |
| SW4 | | X | | | | | | | | X |

Practitioners can formulate correct policies and researchers can conduct theoretical studies to develop appropriate models and solution procedures.

## 3. Scheduling approaches: reactive versus proactive scheduling

In a scheduling environment, it is desirable that realised schedules have high system performance and that they do not deviate significantly from initial schedules. To achieve these objectives and cope with the uncertainty in scheduling processes two policies are considered: *reactive* scheduling and *proactive* scheduling.

*Reactive scheduling* does not directly consider the uncertainty in generating schedules, but revises the schedule when unexpected events or disruptions occur. In other words, reactive scheduling aims at finding the ways to (ideally) optimally *react* to disruptions after they occur. The reaction generally takes the form of either modifying the existing initial schedule (repairing), or generating a completely new schedule from scratch. In the studies which consider both robustness and stability, the primary concern when reacting to a disruption, generally, is to minimise the deviation between the new schedule and the initial schedule (i.e., optimising stability) although minimising degradation in the performance measure (robustness) can also be considered in addition (e.g. Wu *et al*. 1993).

*Proactive scheduling* on the other hand, considers future disruptions when generating initial schedules. It is concerned with generating an initial schedule that minimises the effects of disruptions on the performance measures; robustness is usually the primary concern although optimising stability can also be considered or even be the primary objective (e.g. Mehta and Uzsoy 1998, 1999).

In Figure 3, four possible implementations are proposed based on these two policies. Note that proactive and reactive scheduling can also be combined: the initial schedule can be created in a proactive manner and then the disruptions can be handled in a reactive manner (e.g. O'Donovan *et al*. 1999). And as Figure 3 shows, there is even a possibility that none of them will be used (called *classical scheduling*); in this case, the schedule acts as a guideline rather than an operational tool.

Setting the values of the scheduling parameters discussed in Section 2 may lead to various scheduling policies. Table 3 presents some examples of these policies resulting from 'when-to-schedule' and 'how-to schedule' decisions. The first four columns show 'how-to-schedule' decisions and the fifth column shows the 'when-to-schedule' decision. That is, the first five columns display *scheduling parameters*. The last column contains the type of the *scheduling policy* that is obtained using the said values of the scheduling parameters. For example, if an off-line scheduling scheme with full amount of data is used, a 'do nothing' type of response is adopted, a classical performance measure is used and the schedule is revised periodically, a classical scheduling policy is obtained (first row of Table 3). Similarly, if an on-line scheduling scheme with partial amount of data is used and the values of the remaining scheduling parameters are kept the same, another classical scheduling policy is obtained (second row of Table 3). To differentiate the latter from the former, they are given different names (Classical 2 and Classical 1, respectively). Note that Table 3 is not an exhaustive list of all possible scheduling policies; it just provides some examples.

The majority of applications in practice are classical; some newer ones are reactive. It is rare to see the stand-alone use of proactive scheduling or its hybrid use with reactive policies. This is partly attributable to the fact that theoretical developments in proactive scheduling are still under their way. This paper will discuss these existing studies and propose some research directions.

## 4. Proactive scheduling

As explained before, proactive scheduling aims at finding a schedule which is good in the face of disruptions. The schedule is not necessarily optimal for the assumed initial problem but it should do well under dynamic and stochastic conditions. In this section, the ways to generate such schedules are discussed. Several measures for proactive scheduling are also presented.

### *4.1. Notations and terms*

Let $P$ be the original scheduling problem. Suppose one has $m$ feasible solutions (schedules), $S_1$ through $S_m$. In most scheduling problems, $m$ is quite a large number. In the classical scheduling theory, the aim is to find an optimal schedule $S_0^*$, according to a selected performance measure such as maximum lateness, maximum completion time, average flow-time, etc. Let $f(S)$ denote the value of this performance measure for schedule $S$. A schedule $S_0^*$ with $f(S_0^*) \leq f(S_i)$ for $i = 1, \ldots, m$, is selected (i.e. a schedule which minimises the performance measure). In what follows, possible disruptions in actual scheduling processes are considered. As illustrated in Figure 4, suppose that there are $n$ scenarios corresponding to possible disruptions. A $0^{\text{th}}$ scenario $P_0$, which corresponds to no disruptions, is also considered; therefore, there are $n + 1$ scenarios in total. The original problem $P_0$ changes into $P_j$ for

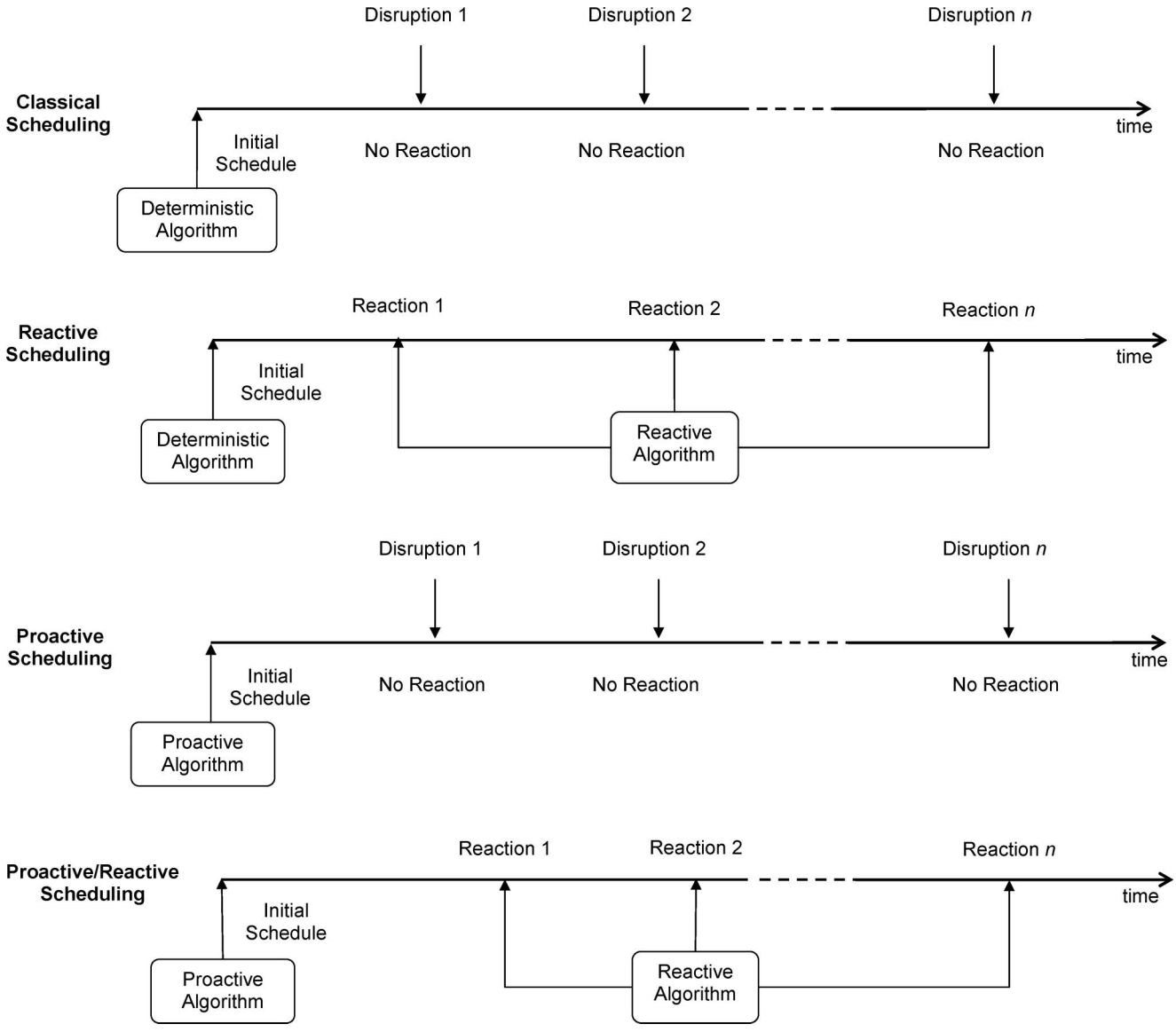


Figure 3. Scheduling policies.

Table 3. Examples of scheduling parameters and policies.

| | HOW TO | | | | |
|---|---|---|---|---|---|
| Scheduling Scheme | Amount of Data | Type of Response | Performance Metric | WHEN TO | Resulting Policy |
| Off-line | Full | Do nothing | Classical | Periodic | Classical 1 |
| On-line | Partial | Do nothing | Classical | Periodic | Classical 2 |
| Off-line | Full | Do nothing | Robustness | Periodic | Proactive 1 |
| Off-line | Full | Do nothing | Stability | Periodic | Proactive 2 |
| Off-line | Full | Reschedule | Classical | Periodic | Reactive 1 |
| Off-line | Full | Repair (Match-up) | Classical | Continuous | Reactive 2 |
| Quasi On-line | Full | Reschedule | Robustness | Periodic | Proactive/Reactive |

scenario $j$ with probability $a_j$. Accordingly, schedule $S_i$ changes into schedule $S_{ij}$ under $j^{th}$ scenario. That is, $S_{ij}$ is the realised version of the initial schedule $S_i$, if the disruptions in scenario $j$ occur. Let $S_j^*$ be the optimal solution to the problem $P_j$, with the objective function value of $f(S_j^*)$.

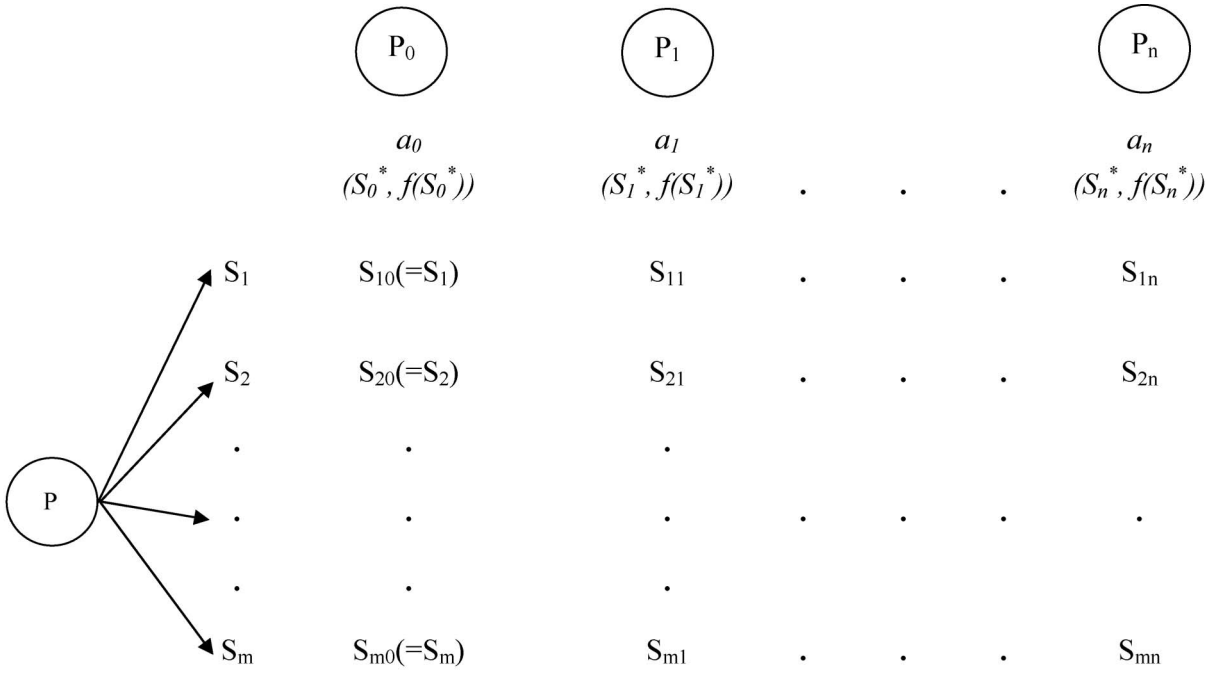


Figure 4. Scenario-based representation of disruptions.

If scenario $j$ were known to occur, one would be looking for the schedule $S_j^*$. However, it is not known which scenario will actually occur in advance. Moreover, it may be extremely difficult to find $S_j^*$ for $P_j$ due to the complexity of the scheduling problem. As explained before, there are numerous performance measures considered in the scheduling literature (tardiness, flowtime, makespan, etc.).

### 4.2. *Robustness and stability policies and measures*

If the feasible solutions $S_1$ through $S_m$ are taken as *actions* of the decision maker and the scenarios $P_0$ through $P_n$ are taken as *states* of the nature, the decision theoretic criteria can easily be used to generate several robustness or stability measures, of which a selection is presented in this section. If the mentioned robustness or stability measures are already discussed in the literature, a reference is also provided. The measures without a reference are new in this study to the best of the authors' knowledge. These measures can be seen as new formulations which use the existing principles from the decision theory.

#### 4.2.1. *Robustness*

Some robustness measures are based on the actual performance of the realised schedules, $f(S_{ij})$ and some are based on *regrets*. The *regret* associated with the initial schedule $S_i$ and the scenario $j$ is defined as the difference between realised and optimal performances, i.e., $f(S_{ij}) - f(S_j^*)$. The former aims at selecting a schedule with a good realised performance, whereas the latter tries to select a schedule whose performance is not bad relative to the best performance.

*4.2.1.1. Robustness policies and measures based on realised performances.*

(1) Minimise the expected realised performance.

$$S^* = S_t, t \in \arg\min_i \{E[\mathrm{f(S_i)}]\},$$

$$\text{where } E[\mathrm{f(S_i)}] = \sum_{\mathrm{j=0}}^{\mathrm{n}} \mathrm{a_j} f(S_{ij})$$

This method selects the schedule whose performance measure is the best on average. This is a risk-neutral approach. This is the most frequently (almost exclusively) used realised performance based robustness measure in the literature. (e.g. Wu *et al.* 1999).

(2) Minimise the worst-case performance.

$$S^* = S_t, t \in \arg\min_i \left\{ \max_j \{\mathrm{f(S_{ij})}\} \right\}$$

This policy selects the schedule whose worst-case performance measure is better than all others (e.g. Artigues *et al.* 2005 use this measure to evaluate a given ordered group assignment). This is a risk-averse approach.

(3) Minimise worst-case scenario's performance, if it is known.

$$S^* = S_t, t \in \arg\min_i \{\mathrm{f(S_{ik})}\}$$

where $k$ is the worst cast scenario.

(4) Minimise most probable scenario's performance.

$$S^* = S_t, t \in \arg\min_i \{\mathrm{f(S_{im})}\}$$

$$\text{where } m \in \arg\max_j \{a_j\}$$

(5) Another policy is to select a schedule such that the expected deviation of the realised schedule's performance from the initial deterministic performance is minimised.

$$S^* = S_t, t \in \arg\min_i \{\sigma_\mathrm{i}\}$$

$$\text{where } \sigma_\mathrm{i} = |f(S_{i0}) - E[f(S_i)]|$$

This policy emphasises the definition: 'the robust schedule is the one whose performance degrades minimally in the face of disruptions' (e.g. Leon *et al*. 1994).

(6) Minimise the variance of realised performance measure.

$$S^* = S_t, t \in \arg\min_i \{Var[\mathrm{f(S_i)}]\},$$

$$\text{where } Var[\mathrm{f(S_i)}] = \sum_{\mathrm{j}=0}^{\mathrm{n}} \mathrm{a_j} f^2(S_{ij}) - \left(\sum_{\mathrm{j}=0}^{\mathrm{n}} \mathrm{a_j} f(S_{ij})\right)^2$$

The main principle of this policy is to find the schedule whose performance degradation is minimal in the face of disruptions, as in the previous measure (e.g. Sevaux and Sörensen 2004).

(7) Another policy is to select the schedule that minimises a measure that is a convex combination of aforementioned measures, for example, selecting the schedule which minimises $rE\left[f\left(S_i^r\right)\right] + (1-r)\sigma_i$, where $r$ is a real number between 0 and 1. The first part of this measure emphasises the fact that a robust schedule should perform well in the face of disruptions. The second part emphasises that a robust schedule's performance should not degrade much in the face of disruptions. By varying $r$ between 0 and 1, different weights can be given to these two aspects of robustness (Leon *et al*. 1994).

*4.2.1.2. Robustness policies and measures based on regret*. Robustness measures in Section 4.2.1.1 determine the robustness of a schedule based on its realised performance. Another way to achieve robustness is to minimise opportunity losses (regrets). First, the regret matrix is constructed, $\Delta = [\delta_{ij}]_{mxn}$, where $\delta_{ij} = |f(S_{ij}) - f(S_j^*)|$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The $ij^{th}$ element of this matrix is the difference between optimal performance of the $j^{th}$ scenario and the realised performance of the $i^{th}$ schedule in this scenario.

(1) Minimise expected regret.

$$S^* = S_t, t \in \arg\min_i \{E[\delta_\mathrm{i}]\},$$

$$\text{where } E[\delta_\mathrm{i}] = \sum_{\mathrm{j}=0}^{\mathrm{n}} \mathrm{a_j}\delta_{ij}$$

This method selects a schedule whose regret is the least on average. This is a risk-neutral approach.

(2) Minimise the worst-case regret.

$$S^* = S_t, t \in \arg\min_i \{\max_j \{\delta_\mathrm{ij}\}\}$$

This is a risk-averse approach and is called the *minimax regret method*. This is the most frequently (almost exclusively) used regret based robustness measure in the literature (e.g. Daniels and Kouvelis 1995).

(3) Minimise worst-case scenario's regret.

$$S^* = S_t, t \in \arg\min_i \{\delta_{ik}\}$$

where $k$ is the worst cast scenario.

This policy selects $S_i$ with $\min_i \{\delta_{ik}\}$, where $P_k$ is the worst-case scenario.

(4) Minimise most probable scenario's regret.

$$S^* = S_t, t \in \arg\min_i \{\delta_{im}\}$$

$$\text{where } m \in \arg\max_j \{a_j\}$$

As discussed in Daniels and Kouvelis (1995), scheduling decisions may be evaluated *ex post*, as if all disruptions had been known in advance of scheduling. In this situation, the decision is concerned with how realised performance compares with the optimal performance that could have been achieved if perfect information had been available. In such a situation the use of regret-based robustness measures is more appropriate.

### 4.2.2. Stability

Estimating the impact of schedule changes is a difficult task. Several measures based on differences between operation starting/completion times, number of disrupted operations, or number of scheduling changes

made have been proposed in the literature. Differences in completion times of tasks are used to account for the impact of schedule change, for it is the most popular approach to measure stability in the literature. Let $C_k^S$ be the completion time of job $k$ under schedule $S$. Let $N$ be the number of jobs. Construct the completion-time differences matrix $D = [d_{ij}]$ where $d_{ij} = \sum_{k=1}^{N} |C_k^{S_{ij}} - C_k^{S_{i0}}|$.

(1) Minimise expected differences.

$$S^* = S_t, t \in \arg\min_i \{E[d_i]\},$$

$$\text{where } E[\delta_i] = \sum_{j=0}^{n} a_j d_{ij}$$

This method selects a schedule whose stability is the best on average. This is a risk-neutral approach. This measure is the most frequently used stability measure in the literature (e.g. Mehta and Uzsoy 1998, 1999, O'Donovan *et al.* 1999). Leus and Herroelen (2005) prove NP-hardness of minimising the weighted version of this measure under single machine breakdown, right-shift rescheduling and common due-date assumptions on several single machine and parallel machine settings.

(2) Minimise expected squares of differences.

$$S^* = S_t, t = \arg\min_i E[d_i^2],$$

$$\text{where } E[d_i^2] = \sum_{j=0}^{n} a_i d_{ij}^2$$

This method selects a schedule whose stability is the best on average. This is a risk-neutral approach. This measure is analogous to 'mean squared error' measure used in statistics.

(3) Minimise total variance of realised completion times:

$$S^* = S_t, t = \arg\min_i \sum_{k=1}^{N} Var[C_k^{r_i}]$$

where $C_k^{r_i}$ is the realised completion time of job $k$ under schedule $i$.

$$\text{Therefore } Var[C_k^{r_i}] = \sum_{j=1}^{n} a_j (C_k^{S_{ij}})^2 - \left( \sum_{j=1}^{n} a_j C_k^{S_{ij}} \right)^2.$$

(4) Minimise the worst-case completion-time difference.

$$S^* = S_t, t \in \arg\min_i \{\max_j \{d_{ij}\}\}$$

This policy selects a schedule whose worst-case stability is greater than all others. This is a risk-averse approach.

(5) Minimise worst case scenario's completion-time difference.

$$S^* = S_t, t \in \arg\min_i \{d_{ik}\}$$

where $k$ is the worst cast scenario.

(6) Minimise most probable scenario's completion-time difference.

$$S^* = S_t, t \in \arg\min_i \{d_{im}\}$$

$$\text{where } m \in \arg\max_j \{a_j\}$$

If the number of scenarios is finite, the calculation of robustness and stability measures, and thus the application of the above policies, is easy. In practice, however, the scenarios cannot be easily defined or determined in advance; in fact the number of potential scenarios is infinite, with a large number of alternative schedules. Thus, the computational burden of calculating robustness or predictability measures can be quite high. A reasonable approach could be to use a good surrogate measure (an easy-to-compute auxiliary measure used instead of the actual robustness or stability measure, which is known to be highly correlated with the actual measure itself) and to determine an efficient scheduling algorithm so that the selected schedule optimises this surrogate measure. Another approach could be employing simulation to estimate the values of robustness or stability measure. Simulation, in fact, is used to evaluate the performances of schedules under uncertainty in the literature (e.g. Kutanoglu and Sabuncuoglu 2001). Most studies that address robustness or stability, however, avoid using simulation owing to the computational burden it introduces.

In some cases, although there are an infinite number of scenarios, a procedure to generate the worst case scenario can be available. In such cases, there is no need to define a surrogate measure or use simulation (see Daniels and Kouvelis 1995, for such an example).

## 5. Recent studies in scheduling robustness and stability

In this section, the scheduling studies which explicitly address robustness and stability by defining measures for them and optimise these measures are focused on. These studies are mostly proactive in nature. Table 4 summarises these studies using the classification framework based on scenario analysis. In 'Environment' section of Table 4, 'Shop Floor' column contains the type of the shop floor. 'Sta./Dyn.' column displays the job arrival characteristic of the scheduling problem under study. If all jobs are present and ready to

Table 4. Recapitulation of studies.

| | Environment | | | Schedule Generation | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | How to | |
| Reference | Shop Floor | Sta./Dyn. | Stoch./Det. | Method | Objective | When to | Scheme | Response |
| Wu *et al.* (1993) | Single Machine | Static | Stochastic (Machine breakdown) | GA Pairwise swapping methods | Minimise deviation of start times or of sequences (stability) Minimise makespan | Continuous (Every Machine Breakdown) | Off-line | Reschedule (same method) |
| Leon *et al.* (1994) | Job Shop | Static | Stochastic (Machine breakdown, processing time variability) | GA | Minimise expected makespan and expected deviation from original makespan using surrogate measures (robustness) | Continuous (Every Machine Breakdown) | Off-line | Right Shift |
| Daniels *et al.* (1995) | Single Machine | Static | Stochastic (Processing time variability) | B&B, other heuristics | Minimise absolute worst case total flowtime difference (robustness) | Periodic | Off-line | Do nothing (left or right shift) |
| Daniels and Carillo (1997) | Single Machine | Static | Stochastic (Processing time variability) | B&B, another heuristic | Maximise likelihood that realised total flowtime is no worse than a target level (robustness) | Periodic | Off-line | Do nothing (left or right shift) |
| Mehta and Uzsoy (1998) | Job Shop | Static | Stochastic (machine breakdown) | OSMH/LP | Minimise deviations of completion times while keeping $L_{MAX}$ low using surrogate measures (stability) | Continuous (Every Machine Breakdown) | Off-line | Right Shift |
| Mehta and Uzsoy (1999) | Single Machine | Static with nonzero ready times | Stochastic (Machine breakdown) | OSMH/LP | Minimise deviations of completion times while keeping $L_{MAX}$ low using surrogate measures (stability) | Continuous (Every Machine Breakdown) | Off-line | Right Shift |
| O'Donovan *et al.* (1999) | Single Machine | Static with nonzero ready times | Stochastic (Machine breakdown, processing time variability) | OSMH and ATC derivatives in combination | Minimise deviations of completion times while keeping total tardiness low (stability) | Continuous (Every Machine Breakdown) | Off-line | Right Shift ATC derivatives |
| Guo and Nonaka (1999) | Flow Shop (3 Machines) | Static | Stochastic (Single machine breakdown in the middle machine) | Analytic analysis | Minimise expected deviations of completion times (stability) | Continuous (Every Machine Breakdown) | Off-line | Reschedule (implement schedule offered by analytic analysis) |
| Wu *et al.* (1999) | Job Shop | Static | Stochastic (Processing time variability) | B&B/ATC | Minimise expected weighted total tardiness using surrogate measures (robustness) | Periodic | Quasi on-line | Right-Shift |

(*continued*)

Table 4. (*Continued*).

| | Environment | | | Schedule Generation | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | How to | |
| Reference | Shop Floor | Sta./Dyn. | Stoch./Det. | Method | Objective | When to | Scheme | Response |
| Yang and Yu (2002) | Single Machine | Static | Stochastic (Processing time variability) | Dynamic Programming, Heuristics with greedy approach and surrogate measures | Minimise absolute, relative and percentage worst case total flowtime difference (robustness) | Periodic | Off-line | Do nothing (left or right shift) |
| Jensen (2001) | Job Shop | Static with nonzero ready times | Stochastic (Machine Breakdowns) | GA | Optimise neighborhood-based robustness (makespan) | Continuous (Every Machine Breakdown) | Off-line | Reschedule (same method) |
| Jensen (2003) | Job Shop | Static with nonzero ready times | Stochastic (Machine Breakdowns) | GA | Optimise neighborhood-based robustness (tardiness and flowtime) | Continuous (Every Machine Breakdown) | Off-line | Reschedule (same method) |
| Al-Fawzan and Haouari (2004) | Resource-constrained Project Scheduling | Static | Stochastic (Processing time variability) | Tabu Search | Minimise makespan Maximise total free slack (robustness) | Periodic | Off-line | Do nothing (left or right shift) |
| Artigues *et al.* (2005) | Job Shop | Static | Deterministic (Multiple schedules to be implemented in case of disruptions) | Heuristic similar to Shifting Bottleneck Algorithm | Maximise flexibility (Minimise number of ordered group assignments and Maximise number of offered schedules) | Periodic | Quasi on-line | Do nothing (left or right shift) |
| Kasperski (2005) | Single Machine | Static | Stochastic (Processing time variability, due-date uncertainty) | Iteratively use Lawler's Algorithm on a specially constructed worst-case scenario | Minimise absolute worst case maximum lateness difference (robustness) | Periodic | Off-line | Do nothing (left or right shift) |
| Sevaux and Sörensen (2004) | Single Machine | Static | Stochastic (release dates) | GA | Minimise expected realised performance and the standard deviation of the realised performance (robustness) | Periodic | Off-line | Do nothing (left or right shift) |
| Van de Vonder *et al.* (2008) | Resource-constrained Project Scheduling | Static | Stochastic (Processing time variability) | Heuristics such as VADE, SCT, Tabu Search | Minimise deviations of completion times (stability) | Periodic | Off-line | Do nothing (left or right shift) |

schedule at time t = 0, the environment is *static*. On the other hand, if jobs continue to arrive dynamically throughout the scheduling horizon, the environment is said to be *dynamic*. 'Stoch./Det.' column shows the uncertainty type present in the scheduling environment: if all job and machine parameters are constant and known in advance the environment is *deterministic*. Otherwise the environment is said to be *stochastic* and type of the stochasticity is also noted. Other columns of the table are self-explanatory. For a more general review of scheduling under uncertainty including reactive policies, the interested reader can see Sabuncuoglu and Bayiz (2000), Vieira *et al*. (2003) and Aytug *et al*. (2005).

The studies that are reviewed here can be broadly divided into two categories: those that model the uncertainty by probability density functions and define robustness or stability measures in terms of these functions (probabilistic approach); and those employing a scenario analysis similar to the one in Section 4 (scenario planning approach). In fact, these two approaches are analogous in the sense that the probabilistic approach can be considered as a scenario planning approach with a continuum of infinitely many scenarios and the probability of each scenario is governed by the probability density functions used in the probabilistic approach. Hence, the robustness or stability measures used in both approaches are analogous.

### 5.1. Probabilistic approach

#### 5.1.1. Robustness studies

Leon *et al*. (1994) study robustness in a job-shop environment. Their aim is to construct a robust initial schedule. Given the right-shift response policy, an *a priori* off-line schedule is developed to achieve high system performance in the presence of machine breakdowns. In their model, the disruptions are machine failures. The times to failure and repair distributions are assumed to be known. The shop-floor performance measure is taken as makespan. The robustness measure for a given schedule is represented as a convex combination of expected makespan of the realised schedule and expected deviation from the original deterministic makespan (a convex combination of first and fifth measures in Section 4.2.1.1). The authors propose several surrogate measures for this robustness measure. In a correlation study, they determine the best one and use a genetic algorithm to minimise this surrogate measure. They test the performance of the proposed algorithm under random machine breakdown and processing time variability. The results indicate that the proposed algorithm outperforms the classical algorithms that focus on minimising makespan only. As for processing time variability, the difference is only significant when variability is high enough.

A similar study in the context of resource-constrained project scheduling is conducted by Al-Fawzan and Haouari (2004). The activity durations are subject to uncertainty in their problem. The authors develop a bicriteria model which aims at generating robust schedules with minimum makespan. Schedule robustness is defined as the sum of *free slacks* of activities. Free slack of an activity is defined as the maximum amount of time the activity can slip without delaying the start of the next activity while maintaining resource feasibility. The authors propose a tabu-search algorithm to approximate the set of efficient (non-dominated) solutions. Schedules are represented as ordered list of activities. At each iteration of the proposed algorithm, the neighbourhood of a current solution is generated by random feasible pairwise swappings of the adjacent activities. The solution to be considered in the next iteration is the schedule with the minimum ($\lambda \times$ makespan $-$ (1-$\lambda$) $\times$ total free slack) value, where $0 \leq \lambda \leq 1$. This single objective TS algorithm is run several times, each time with an increased value of $\lambda$. During each run of the TS, a set of non-dominated schedules is also maintained. This set is updated by appending current neighbourhood and removing all dominated solutions at each iteration. The final approximate set is generated by removing the dominated schedules from the union of the sets obtained at the end of each run. The authors' computational results indicate that the proposed bicritaria TS algorithm in most cases generates a moderately sized set of efficient solutions. Also its performance on the single objective makespan minimisation is observed to be comparable to special purpose (that solely aim at minimising makespan) tabu search algorithms.

In another study, Wu *et al*. (1999) propose a graph-theoretic decomposition to the job shop scheduling problem to achieve schedule robustness. In this study, the authors combine good properties of off-line and on-line scheduling and propose a quasi-online method for the job-shop scheduling problem. Their robustness measure is expected average weighted tardiness. They use a graph representation of this problem, in which conjunctive arcs represent precedence constraints and disjunctive arcs join the operations competing for the same resource. They propose a branch and bound algorithm that processes disjunctive arcs and change some of them into conjunctive arcs. This effectively makes some of the scheduling decisions. The remaining scheduling decisions are made dynamically by applying the ATC heuristic. In their paper, this approach is

called a *process first schedule later (PFSL)* scheme. They compare their approach with the off-line algorithm (IATC) and on-line algorithm (ATC). Their computational experiments show that the *PFSL* scheme yields more robust performance under a wide range of disturbances (various levels of processing time variability) than traditional off-line and on-line methods.

In a similar study, Artigues *et al.* (2005), propose to generate a family of schedules instead of a unique one to maintain schedule robustness in a job shop environment. The families of schedules are represented by ordered group assignments that define a sequence of groups for each machine such that the operations within every group are totally permutable. This approach introduces *flexibility* in the sense that the decision maker can easily switch from one solution to another one in the family in case of disruptions. The authors define two criteria to maximise the flexibility of ordered group assignments: minimising the number of groups and maximising the number of schedules represented by the groups. They first study in a single machine setting with due-dates. A polynomial time algorithm is proposed to minimise the number of groups if all jobs are ready at time $t = 0$. They also develop a polynomial time dynamic programming algorithm to maximise the number of schedules if release dates of jobs are not all zero and a given sequence has to be represented by the ordered group assignment. The authors use this single machine dynamic programming algorithm as a basis for a heuristic that generates ordered group assignments for the job shop problem in the spirit of the Shifting Bottleneck Heuristic. Their computational results indicate that the proposed algorithm increases the applicability of the given schedule by introducing significant flexibility while keeping the makespan of the represented schedules by the ordered group assignment acceptable.

Jensen (2003) generates robust schedules in a shop job environment subject to random machine breakdowns where the performance measure is taken as makespan. The author defines two *neighbourhood-based* robustness measures. The first measure is the average makespan of the given schedule's neighbourhood, where neighbourhood is defined as the set of all the schedules that can be obtained by a pairwise swapping of two consecutive operations on the same machine. The second robustness measure is an estimate (upper bound) of the first one. The author compared the performances of these two robustness measures to the 'minimise makespan' and 'maximise slack' criteria. All four measures are implemented into specific genetic algorithms and their performances (expected realised makespan after a simulated single breakdown) are compared under five different rescheduling methods. The rescheduling method varies from the right-shifting to rescheduling the whole system from scratch. The computational results indicate that it is possible to find a schedule having both a low makespan and a low robustness measure. He also observed that for a moderate breakdown time, slack-based robustness measure works the best for simple rescheduling methods (e.g. right-shift) whereas neighbourhood-based robustness measures ourperform the slack-based measure for more sophisticated rescheduling methods (e.g. reschedule from scratch). For very large breakdown times, the author's computational results indicate that the slack-based robustness measure is superior in most cases.

Jensen (2001) analyses the performance of the neighbourhood-based robustness measures in a job shop environment subject to machine breakdowns where performance measures are taken as maximum tardiness, total tardiness or total flowtime. For tardiness problems, the author also proposes to minimise a measure of lateness instead of tardiness to increase the slack in the schedule, which may improve the rescheduling performance. In other words, corresponding lateness measures are taken as a simple robustness measure for tardiness problems. The author compares the performance of the neighbourhood-based robustness measures to the simple robustness measure and to ordinary tardiness-minimising criterion. His computational experiments indicate that lateness-based robustness measures improve schedule robustness for problems with loose due-dates while they are equivalent to ordinary scheduling on tight problems. It is also observed that neighbourhood-based robustness measures improve schedule robustness when used with simple rescheduling techniques for all cases and their improvement found to be better than improvement gained from lateness-based robustness measure in most cases. On the other hand, with more sophisticated rescheduling methods, neighbourhood-based robustness measures do not improve schedule robustness for tight total tardiness and total flowtime problems.

Sevaux and Sörensen (2004) study robustness in a single machine environment with stochastic release dates. The performance measure is the total weighted number of tardy jobs. The authors develop a genetic algorithm where the expected realised performance and the standard deviation of the realised performance (first and sixth measures of Section 4.2.1.1) are employed as fitness functions to generate robust schedules. Their computational experiments show that the proposed robust fitness functions improve schedule robustness as compared to using the deterministic fitness function (total number of tardy jobs).

#### 5.1.2. Stability studies

One of the earlier studies in this area is by Wu *et al.* (1993) who consider single-machine rescheduling problem with machine disruptions. They reschedule the jobs in response to each machine failure so that minimum makespan is achieved with high schedule stability. Note that these two goals often conflict; minimising makespan usually requires some changes in the schedule, but high schedule stability can be achieved by minimising the number of changes. Measuring schedule stability is a difficult task unlike a regular system-performance measure such as makespan. The authors consider two criteria for stability. The first is the deviation of revised schedule with regard to job starting times. This criterion is useful if the secondary resources such as tools and fixtures are delivered according to the original schedule. The second criterion is the deviation of the revised schedule from the original schedule in terms of job sequence. This criterion is useful if there are sequence-dependent tooling, fixtures, set-ups, etc. The measure for the first criterion is the average absolute deviation from the original job starting times; these can be easily calculated (first measure in Section 4.2.2). However, the second criterion is difficult to measure. One possible surrogate measure for the second criterion is the sum of absolute deviations of job starting times from the *right-shift* schedule. Recall that the right-shift schedule maintains the original sequence and represents minimal disruption to the original schedule when job sequence is important. The authors have two problem types that differ with respect to the stability criterion in use. Since the problem is NP-hard, even without stability considerations, they use heuristics to solve it. The first heuristic is based on pairwise swapping. The second type utilises genetic algorithms. All heuristics construct a list of non-dominated schedules. To test the performances of these algorithms, they conduct two sets of experiments. In the first set, they generate small problems and compare their solutions with the optimal solution obtained by a mixed integer-programming model. In the second set, they measure the effectiveness of their heuristics with large problems. In the first set of experiments, all heuristics are able to find optimal solutions. In the second set of experiments, they confirm that stability of the schedules could be improved significantly with little sacrifice in makespan. No significant difference is observed between algorithms but the computational burden of the pairwise-swapping heuristic is more than that of the adjacent pairwise swapping heuristic; the genetic algorithm is in between.

In another study, Mehta and Uzsoy (1998, 1999) generate initial *stable* schedules under conditions of random machine breakdowns. They call their initial schedules as *predictable* schedules. Like Wu *et al.* (1993), Mehta and Uzsoy argue that a good predictable schedule should have not only a good shop floor performance, but should also take *predictability* (i.e. stability) into account. That is, deviations from the original schedule should be minimal because many other decisions (such as purchasing, tooling, etc.) on the shop floor are planned according to this initial schedule. Their term 'predictable scheduling' is an alternative approach to what is known as scheduling/rescheduling in the classical scheduling literature. The objective of this approach is to generate an initial schedule such that deviation from the initial schedule during execution is minimised while keeping shop floor performance degradation to an acceptable level.

In their first paper, Mehta and Uzsoy (1999) study the single machine scheduling problem where jobs have nonzero ready times and there are random machine breakdowns. The time-to-failure and repair duration distributions are assumed to be known *a priori*. In the second paper (1998), they study the job-shop scheduling problem with random machine breakdowns. In both studies, they use maximum lateness as the shop floor performance measure. Unlike Wu *et al.* (1993), they try to minimise deviations while generating an initial schedule, not when rescheduling after a breakdown. Deviation from the initial schedule is measured in terms of expected absolute deviation of job-completion times (the first measure in Section 4.2.2). The authors offer a two stage approach. In the first stage a job sequence is determined that minimises the maximum lateness. They use the algorithm and shifting bottleneck algorithm of Carlier (1982) for the single machine case and the job-shop case, respectively. In the second stage, they insert some idle time in this sequence. The amounts of idle time are large enough to provide protection against machine breakdowns but they are small enough so that maximum lateness does not greatly increase. Their computation results indicate that stability can be easily improved while slightly increasing maximum lateness.

O'Donovan *et al.* (1999) conjoin reactive and proactive approaches and examine the scheduling/rescheduling policy using stability and efficiency measures in a single-machine environment. Schedule efficiency is measured by total tardiness; stability is measured by absolute completion-time deviations from the initial schedule (the first measure in Section 4.2.2). The system under study has nonzero job-ready times and random machine breakdowns. This study is similar to the one by Mehta and Uzsoy (1999) except that total tardiness instead of maximum lateness is used as the system performance measure. They use pure ATC (Morton and Rachamadugu 1982) and ATC

with inserted idle times for initial schedule generation. Rescheduling alternatives are ATC, a modified ATC (which calculates the slack of a job based on its predicted completion time, taking inserted idle times into account) and right-shift scheduling. Experimental results indicate that ATC with inserted idle times for scheduling and modified ATC for rescheduling are the best for stability. The authors also consider *sensitive jobs*. They argue that in some production environments, jobs are sensitive to disturbances that have just occurred and the degree of sensitivity differs from job to job (the impact of a disruption is not felt equally by all jobs). In this model, they propose ATC with inserted idle times for scheduling and Smart ATC for rescheduling. Smart ATC is similar to modified ATC, but it uses estimated affected processing time instead of processing time for a job.

Van de Vonder *et al*. (2008) generate stable initial schedules for resource-constrained project scheduling problem under activity duration variability. The used stability measure is the first measure in Section 4.2.2. Like Mehta and Uzsoy (1998), the authors follow a two-stage approach: in the first stage, an initial schedule with the objective of minimising makespan is generated using a deterministic algorithm, where activity durations are set to their mean values. In the second stage, additional idle times are inserted at the start of each activity to protect the schedule from the negative effects of activity duration variation. The amount of inserted idle times are determined using several heuristics. VADE heuristic (Virtual Activity Duration Extension) iteratively uses standard deviations of activity durations to insert 1 unit of idleness in front of selected activities. At each iteration, the resulting schedule is simulated to evaluate its stability value and is also taken as the input for the next iteration. The schedule with the minimum stability measure will be the output of VADE procedure. STC (Starting Time Criticality) heuristic works in the same iterative manner, but activity to insert additional idle time is selected considering both weights and variances of the activities. RFDFF (Resource Flow Dependent Float Factor) is used as a benchmark. This heuristic inserts an additional idle time that is equal to a multiple (the coefficient is based on weight of the selected activity and weights of its predecessors) of float of the selected activity. The authors also propose a iterative improvement heuristic which takes a schedule generated by the aforementioned heuristics as input. A neighbourhood of the current solution is constructed by sliding a selected activity to all possible discrete starting times within a time window that would not affect other activities. The best starting time is selected by evaluating all the neighbourhood via simulation and this schedule is taken as input for the next iteration. Finally a tabu search is also proposed. The authors' computational experiments show that STC is generally the best heuristic if no improvement phase is considered. They also observe that if all project activities are subject to significant uncertainty, tabu search yields the best result but is computationally expensive. STC (or VADE) with an improvement phase results in an almost equally performance.

Guo and Nonaka (1999) study stability in a 3-machine flow shop where the performance measure is makespan and the middle machine is subject to a single machine breakdown. Their stability measure (called *robust function* by the authors) is the first measure in Section 4.2.2. The authors develop an algorithm to optimally reschedule the system by considering different intervals on time of the machine breakdown ($t$) and downtime duration ($D$). Hence, different schedules would be obtained for different $D$ and $t$ values after rescheduling. The authors propose to first generate several optimal schedules without considering the machine breakdown. Each optimal schedule is then analysed according to the proposed algorithm for different $D$ and $t$ values to calculate its expected stability value. The schedule with the minimum stability measure is then selected to set out and it is rescheduled according to the proposed algorithm when the breakdown occurs during its implementation.

### 5.2. *Scenario planning approach*

Daniels and Kouvelis (1995) generate initial robust schedules to hedge against processing time variability in a single machine environment where the performance measure is total flowtime. The authors propose a scenario-based representation and analysis of uncertainty rather than using stochastic models. They use the second measure and policy outlined in Section 4.2.1.2 for robustness. Recall that this policy finds the schedule with the least performance degradation in its worst case scenario. They formulate mathematical programs called the *absolute deviation robust scheduling problem* ADRSP and the *relative deviation robust scheduling problem* RDRSP. Their solutions give robust schedules, when the number of scenarios is finite. The ADRSP and RDRSP differ from one another only in their definition of robustness. The first one measures the deviations from optimal absolutely, whereas the other measures it relatively (similar to percentage deviation). They prove that this problem is NP hard. In practice, the number of scenarios is infinite and the problem is expected to be more difficult because possible processing-time values for jobs are given as ranges (for example the processing time of job 1, $P_1 \in [a, b]$). However Daniels and Kouvelis (1995) prove that a properly selected finite set of scenarios is

enough to determine the worst-case absolute deviation of a given sequence and they construct a procedure that does the worst case evaluation in polynomial time. They develop a branch and bound algorithm and two $O(n \log n)$ surrogate relaxation heuristics that utilise this procedure to generate a robust sequence. The computational burden of the branch and bound algorithm, which solves the problem optimally, grows rapidly with problem size. Still the number of sequences evaluated is small as compared to total number of sequences, $n!$, where $n$ is the number of jobs. The heuristics require far less computational time because they evaluate far fewer permutations of jobs to determine approximate robust sequences; their computational burden grows only modestly as problem size increases. Moreover, they closely approximate the optimal absolute deviations. The authors compare their solutions to SEPT (shortest expected processing time) solutions, which are used in practice to generate the optimal sequence of jobs. They observe that SEPT performs poorly in terms of robustness.

The study of Daniels and Kouvelis (1995) hedges against the worst contingency that may arise without considering any specific probability distribution. The approach is known as the *robustness approach* in the literature. The reader is referred to Kouvelis and Yu (1996), who apply this method to various problems such as linear programming, assignment problem, shortest path problem, etc. as well as scheduling.

Yang and Yu (2002) study the same problem as Daniels and Kouvelis (1995) - minimise total flow time in a single machine environment with processing time variability- with discrete, finite set of processing time scenarios rather than interval data. The authors consider three robustness measures: 1) minimise worst-case scenario cost; 2) minimise worst-case scenario absolute regret; and 3) minimise worst-case scenario percentage regret. The problem is proved to be NP-complete for all three measures even in the case of two scenarios. The authors show that all three measures can be solved by a single method. An exact $O(2^n)$ dynamic programming algorithm as well as two polynomial time heuristics (a greedy heuristic and a heuristic which uses average total flowtime over all scenarios as a surrogate robustness measure) are presented. Their computational experiments indicate that the greedy heuristic outperforms the surrogate heuristic, the difference becoming more and more obvious as the number of scenarios increase.

Kasperski (2005) studies robustness in a single machine environment with precedence constraints with maximum lateness criterion. Job processing times and due-dates are uncertain and each of them involves a range of possible realisations. Like Daniels and Kouvelis (1995), the author takes a robustness approach aiming to minimise worst-cast scenario performance (minimax regret). Note that this robustness measure is the second measure in Section 4.2.1.2. The author develops a polynomial time algorithm, which repeatedly uses Lawler's algorithm on specially constructed worst-case scenario candidates.

In another study Daniels and Carillo (1997) generate robust schedules for a single machine environment with processing time uncertainty where the performance measure is total flowtime. They assume that the processing time of an individual job can be specified only imprecisely and this uncertainty is captured within a set of processing time scenarios, where each scenario can be realised with some positive probability. Alternative job sequences are evaluated according to the likelihood that actual system performance will be no worse than a given target level $T$. The scheduling objective is to determine the sequence (which is called *$\beta$-robust sequence*) that maximises this likelihood. The authors prove that this problem is NP-hard and formulate the problem as an integer program (IP). They develop a branch-and-bound algorithm to solve this IP. They also develop a heuristic where only a few specially generated processing time scenarios are evaluated. Their computational results indicate that $\beta$-robust schedules perform very well with respect to the expected total flowtime measure in addition to maximising the likelihood of achieving a flowtime performance no worse than a given target level. The computational results also show that the proposed heuristic performs well.

## 6. Concluding remarks and future research directions

In this paper, proactive scheduling problem is analysed and several robustness and stability measures are presented. The scheduling decisions are classified in terms of when-to-schedule and how-to-schedule and the required response types and methods to cope with these disruptions are identified. The existing studies are also reviewed. One can make the following observations and the related further research directions in this area:

(1) An analysis of off-line scheduling and on-line scheduling methods is needed in a dynamic and stochastic environment that includes the consideration of robustness and stability measures.
(2) The relative performance of full scheduling and partial scheduling in a dynamic and stochastic environment is needed for both robustness and stability measures.
(3) Even though there are some studies that analyse rescheduling frequency, the relative weaknesses and strengths of different types of

response (do nothing, reschedule or repair) are not thoroughly known.

(4) Even though the scheduling decisions are analysed to some extent by Sabuncuoglu and Kizilisik (2003), when-to-schedule policies (periodic, continuous, or adaptive) also need further research. Specifically, it would be interesting to know the conditions under which a particular policy is better than others.

(5) Should practitioners consider robustness, stability, or both? Are robustness and stability conflicting objectives? What is the trade-off between these performance metrics (stability, robustness and efficiency)?

(6) Can one develop a bicriteria approach that considers both the stability and robustness measures simultaneously?

(7) As Mehta and Uzsoy (1998) have stated, one needs to develop better surrogate measures for stability (and/or robustness).

(8) As also acknowledged by Leon *et al*. (1994), the existing studies should be extended for other performance measures in addition to makespan.

## References


Abumaizar, R.J. and Svestka, J.A., 1997. Rescheduling job shops under disruptions. *International Journal of Production Research*, 35, 2065–2082.

Akturk, M.S. and Gorgulu, E., 1999. Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, 112, 81–97.

Al-Fawzan, M.A. and Haouari, M., 2005. A bi-objective modal for robust resource constrained project scheduling. *International Journal of Production Economics*, 96, 175–187.

Artigues, C., Billaut, J.C., and Esswein, C., 2005. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165, 314–328.

Aytug, H., *et al*., 2005. Executing production schedules in the face of uncertainties: a review and some future directions. *European Journal of Operational Research*, 161 (1), 86–110.

Bean, J.C., *et al*., 1991. Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39 (3), 470–483.

Carlier, J., 1982. The one-machine sequencing problem. *European Journal of Operational Research*, 11, 42–47.

Church, L.K. and Uzsoy, R., 1992. Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, 5, 153–163.

Daniels, R.L. and Carillo, J.E., 1997. $\beta$-robust scheduling for single machine systems with uncertain processing times. *IIE Transactions*, 29, 977–985.

Daniels, R.L. and Kouvelis, P., 1995. Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41 (2), 363–376.

Davenport, A.J. and Beck, J.C., 2000. A survey of techniques for scheduling with uncertainty [online]. Available from: http://eil.utoronto.ca/profiles/chris/chris.papers.html [Accessed 18 April 2008].

Guo, B. and Nonaka, Y., 1999. Rescheduling and optimization of schedules considering machine failures. *International Journal of Production Economics*, 60–61, 503–513.

Herroelen, W. and Leus, E., 2005. Project scheduling under uncertainty: survey and research potentials. *European Journal of Operational Research*, 165 (2), 289–306.

Jensen, M.T., 2001. Improving robustness and flexibility of tardiness and total flow time job shops using robustness measures. *Applied Soft Computing*, 1, 35–52.

Jensen, M.T., 2003. Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 7 (3), 275–288.

Kasperski, A., 2005. Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion. *Operations Research Letters*, 33, 431–436.

Kouvelis, P. and Yu, G., 1997. *Robust discrete optimization and its applications*. Dordrecht: Kluwer Academic.

Kutanoglu, E. and Sabuncuoglu, I., 2001. Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop. *Journal of Manufacturing Systems*, 20 (4), 264–279.

Leon, V.J., Wu, S.D., and Storer, R.H., 1994. Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26 (5), 32–43.

Leus, R. and Herroelen, W., 2005. The complexity of machine scheduling for stability with a single disrupted job. *Operations Research Letters*, 33, 151–156.

Mehta, S.V. and Uzsoy, R., 1998. Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, 14 (3), 365–378.

Mehta, S.V. and Uzsoy, R., 1999. Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing*, 12 (1), 15–38.

Morton, T.E. and Rachamadugu, R.V., 1982. Myopic heuristics for the single machine weighted tardiness problem. Tech. Report CMU-RI-TR-83-09, Robotics Institute, Carnegie Mellon University.

O'Donovan, R., Uzsoy, R., and McKay, K.N., 1999. Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 37 (18), 4217–4233.

Raman, N., Rachamadugu, R.V., and Talbot, B., 1989. Real-time scheduling of an automated manufacturing center. *European Journal of Operations Research*, 40, 222–242.

Sabuncuoglu, I. and Bayiz, M., 2000. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126, 567–586.

Sabuncuoglu, I. and Hommertzheim, D.L., 1992. Dynamic dispatching algorithm for scheduling machines and AGVs in a flexible manufacturing system. *International Journal of Production Research*, 30, 1059–1080.

Sabuncuoglu, I. and Karabuk, S., 1999. Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, 18 (4), 1–16.

Sabuncuoglu, I. and Kizilisik, O.B., 2003. Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research*, 41 (17), 4211–4231.

Sevaux, M. and Sörensen, K., 2004. A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2, 129–147.

Sotskov, Y., Sotskova, N.Y., and Werner, F., 1997. Stability of an optimal schedule in a job shop. *Omega: the International Journal of Management Science*, 25 (4), 397–414.

Van de Vonder, S., Demeulemeester, E., and Herroelen, W., 2008. Proactive heuristic procedures for robust project scheduling: an experimental analysis. *European Journal of Operational Research*, 189 (3), 723–733 (in press).

Vieira, G.E., Herrmann, J.W., and Lin, E., 2003. Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling*, 6, 39–62.

Wu, S.D., Byeon, E., and Storer, R.H., 1999. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47 (1), 113–124.

Wu, S.D., Storer, R.N., and Chang, P., 1993. One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research*, 20 (1), 1–14.

Yamamoto, M. and Nof, S.Y., 1985. Scheduling/rescheduling in a manufacturing operating system environment. *International Journal of Production Research*, 23 (4), 705–722.

Yang, J. and Yu, G., 2002. On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6, 17–33.