

# A survey of dynamic scheduling in manufacturing systems

Djamila Ouelhadj · Sanja Petrovic

Published online: 28 October 2008  
© Springer Science+Business Media, LLC 2008

**Abstract** In most real-world environments, scheduling is an ongoing reactive process where the presence of a variety of unexpected disruptions is usually inevitable, and continually forces reconsideration and revision of pre-established schedules. Many of the approaches developed to solve the problem of static scheduling are often impractical in real-world environments, and the near-optimal schedules with respect to the estimated data may become obsolete when they are released to the shop floor. This paper outlines the limitations of the static approaches to scheduling in the presence of real-time information and presents a number of issues that have come up in recent years on dynamic scheduling.

The paper defines the problem of dynamic scheduling and provides a review of the state-of-the-art of currently developing research on dynamic scheduling. The principles of several dynamic scheduling techniques, namely, heuristics, meta-heuristics, multi-agent systems, and other artificial intelligence techniques are described in detail, followed by a discussion and comparison of their potential.

**Keywords** Dynamic scheduling · Robust scheduling · Predictive–reactive scheduling · Agent-based scheduling

---

D. Ouelhadj (✉) · S. Petrovic  
Automated Scheduling, Optimisation and Planning Research  
Group, School of Computer Science, University of Nottingham,  
Nottingham NG8 1BB, UK  
e-mail: [dxs@cs.nott.ac.uk](mailto:dxs@cs.nott.ac.uk)

S. Petrovic  
e-mail: [sxp@cs.nott.ac.uk](mailto:sxp@cs.nott.ac.uk)

## 1 Introduction

Most manufacturing systems operate in dynamic environments where usually inevitable unpredictable real-time events may cause a change in the scheduled plans, and a previously feasible schedule may turn infeasible when it is released to the shop floor. Examples of such real-time events include machine failures, arrival of urgent jobs, due date changes, etc. MacCarthy and Liu (1993) addressed the nature of the gap between the scheduling theory and scheduling practice, the failure of classical scheduling theory to respond to the needs of practical environments, and recent trends in scheduling research which attempt to make it more relevant and applicable. Shukla and Chen (1996), in their comprehensive survey on intelligent real-time control in flexible manufacturing systems, stated that comparison of theory and scheduling practice showed very little correspondence between the two. Cowling and Johansson (2002) addressed an important gap between scheduling theory and practice, and stated that scheduling models and algorithms are unable to make use of real-time information.

The problem of scheduling in the presence of real-time events, termed *dynamic scheduling*, is of great importance for the successful implementation of real-world scheduling systems. However, very few surveys have been published in this area. In this paper, we focus on a number of issues that have come up in recent years on dynamic scheduling in manufacturing systems. We are primarily concerned with the issue of how to handle the occurrence of real-time events during the execution of a given schedule on the shop floor.

The paper is organised as follows. Section 2 defines the problem of dynamic scheduling and the categories of real-time events. Next, Section 3 discusses the rescheduling policies and rescheduling strategies. Section 4 gives a review on previous research work on techniques used to solve the

problem of dynamic scheduling. Section 5 presents a comparative study of the different techniques. Finally, summary and conclusions are presented in Section 6.

## 2 The dynamic scheduling problem

Literature on dynamic scheduling has considered a significant number of real-time events and their effects considering various manufacturing systems, including single machine systems, parallel machine systems, flow shops, job shops, and flexible manufacturing systems.

Real-time events have been classified into two categories (Stoop and Weirs 1996; Suresh and Chaudhuri 1993; Cowling and Johansson 2002; Vieira et al. 2003):

- **Resource-related:** machine breakdown, operator illness, unavailability or tool failures, loading limits, delay in the arrival or shortage of materials, defective material (material with wrong specification), etc.
- **Job-related:** rush jobs, job cancellation, due date changes, early or late arrival of jobs, change in job priority, changes in job processing time, etc.

Dynamic scheduling has been defined under three categories (Mehta and Uzsoy 1999; Vieira et al. 2000a, 2003; Aytug et al. 2005; Herroelen and Leus 2005): completely reactive scheduling, predictive–reactive scheduling, and robust pro-active scheduling.

### 2.1 Completely reactive scheduling

In completely reactive scheduling no firm schedule is generated in advance and decisions are made locally in real-time. Priority dispatching rules are frequently used. A dispatching rule is used to select the next job with highest priority to be processed from a set of jobs awaiting service at a machine that becomes free. The priority of a job is determined based on job and machine attributes. Dispatching rules are quick, usually intuitive, and easy to implement. However, global scheduling has the potential to significantly improve shop performance compared to myopic dispatching rules, where it is hard to predict system performance as decisions are made locally in real-time.

### 2.2 Predictive–reactive scheduling

Predictive–reactive scheduling is the most common dynamic scheduling approach used in manufacturing systems. Most of the definitions reported in the literature on dynamic scheduling refer to predictive–reactive scheduling. Predictive–reactive scheduling is a scheduling/rescheduling process in which schedules are revised in response to real-time events.

Most of the predictive–reactive scheduling strategies are based on simple schedule adjustments which consider only shop efficiency. The new schedule may deviate significantly from the original schedule, which can seriously affect other planning activities that are based on the original schedule and may lead to poor performance of the schedule. It is therefore desirable to generate predictive–reactive schedules that are robust. Robust predictive–reactive scheduling focuses on building predictive–reactive schedules to minimise the effects of disruption on the performance measure value of the realised schedule (Wu et al. 1991, 1993; Leon et al. 1994). A typical solution to generate a robust schedule is to reschedule considering both shop efficiency and deviation from the original schedule (stability) simultaneously. Stability measures the deviation from the original predictive schedule caused by schedule revision to quantify the undesirability of making changes to the initial schedule (Wu et al. 1991, 1993; Cowling and Johansson 2002; Leus and Herroelen 2005). Wu et al. (1991, 1993) defined a bi-criterion robustness measure for one-machine rescheduling problem with machine breakdown. The criteria include the minimisation of the makespan (schedule efficiency) and the impact of schedule change (schedule stability). For the stability, they investigated two measures: the deviation from the original job starting time, and the deviation from the original sequence. The experimental results showed the effectiveness of the robustness measure due to the fact that the schedule stability can be increased significantly with little or no reduction in makespan. In the same order of idea, Abu-maizar and Svestka (1997) used two measures to define a robust schedule: efficiency (makespan) and stability (starting time deviation and sequence deviations). The scheduling objective is to maximise shop efficiency, and at the same time minimise system impact caused by schedule changes. Jensen (2001) investigated different robustness measures to improve tardiness and total flow-time for machine breakdowns. Leon et al. (1994) developed robustness measures and robust scheduling to deal with machine breakdowns and processing time variability when a right-shift repair strategy is used. The robustness is defined as the minimisation of the bi-criterion objective function expressed in terms of both expected makespan and expected delay. The expected delay is the deviation between the deterministic makespan before disruption, and the actual makespan after applying right shifting. The experimental results showed that robust schedules significantly outperform schedules based on makespan alone. Daniels and Kouvelis (1995) defined robustness measures for a single machine environment to cope against processing time uncertainty where the scheduling objective is to minimise the flow time of jobs. The robustness is defined as the minimisation of both the flow time and the absolute deviation from the original schedule caused by schedule revision. Extensive computational results reported the ef-

efficiency and effectiveness of the proposed robustness measures. Cowling and Johansson (2002), and Ouelhadj et al. (2003b) defined general measures of utility and stability to guide the decision as what strategy should be used to react to real-time events in order to define a robust schedule. Utility measures the change in the value of the schedule objective function following a schedule revision. It is expressed by the difference between the value of the objective function of the new schedule after reacting to the real-time events and the objective function of the predictive schedule before taking into account real-time events. They have then investigated a number of utility and stability measures for single machine scheduling model with the objective of minimising the average completion time.

### 2.3 Robust pro-active scheduling

Robust pro-active scheduling approaches focus on building predictive schedules which satisfy performance requirements predictably in a dynamic environment (Mehta and Uzsoy 1999; Vieira et al. 2003). The main difficulty of this approach is the determination of the predictability measures. Mehta and Uzsoy (1999) proposed a predictable scheduling model for a single machine subject to breakdowns with the objective to minimise the maximum lateness. The effect of disruption is measured by the deviation of the job completion time of the realised schedule from its planned completion time in the predictive schedule. The deviation is reduced by inserting additional time in the predictive schedule with the objective of achieving high predictability. Extensive computational experiments showed that predictable scheduling provides a significant improvement in predictability at the expense of very little degradation in the maximum lateness. O'Donovan et al. (1999) extended the predictable scheduling approach of Mehta and Uzsoy where the measure of schedule performance is the tardiness of jobs.

## 3 Rescheduling in the presence of real-time events

Rescheduling in the presence of real-time events needs to address two issues: how and when to react to real-time events. The first issue concerns the definition of rescheduling strategies to react to real-time events, and the second issue addresses the problem of when to reschedule.

### 3.1 Rescheduling strategies

Regarding the first issue, what strategies to use to reschedule, the literature provided two main rescheduling strategies (Sabuncuoglu and Bayiz 2000; Cowling and Johansson 2002; Vieira et al. 2003): schedule repair, and complete rescheduling.

Schedule repair refers to some local adjustment of the current schedule and may be preferable because of the potential saving in CPU times and the stability of the system is preserved.

Complete rescheduling regenerates a new schedule from scratch. Complete rescheduling might, in principle, be better in maintaining optimal solutions, but these solutions are rarely achievable in practice and require prohibitive computation time. Moreover, complete rescheduling can result in instability and lack of continuity in detailed plant schedules, leading to additional production costs attributable to what has been termed shop floor nervousness.

Sun and Xue (2001), and Dorn et al. (1995) reported that most of reactive scheduling systems attempt to revise only part of the originally created schedule for responding to the production environment changes without rescheduling from scratch. Abumaizar and Svestka (1997) stated that in practice rescheduling has been done by schedule repair, while complete rescheduling has been used also to a limited degree. Sabuncuoglu and Bayiz (2000) demonstrated the potential effectiveness of schedule repair in terms of stability and CPU time compared with complete rescheduling.

Another problem of practical importance is the decision whether to reschedule from scratch (complete rescheduling) or schedule repair, and which schedule repair strategy to choose to react to real-time events. To deal with this problem, simulation and robustness measures were used to evaluate the performance of the rescheduling strategies and to select the best strategy. Wu et al. (1991, 1993), Daniels and Kouvelis (1995), Abumaizar and Svestka (1997), and Jensen (2001) used robustness measures (efficiency and stability) to decide on the best rescheduling strategy to apply. Cowling and Johansson (2002), and Ouelhadj et al. (2003b) used utility and stability measures to assess the performance of various schedule repair and complete rescheduling strategies, and to select the best rescheduling strategy.

### 3.2 When to reschedule

Regarding the second issue, when to reschedule, three policies have been proposed in the literature (Sabuncuoglu and Bayiz 2000; Vieira et al. 2003): periodic, event driven, and hybrid. The *periodic* and *hybrid policies* have received special attention under the name *rolling time horizon* (Church and Uzsoy 1992; Ovacik and Uzsoy 1994; Sabuncuoglu and Karabuk 1999; Vieira et al. 2000a; Aytug et al. 2005).

In the *periodic policy*, schedules are generated at regular intervals, which gather all available information from the shop floor. The dynamic scheduling problem is decomposed into a series of static problems that can be solved by using classical scheduling algorithms. The schedule is then executed and not revised until the next period begins, where the planning horizon is renewed by taking into account new information gathered from the current shop floor

status. The periodic policy yields more schedule stability and less schedule nervousness. Unfortunately, following an established schedule in the face of significant changes in the shop floor status may compromise performance since unwanted products or intermediates may be produced. Determining the rescheduling period is also a difficult task.

The primary application of the rolling horizon approach to dynamic scheduling is due to Muhlemann et al. (1982). They investigated how the frequency of scheduling in a dynamic job shop environment affected the performance where the processing time variations and machine breakdowns occur randomly. At each rescheduling period, a static schedule is generated for current jobs using a dispatching rule. As anticipated, performance generally deteriorates when the rescheduling period increases. Ovacik and Uzsoy (1994) used the rolling horizon policy for a single machine problem with sequence-dependent set-up time to minimise maximum lateness. They found that rolling horizon scheduling outperforms myopic dispatching rules. Sabuncuoglu and Karabuk (1999) studied the rescheduling frequency in a multi-process flexible manufacturing system environment for machine breakdowns and processing time variations. The performance of the system is measured for mean tardiness and makespan criteria. The results on scheduling frequency indicated that a periodic response with an appropriate period length would be sufficient to cope with real-time events. It was observed that machine breakdowns have more significant impact on the system performance than processing time variations.

In *Event driven* policy rescheduling is triggered in response to an unexpected event that alters the current system status. Most of the approaches to dynamic scheduling use this policy.

Yamamoto and Nof (1985) studied the event driven rescheduling policy for job shop scheduling environment with random machine breakdowns. Rescheduling is triggered whenever a machine breakdown occurs. The results indicated that event driven rescheduling with lower computational burden and higher predictability outperforms the sequencing periodic policy and dispatching rules. Vieira et al. (2000a) described analytical models to estimate the performance of a single machine system under periodic and event-driven rescheduling strategies in an environment where jobs arrive dynamically. They proposed to evaluate the performance of periodic rescheduling and event driven rescheduling using analytical models that can easily and quickly estimate important performance measures, such as average flow time and machine utilisation. Vieira et al. (2000b) extended that study by investigating parallel machine systems. It was shown that rescheduling frequency can significantly affect the system performance (average flow time). A lower rescheduling frequency lowers the number of set ups. A higher rescheduling frequency allows the system to react

more quickly to disruptions but may increase the number of set-ups. All these studies agreed that the event driven rescheduling is much better than periodic rescheduling.

A *hybrid policy* reschedules the system periodically and also when an exception occurs. Events usually considered are machine breakdowns, arrival of urgent jobs, cancellation of jobs, or job priority changes.

Church and Uzsoy (1992) developed a hybrid event-driven rescheduling policy for rescheduling in a single-machine and parallel machine environment with dynamic job arrivals. Their system does rescheduling periodically. Events classified as regular occurring between periodic rescheduling are ignored until the next rescheduling moment. However, when an event is classified as urgent, complete rescheduling is immediately performed. The results indicated that the performance of periodic scheduling deteriorates as the length of rescheduling period increases, while event driven method achieves a reasonably good performance.

#### 4 Dynamic scheduling techniques

Dynamic scheduling has been solved using the following techniques (Suresh and Chaudhuri 1993; Shukla and Chen 1996; Stoop and Weirs 1996; Brandimarte and Villa 1999): heuristics, meta-heuristics, knowledge-based systems, fuzzy logic, neural networks, hybrid techniques, and multi-agent systems.

##### 4.1 Heuristics

Heuristics in this context are problem specific schedule repair methods, which do not guarantee to find an optimal schedule, but have the ability to find reasonably good solutions in a short time. The most common schedule repair heuristics are: right-shift schedule repair, match-up schedule repair, and partial schedule repair. The right-shift heuristic shifts the remaining operations of the schedule forwards in time by the amount of downtime in the event of machine failure. Match-up schedule repair strategy reschedules to match-up with the pre-schedule at some point in the future. Partial schedule repair reschedules only the operations in failure. Dispatching rules are also heuristics that have played a significant role in completely reactive scheduling. However, in completely reactive scheduling no firm schedule is generated in advance and dispatching rules are used in real-time to select the next job waiting for processing at a resource.

Yamamoto and Nof (1985) investigated the performance of the right-shift heuristic compared with dispatching rules and complete rescheduling using branch and bound. The experimental results showed that right shifting outperforms

priority rules and complete rescheduling. Mehta and Uzsoy (1999), and O'Donovan et al. (1999) used the right-shift heuristic for inserting idle time to define predictable schedules. Abumaizar and Svestka (1997) compared the performance of partial schedule repair (affected operations schedule repair), complete rescheduling, and right shift schedule repair with respect to measures of efficiency (makespan) and stability (deviation from the initial schedule). The partial schedule repair heuristic reschedules only the operations directly and indirectly affected by the disruption so as to minimise both the increase in makespan and the deviation from the initial schedule. The results demonstrated that the heuristic reduces much of the deviation and computational complexity associated with complete rescheduling and right shifting. Right shift gives the worst performance with respect to makespan due to the fact that the method is a simple shifting of the schedule by the amount of the disruption. Thus, the longer the disruption, the larger the expected shift, and the greater the increase in makespan.

Bean et al. (1991) proposed a match-up schedule repair heuristic for the shop floor rescheduling with multiple resources in the presence of machine breakdowns. The strategy reschedules to match up with the pre-schedule at some point in the future whenever a machine breakdown occurs. The experimental results showed that the method provides near-optimal solutions while achieving higher predictability than complete rescheduling. Later, Akturk and Gorgulu (1999) applied this approach for flow shop rescheduling. The results indicated that the match-up heuristic is very effective in terms of schedule quality, computation time, and schedule stability.

More specific schedule repair heuristics have been also proposed in the literature. Nof and Grant (1991) proposed several rescheduling strategies for process time variations, machine breakdown, and new job arrival in a manufacturing cell. The rescheduling strategies are: rerouting the jobs to alternative machines, job-splitting (for batch production), and complete rescheduling. Kutanoglu and Sabuncuoglu (2001) proposed several schedule repair heuristics in the presence of machine failures. The schedule repair heuristics are based on rerouting the jobs to their alternative machines. Lee and Uzsoy (1999) considered the problem of minimising makespan on a single batch-processing machine for oven scheduling in a semiconductor-manufacturing environment with dynamic job arrivals. They proposed and evaluated the performance of two schedule repair heuristics, namely delay schedule repair heuristic (delays the processing of a batch to integrate jobs arriving very soon in the future), and update schedule repair heuristic (updates the release time of the job to delay in the batch). The results indicated that the heuristics showed an excellent average performance with a modest computational burden.

A variety of dispatching rules have been used to react to real-time events in completely reactive scheduling. It was found that no rule performs well for all criteria. Hence, many investigations were carried out towards recognising a combination of several dispatching rules to find a range of system states in which the relative performance of each rule is highest. In order to assess the performance of various dispatching rules under different dynamic and stochastic conditions of the shop floor, simulation was used. Ramasesh (1990), and Rajendran and Holthaus (1999) presented excellent state-of-the-art surveys of dispatching rules in dynamic job shops and flow shops. They evaluated the performance of a variety of dispatching rules with respect to some common performance criteria discussed in the literature, such as variance of flow time, minimum and maximum flow time, mean tardiness, maximum tardiness and variance of tardiness, etc. They classified these rules into five categories: rules involving process times, rules involving due dates, simple rules involving neither process times nor due dates, rules involving shop floor conditions, and rules involving two or more of the first four categories. It has been observed that no single rule performs well for all important criteria related to flow time and tardiness of jobs. In general, it has been noted that process time based rules perform better under tight load conditions, while due date based rules perform better under light load conditions. Sabuncuoglu (1998) presented a comprehensive simulation study on scheduling rules for flexible manufacturing systems in the presence of various levels of breakdown rates and changes in processing times. He reported that no single rule is the best under all possible conditions. A comprehensive bibliography is also presented in the paper. Shafaei and Brunn (1999) in their simulation study investigated the performance of a number of scheduling rules for a dynamic job shop. The performance measure considered is an economic objective, which includes the main costs involved in a scheduling decision. Jain and Elmaraghy (1997) proposed various schedule repair heuristics for production rescheduling in flexible manufacturing systems for machine breakdown, arrival of rush jobs, increased job priority, and job cancellation. When a machine breakdown occurs, the remaining operations are performed on alternative machines. For arrival of new jobs, if the new job is not a rush job, then priority is assigned based on EDD (Earliest Due Date) or FCFS (First Come First Served) dispatching rules, otherwise highest priority is assigned to it and all the disturbed tasks are moved forward in time. When a job priority is increased or a job is cancelled, the remaining tasks are shifted forward in time on their respective machines.

## 4.2 Meta-heuristics: tabu search, simulated annealing, and genetic algorithms

In recent years, meta-heuristics (tabu search, simulated annealing, and genetic algorithms) have been successfully used to solve production scheduling problems. Meta-heuristics are high level heuristics which guide local search heuristics to escape from local optima (Glover and Laguna 1997; Pham and Karaboga 2000). Local search heuristics are neighbourhood search methods based on the idea of searching neighbourhoods. In local neighbourhood search, the search starts from some given solution, and tries iteratively to move to a better solution in an appropriately defined neighbourhood of the current solution using move operators. The search process stops when no better solution can be found in the neighbourhood of the current solution, which is the local optimum. Meta-heuristics such as tabu search, simulated annealing, and genetic algorithms improve local search to escape local optima by either accepting worse solutions, or by generating good starting solutions for the local search in a more intelligent way than just providing random initial solutions.

Tabu search, simulated annealing, and genetic algorithms have been widely used to solve static deterministic production scheduling problems in several domains including job shops, open shops, flow-hops, flexible manufacturing systems, batch processing, etc. However, little research work has addressed the use of meta-heuristics in dynamic scheduling. Dorn et al. (1995) and Zweben et al. (1994) discussed the importance of using meta-heuristics to schedule repair instead of using local search or simple heuristics as they can be trapped in a poor local optimum. Mehta and Uzsoy (1999) used tabu search to search for predictable schedules. Dorn et al. (1995) used tabu search to repair a schedule caused by uncertain time processes in steel continuous caster scheduling. Zweben et al. (1994) used simulated annealing to repair schedules for space shuttle ground operations. To repair a schedule, the system chooses between five repair heuristics using a choice-function, and uses simulated annealing to perform multiple repair iterations. It was found that tabu search and simulated annealing generate good quality schedules in a short time of period.

Chryssolouris and Subramaniam (2001) used genetic algorithms for dynamic scheduling of manufacturing job shops in the presence of machine breakdown and alternate job routine. Two performance measures were used, namely mean job tardiness and mean job cost. Whenever a dynamic event occurs, genetic algorithms are used to propose an alternative schedule. In addition, the solution of genetic algorithms was compared to several common dispatching rules. The results indicated that the performance of genetic algorithms is significantly superior to that of the common dispatching rules. Rossi and Dini (2000) used genetic algo-

rithms for dynamic batch scheduling of flexible manufacturing systems. They considered the following real-time events: arrival of a new batch, unavailability of parts to be machined (due to the failure of feeding systems, the presence of defects on work pieces, etc.), and machine breakdowns (due to unavailability of tools, unplanned maintenance, etc.). Schedules produced using dispatching rules were improved using genetic algorithms. The results showed that genetic algorithms greatly reduce the makespan. Leon et al. (1994) and Jensen (2001) used genetic algorithms to generate robust schedules and to evaluate the performance of various robustness measures. Wu et al. (1991, 1993) compared the performance of genetic algorithms and local search heuristics to generate robust schedules. The results showed the performance of genetic algorithms in generating schedules with much better makespan and stability than local search heuristics. However, Bierwirth and Mattfeld (1999) reported in their experimental results that the capabilities of genetic algorithms vanish with an increasing problem size, and they are not efficient to find a near-optimal solution in a reasonable time.

## 4.3 Multi-agent based dynamic scheduling

Most of the scheduling systems developed in industrial environments have traditionally been viewed as a top-down process of command and response that relies heavily on centralised and hierarchical models (Parunak 1996; Gou et al. 1998; Shen and Norrie 1999; Bongaerts et al. 2000; Shen et al. 2001). To ensure consistency of data across the entire enterprise, centralised and hierarchical scheduling systems (Figs. 1 and 2) rely heavily on central databases. To optimise performance, scheduling decisions are made centrally at the level of the supervisor, and then distributed to the manufacturing resource level for execution. A common architecture gives a central computer responsibility for scheduling, dispatching resources, monitoring any deviation, and dispatching corrective actions.

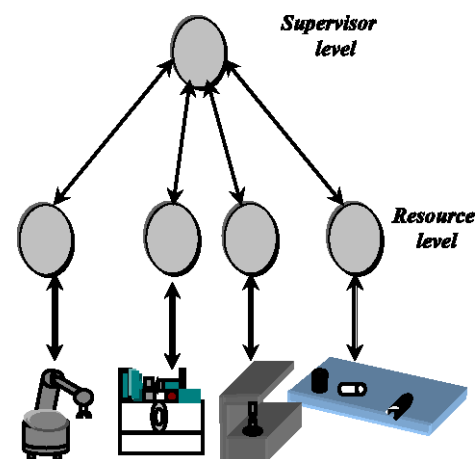
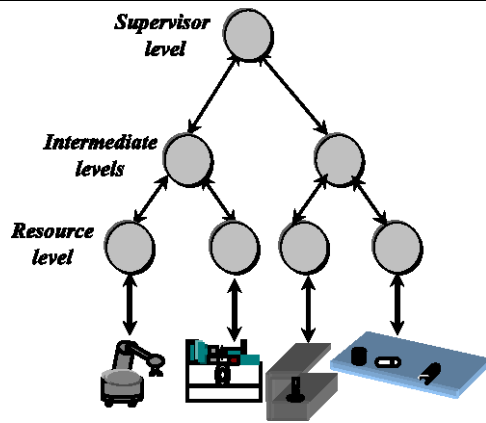


Fig. 1 Centralised architecture



**Fig. 2** Centralised hierarchical architecture

Centralised and hierarchical scheduling systems present a number of drawbacks (Parunak 1996; Tharumarajah and Bemelman 1997; Shen and Norrie 1999; Bongaerts et al. 2000; Brennan and Norrie 2001). The primary drawback is the existence of a central computer, which constitutes a bottleneck that can limit the capacity of the shop, and it is a single point of failure that can bring down the entire shop. Furthermore, modifying the configuration of hierarchically controlled manufacturing systems is expensive and time consuming as it involves expensive software rewriting. Hierarchical scheduling systems are becoming increasingly complex with the integration of manufacturing system components. Another disadvantage is that the up and down flow of information increases the latency time of decision-making. Moreover, practical experience has indicated that hierarchical centralised scheduling systems tend to have problems reacting to disturbances and may fail to respond effectively to the presence of real-time events. When a disturbance occurs, it is fed back to the high level in the hierarchy, and only after the scheduler has been adapted, the new schedule triggers a new flow of commands that forms the reaction to the disturbance. This up and down movement of information results in a slow response time leading to a low robustness. Despite the fact that centralised and hierarchical scheduling systems may provide globally better schedules in environments where real-time disturbances are rare, increasingly they are being found to be inefficient to respond to highly dynamic environments. Therefore, centralised and hierarchical scheduling is complex, difficult to maintain and reconfigure, inflexible, costly, and slow to satisfy the needs of today's complex manufacturing environments.

Global competitive pressure in manufacturing has resulted in fundamental changes in the operation of manufacturing systems. Today's systems must rapidly adapt to disturbances while maintaining shorter product cycles, improving productivity, and increasing operational flexibility. To face this challenge, the current trend has been towards

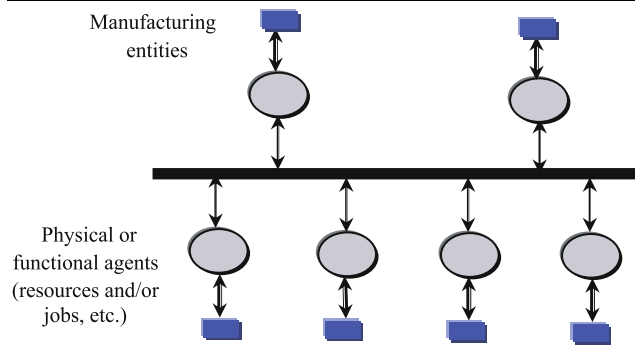
highly automated systems that are intended to offer robustness, stability, adaptability, and efficient use of available resources through a modular and distributed design (Parunak 1996; Brennan and Norrie 2001; Shen et al. 2001). There is a growing trend towards distributed shop floor organisations as a result of the need for enhanced levels of responsiveness from the shop floor to changes in markets and technologies. The primary motivation in designing these systems is to decentralise the control of the manufacturing system, thereby reducing the complexity and cost, increasing flexibility, and enhancing fault tolerance.

There is substantial evidence that multi-agent systems are one of the most promising approaches to building complex, robust, and cost-effective next-generation manufacturing scheduling systems because of their autonomous, distributed and dynamic nature, and robustness against failures (Parunak 1996, 2000; Shen et al. 2001; Brennan and Norrie 2001). A Multi-Agent System is a network of problem solvers that work together to solve problems that are beyond their individual capabilities (O'Hare and Jennings 1996).

The use of multi-agent systems to solve the problem of dynamic scheduling is motivated by the following key points (Parunak 1996, 2000; Shen and Norrie 1999; Cowling et al. 2000; Brennan and Norrie 2001; Shen et al. 2001). Firstly, multi-agent based scheduling systems recognise that data and control are distributed through the factory. These systems are composed of autonomous agents attached to each physical or functional manufacturing entity in the facility (resources, operators, parts, jobs, etc.). Local autonomy allows the agents to take the responsibility to carry out local scheduling for one or more entities in the production process and to respond locally and efficiently to local variations, increasing the robustness and flexibility of the system. Secondly, these individual agents have considerable latitude in responding to local conditions and interacting and cooperating with each other in order to achieve global optimal and robust schedules. The overall system performance is not globally planned, but emerges through the dynamic interaction of the agents in real-time. Thus, the system emerges from the concurrent independent local decisions of the agents. Thirdly, the software for each agent is much shorter and simpler than it would be for a centralised approach, and as a result is easier to write, install and maintain. Furthermore, it is possible to integrate new resources or remove existing ones with their attached agents to from the factory without making any changes to the existing software network.

#### 4.3.1 Multi-agent scheduling architectures

An increasing number of enterprises are turning to agent technology to address the complex and dynamic environments common to most enterprises and successful results



**Fig. 3** Autonomous architecture

have been achieved. Two main multi-agent architectures for dynamic scheduling have been implemented: autonomous architectures and mediator architectures. They are described in more detail in the following sub-sections.

**4.3.1.1 Autonomous architectures** In autonomous architectures (Fig. 3), agents representing manufacturing entities such as resources and jobs have the ability to generate their local schedules, react locally to local changes, and cooperate directly with each other to generate global optimal and robust schedules.

Yams (Yet Another Manufacturing System) (Parunak 1987) is one of the earliest agent-based manufacturing system which assigns an agent to each node in a control hierarchy (factory, cell, workstation, machine, jobs). The main idea of Yams is that the job agents negotiate with resource agents to assign tasks to the machine agents using the contract net protocol. Shaw (1988) developed a dynamic scheduling system in a cellular manufacturing system. In their work a manufacturing cell agent could sub-contract work to other cells through the contract net protocol (Smith 1980). Request for bid messages are broadcast to cells which evaluate operations specification and submit bids. Bids describe the estimation on the earliest finishing time or shortest processing time of the operations. The cell that optimises a predefined criterion is selected to perform the operation. Goldsmith and Interrante (1998), and Ouelhadj et al. (1998, 1999, 2000) proposed a simple multi-agent architecture for dynamic scheduling in flexible manufacturing systems which involves only resource agents. The resource agents are responsible for dynamic scheduling of the resources and they have no control over each other. They negotiate using the contract net protocol to produce a global schedule. Each resource agent performs the following functions: scheduling, detection, diagnosis, and error handling. Resource agents react locally to real-time events occurring on the corresponding resource using the corrective actions described in a knowledge base. When real-time events occur, such as machine breakdown, the resource agent renegotiates the operations in failure to find alternative resource agents.

For an increased flexibility and robustness, Sousa and Ramos (1999) proposed a multi-agent architecture for dynamic scheduling in manufacturing systems which involves job and resource agents. The job agents negotiate the operations of the job with the resource agents using the contract net protocol. When a resource agent detects a malfunction, it sends a machine fault message to every job agent that has contracted its operations. On receiving the machine fault message, the job agent renegotiates the operations in failure with other resource agents capable of performing the operations. Cowling et al. (2001, 2003, 2004) and Ouelhadj et al. (2003a, 2003b) proposed a novel multi-agent architecture for integrated and dynamic scheduling in steel production. Each steel production process is represented by an agent, including the continuous caster agents, the hot strip mill agent, the slabyard agent, and the user agent. The hot strip mill and continuous caster agents perform the robust predictive–reactive scheduling of the hot strip mill and the continuous caster, respectively. Robust predictive–reactive scheduling generates robust predictive–reactive schedules in the presence of real-time events using utility, stability, and robustness measures and a variety of rescheduling heuristics.

Recently levelled commitment contracts were proposed as an extension of the contract net protocol for increasing the economic efficiency of contracts between agents in the presence of incomplete information about future events. Sandholm (2000) described a levelled commitment contracting protocol for automated contracting in distributed manufacturing. The extended protocol allows self-interested agents to efficiently accommodate future events by giving the possibility for each agent to decommit from the contract by simply paying a de-commitment penalty to the other contract party. A de-commitment penalty is assigned to both agents in a contract to be freed from the contract.

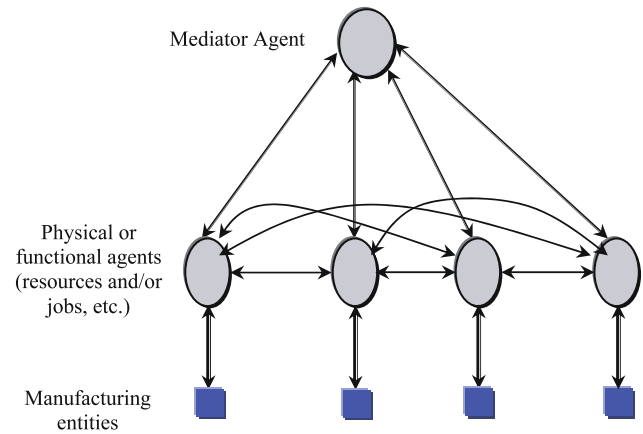
Some multi-agent-based scheduling systems used auction and currency mechanisms for inter-agent negotiation. Agents exchange resources for money at prices determined through communication of bids. Lin and Solberg (1992, 1994) proposed an autonomous multi-agent architecture for shop floor dynamic scheduling based on a currency model that combined the scheduling objectives and price mechanism. Their model represents jobs, resources, and parts by agents. Job agents negotiate with resource agents via a contract net bidding mechanism to optimise a weighted objective that is a function of due date, price, quality, and other user defined factors. The part agent enters the system with a certain currency, and solicits and evaluates bids from several resource agents capable of fulfilling the processing requirements, and selects the one that optimises its objective. Each resource agent sets its charging price based on its status, then it decides on the basis of the currency offered which of the announced jobs to consider more interesting for a possible



bid. The job agent tries to minimise the price paid, but the resource agent's goal is to maximise the price charged. Each deal is completed once the job and resource agents are mutually committed. When a resource agent is in failure, it informs the corresponding job agent, and the latter proceeds to a renegotiation process on the operations in failure with the resource agents. AARIA (Autonomous Agents for Rock Island Arsenal) (Parunak et al. 1997) is an autonomous multi-agent architecture developed for scheduling in an army manufacturing facility. Manufacturing resources, parts, and people are encapsulated as autonomous agents. The system incorporates features of schedule optimisation and fault recovery. Agents cooperate using the currency negotiation protocol.

Other multi-agent based dynamic scheduling systems used learning approaches for dynamic scheduling. Aydin and Öztemel (2000) proposed a dynamic job shop scheduling using reinforcement learning agents. The agent is trained by an improved reinforcement-learning algorithm through the learning stage and then successfully makes the decisions to schedule the operations. The scheduling system consists of two parts: the simulator and the intelligent agent. The agent selects the most appropriate priority rule to select a job to assign to a machine according to the shop conditions, while the simulator performs scheduling activities using the rule selected by the agent. Pendharkar (1999) proposed a multi-agent learning approach for dynamic scheduling. In the multi-agent architecture, the work areas are controlled by agents with a knowledge base containing the dispatching rules. The agents use genetic algorithms-based learning to update the rules in the knowledge-base at periodic intervals of time. The higher frequency of learning may help an agent to quickly adapt to variations on the shop floor.

**4.3.1.2 Mediator architectures** Despite the good performance of autonomous architectures, they usually face problems in providing globally optimised schedules and predictability in the presence of a large number of agents, such as virtual enterprises (Brennan and Norrie 2001; Shen and Norrie 1999; Bongaerts et al. 2000; Shen et al. 2001; Tharumarajah 2001). Several researchers have proposed mediator architectures for dynamic scheduling in such complex environments to combine robustness, optimality, and predictability. The mediator architecture provides computational simplicity, while being quite suitable for developing distributed industrial systems that are complex, dynamic, and composed of a large number of resource agents. Brennan and Norrie (2001), Bongaerts et al. (2000), and Cavalieri et al. (2000) showed in their comparative studies that mediator architectures have improved performance relative to autonomous architectures, because of their ability to plan further in the future in combination with their ability to react to disturbances, which can result in globally satisfactory performance.



**Fig. 4** Mediator architecture

A mediator architecture has a basic structure consisting of autonomous cooperating local agents that are capable to negotiate with each other in order to achieve production targets (Gou et al. 1998; Shen and Norrie 1999; Bongaerts et al. 2000; Shen et al. 2001). This basic structure is extended with mediator agents to coordinate the behaviour of the local agents to perform global dynamic scheduling, see Fig. 4. The mediator agents operate concurrently with local agents and contribute to the same decision making processes as the local agents. The local agents maintain their decision making autonomously, but may request advice from the mediator agents. They have the ability to advice, impose or update decisions taken by the resource agents in order to satisfy the global objectives and resolve the conflict situations. The mediator agent has an overview of the entire system, while the local agents can have a more detailed and up-to-date view of the local situations. As such, local agents can react more quickly to disturbances, while mediator agents can coordinate the agents' behaviour and often improve the global performance.

A very basic mediator architecture was proposed by Ramos (1994) for dynamic scheduling in flexible manufacturing systems. The architecture is composed of task agents, task manager agent, resource agents, and resource mediator agent. Task manager agent creates the task agents. The resource mediator agent negotiates with the resource agents the execution the tasks using the contract net protocol. When failures occur on the resources, messages on the operations in failure are sent to the mediator resource agent, which proceeds to a renegotiation process with other resource agents. This mode of handling the failures is quite simple.

For an increased robustness in complex manufacturing systems, some authors proposed the integration of mediator agents to each level of the manufacturing facility. Maturana et al. (1999) proposed the mediator architecture MetaMorph for dynamic scheduling of large heterogeneous manufacturing systems to address virtual enterprise issues combining

sub-tasking and virtual clustering of agents. Virtual enterprise partnership issues are associated with the unification of heterogeneous manufacturing subsystems into a large, and dynamic virtual coalition of co-operative subsystems. There are two main types of agents in the architecture: resource agents and mediator agents. Resource agents are used to represent manufacturing devices and operations, while mediator agents are used to coordinate the resource agents using the contract net protocol. Malfunctioning of a resource agent is kept at a local level. A resource breakdown is simulated by introducing a breakdown period into the resource. Each job allocated within the halt-period is rescheduled to other available time slots found in the same resource (the malfunctioning resource) or in a different resource. Shen et al. (2000) developed MetaMorph II for integrating the manufacturing enterprise's activities such as design, planning, scheduling, simulation, execution, material supply, and marketing services. In this architecture, the manufacturing system is organised through a hierarchy of sub-system mediators. Four types of mediators were introduced: enterprise, resource, marketing, and design mediators. Each sub-system is an agent-based system integrated into the system through a special mediator. The manufacturing resource agents are coordinated by appropriate mediators at all levels of the system. A high-level resource mediator coordinates low-level mediators such as machine, tool, worker, and transportation mediators. Cooperation among resource agents is realised by combining the mediation mechanism and the contract net protocol. Several schedule repair mechanisms have been developed for responding to the presence of real-time events, such as: arrival of new jobs, cancellation of jobs, machine breakdown, and delays in processing time of jobs. Sun and Xue (2001) developed a mediator reactive scheduling architecture for responding to changes in jobs and manufacturing resources. Manufacturing resources including facilities and resources are represented by agents that are coordinated by two mediators, namely a facility mediator and a personnel mediator, using the contract net protocol. Reactive scheduling is conducted to modify the created schedule to respond to changes of jobs such as cancellation of jobs or insertion of urgent jobs, and manufacturing conditions such as machine breakdowns, or a person's sudden sickness during the production process. Match up rescheduling strategy and agent-based collaboration are used to repair only part of the originally created schedule for improving the reactive scheduling efficiency, while maintaining the scheduling quality.

#### 4.4 Other artificial intelligence techniques

A number of dynamic scheduling problems have adopted artificial intelligence techniques such as knowledge-based systems, neural networks, case-based reasoning, fuzzy logic, Petri nets, etc. which are discussed below.

The basic motivation of knowledge-based approaches is that there is a wide variety of technical expertise on the corrective actions to undertake in the presence of real-time events. Knowledge-based systems focus on capturing the expertise or the experience of the expert in a specific domain and an inference mechanism is used to derive conclusions or recommendations regarding the corrective action to undertake. ISIS (Fox 1994; Smith 1995) developed at Carnegie Mellon in 1982, was the first attempt to use knowledge-based systems in job shop scheduling. ISIS performs a constrained-direct search to derive a schedule. The dynamic situations are handled by rescheduling the affected jobs by selectively relaxing some of the constraints. OPIS (Smith 1994) is a successor of ISIS. OPIS is a knowledge-based system developed originally for manufacturing production scheduling which uses an opportunistic problem solving process to incrementally generate and revise schedules in response to changes. OPIS implemented a blackboard architecture wherein a set of distinct heuristics, referred to as knowledge sources, are selectively employed to generate and revise the overall schedule. The schedule repair heuristics defined in OPIS are: job scheduler, resource scheduler, right-shifter, left-shifter, and demand swapper. IOSS (Park et al. 1996) is another interactive scheduling knowledge-based scheduling system based on opportunistic and interactive repair-based problem solving within blackboard architecture. SONIA (Le Pape 1994) is a knowledge-based job-shop predictive-reactive scheduling system. Various schedule repair heuristics were defined such as relaxing due dates, extending work shifts, operation postponed until the next shift, and reduction of idle times of resources by permuting operations. Some researchers (Belz and Mertens 1996) combined knowledge-based systems and simulation to pursue a richer modelling capacity of scheduling to decide on the best corrective actions to handle the real-time events. Some knowledge-based systems were developed to assist the user to react interactively to real-time events (Dutta 1990; Sarin and Salgame 1990; Henning and Cerda 2000; O'Kane 2000).

Miyashita and Sycara (1995) developed the framework CABINS for schedule repair in a job shop using case-based reasoning. Cases represent suitable repair actions. Case based reasoning allows capturing and re-use of this knowledge to repair similar situations. The schedule is repaired incrementally, when necessary, using the cases stored in the system.

Neural networks, Petri nets, and fuzzy logic have also been used to solve the problem of dynamic scheduling. Extensive discussions of these techniques can be found in Suresh and Chaudhuri (1993), Szelke and Kerr (1994), Zweben and Fox (1994), Kerr and Szelke (1995), Meziane et al. (2000).

To derive better dynamic scheduling systems, some researchers developed hybrid systems which combine various

artificial intelligence techniques. Jahangirian and Conroy (2000), and Li et al. (2000) developed a hybrid framework for dynamic scheduling consisting of a knowledge-base describing the dispatching rules, a simulation module to evaluate the performance of the dispatching rules, an artificial neural network, and genetic algorithms for machine learning to tailor the approaches to specific problem instances. Dorn (1995) used case-based reasoning and fuzzy logic for reactive scheduling of the continuous caster in the steel industry. Schmidt (1994) used fuzzy logic to diagnose critical jobs in order to reschedule them. As a result, the decision-maker on the shop floor gets the information concerning which jobs must be rescheduled now, soon, later or probably not at all. Dorn et al. (1994) used fuzzy logic for dynamic scheduling of steel continuous casting. A fuzzy logic based decision support system for parallel machine scheduling and rescheduling in the presence of uncertain disruptions in a pottery company was presented in Petrovic and Duenas (2006). The uncertain disruption considered was material shortage described by the number of disruption occurrences and disruption repair period. These parameters were specified imprecisely and modelled using fuzzy sets and level 2 fuzzy sets, respectively. Fuzzy rules were proposed to determine when to reschedule and which rescheduling method to use. Garetti and Taisch (1995), and Garner and Ridley (1994) used knowledge-based systems and neural networks in reactive scheduling. The neural networks were used to decide on the best set of dispatching rules when a real-time event occurs. Ruiz et al. (2001) proposed a fault diagnosis system for reactive scheduling in multipurpose batch chemical plants. The system combines the adaptive learning diagnostic procedure of neural networks and a knowledge-based expert system.

## 5 Comparison of solution techniques

In order to ascertain the value of the various solution techniques, there has been some published work comparing some of these techniques. These comparisons help us reason about what techniques are most suitable for dynamic scheduling. Advantages and disadvantages of these techniques have been provided by Suresh and Chaudhuri (1993), Shukla and Chen (1996), Stoop and Weirs (1996), and Brandimarte and Villa (1999).

Heuristics have been widely used to react to the presence of real-time events because of their simplicity, but they may become stuck in poor local optima. To overcome this, meta-heuristics such as tabu search, simulated annealing, and genetic algorithms have been proposed. Several comparative studies have been provided in the literature to compare the performance of tabu search, genetic algorithms, and simulated annealing. Unlike simulated annealing and

tabu search based on manipulating one feasible solution, genetic algorithms manipulate a population of feasible solutions. Genetic algorithms were found not efficient to find a near-optimal solution in a reasonable time compared to tabu search and simulated annealing which operate on a single configuration and not on an entire population (Glover et al. 1995; Jozefowska et al. 1998; Youssef et al. 2001; Zhou et al. 2001). Knowledge-based systems possess the potential for automating human expert reasoning and heuristic knowledge to run production scheduling systems. They model the shop floor by means of many hard and soft constraints. However, they usually lack the ability to optimise the system and require considerable effort to build and maintain. They are aimed at generating feasible schedules conforming to the domain knowledge. In terms of effectiveness of the decision-making capability, knowledge-based systems are limited by the quality and integrity of the specific domain knowledge. Fuzzy logic has not yet been explored to its fullest potential. Neural networks cannot guarantee to provide optimal decisions, but their learning capability makes them ideally suited for rapidly changing systems. Integrating neural networks, simulation, and expert systems seems to have a lot of promise.

Most scheduling systems developed in manufacturing environments are centralised and hierarchical. Centralised scheduling systems provide a consistent global view of the state of the enterprise and globally better schedules. However, practical experience has indicated that these systems tend to have problems with reactivity to disturbances. A large research field, currently subject of many in depth studies, regards the use of multi-agent systems in dynamic scheduling. The primary motivation in designing these systems is to decentralise the control of manufacturing systems, thereby reducing the complexity, increasing flexibility, and enhancing fault tolerance. Refusing the traditional idea of a central scheduling system, which establishes a manufacturing plan for all the machines and jobs, multi-agent systems assume the presence of several agents with a good deal of decision making autonomy, distributed inside the manufacturing system. The agents interact and cooperate with each other in order to achieve effective global performances. Local autonomy allows the agents to take the responsibility for carrying out local scheduling for one or more functional or physical components in the production process (such as machines and jobs). Agents have the ability to observe their environment and to communicate and cooperate with each other in order to ensure that local schedules lead to globally desirable schedules. Local autonomy allows the agents to respond locally to local variations, increasing the robustness and the flexibility of the system.

Several comparative studies have discussed the features of multi-agent systems that make them attractive candidates for implementing dynamic scheduling in contrast to centralised and hierarchical scheduling systems. Parunak (1996,

2000) demonstrated that multi-agent systems are well suited for applications that are modular, decentralised, likely to change frequently, ill-structured, and complex. Duffie and Piper (1986), Tharumarajah and Bemelman (1997), and Brennan and Norrie (2001) presented in their comparative studies various advantages of multi-agent systems, such as heterogeneity, high modularity, high flexibility, high robustness against failures, reduced complexity, and reduced software development cost. According to Sandholm (2000), the most important point that supports multi-agent systems is reactivity: agents can locally react to local changes faster than a centralised system could. Multi-agent systems provide the foundation for the design of an architecture that is reliable, maintainable, flexible, robust, and stable. Shen and Norrie (1999) discussed the advantages of using multi-agent systems in production scheduling which provide capabilities of integration, robustness and reactivity, flexibility, heterogeneity, and autonomy.

Two main multi-agent architectures for dynamic scheduling have been investigated: autonomous and mediator architectures. Autonomous architectures are a highly distributed form of control, where agents cooperate directly towards a common goal. In mediator architectures, the agents cooperate via a mediator agent. The comparative studies of Brennan and Norrie (2001), Shen and Norrie (1999), Bongaerts et al. (2000), Shen et al. (2001), and Tharumarajah (2001) reported that autonomous architectures have prospects of integrity, cost-efficiency, high flexibility, and a high robustness against disturbances. They are well suitable for applications with a small number of agents. However, they have problems in providing globally optimised performance, and the behaviour of the system can be unpredictable in complex environments with large number of agents. In contrast, mediator architectures show improved performance relative to autonomous architectures for developing distributed manufacturing systems, which are complex and composed of a large number of agents such as virtual enterprises. They combine robustness against disturbances with global performance optimisation and predictability.

## 6 Conclusion

A vast majority of the literature dealing with production scheduling has primarily been focused on finding optimal or near-optimal predictive schedules for simple scheduling models with respect to various criteria assuming that all problem characteristics are known. Such predictive schedules are often produced in advance in order to direct production operations and to support other planning activities. Unfortunately, most manufacturing systems operate in dynamic environments subject to various real-time events, which may render the predictive optimal schedule neither

feasible nor optimal. Therefore, dynamic scheduling is of great importance for the successful implementation of real-world scheduling systems.

We have identified two categories of real-time information commonly considered in the literature: real-time events related to resources, and real-time events related to jobs.

Dynamic scheduling has been defined under three categories: completely reactive scheduling, predictive–reactive (robust) scheduling, and robust pro-active scheduling. In completely reactive scheduling, schedules are easily generated using dispatching rules. However, the solution quality is poor due to the myopic nature of the rules which fail to provide any plan for other activities, and it is hard to predict system performance as decisions are made locally in real-time and they typically do not use global information. Predictive–reactive scheduling is the most common approach in dynamic scheduling. Predictive–reactive approaches search in a larger solution space, generate high quality schedules, and can generate better system performance to increase productivity and minimise operating costs compared with completely reactive scheduling. Simple schedule adjustments require little effort and are easy to implement. However, they may lead to poor system performance. Generating robust schedules lead to better system performance, even though robustness measures are not easy to define. Robustness is one of the key factors to preserve the stability of manufacturing systems in the presence of uncertainties. Little research work has been done on the generation of robust schedules; more research is needed towards the development of more general efficient robustness measures.

We have discussed two main alternatives to deal with the problem of updating schedules in the most effective way in the presence of real-time events: schedule repair, and complete rescheduling. Schedule repair refers to some local adjustment of the current schedule. Complete rescheduling regenerates a schedule from scratch. Complete rescheduling might in principle be better capable of maintaining optimal solutions. However, such solutions are rarely achievable in practice and computation times are likely to be prohibitive. Furthermore, frequent schedule regeneration can increase system nervousness and result in instability and lack of continuity in detailed plant schedules. Schedule repair is very practical because of the potential saving in CPU times and the stability of the system is maintained.

Several dynamic scheduling methods have been presented including: heuristics, meta-heuristics, knowledge-based systems, fuzzy logic, neural networks, Petri nets, hybrid techniques, and multi-agent systems. The comparative study provided evidence that multi-agent systems are a very promising area of current and future research in dynamic scheduling. Although there have been some research on agent-based scheduling systems, more work is still needed. In addition, in developing practical integrated

dynamic scheduling systems, it is necessary to combine together different techniques such as operational research and artificial intelligence to endow the scheduling system with the required flexibility and robustness.

## References

- Abumaizar, R. J., & Svestka, J. A. (1997). Rescheduling job shops under random disruptions. *International Journal of Production Research*, 35(7), 2065–2082.
- Akturk, M. S., & Gorgulu, E. (1999). Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, 112(1), 81–97.
- Aydin, M. E., & Öztemel, E. (2000). Job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33(2–3), 169–178.
- Aytug, H., Lawley, M. A., McKay, K., Mohan, S., & Uzsoy, R. (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., Birge, J. R., Mittenthal, J., & Noon, C. E. (1991). Match up scheduling with multiple resources release dates and disruptions. *Journal of Operations Research*, 39(3), 471–483.
- Belz, R., & Mertens, P. (1996). Combining knowledge-based systems and simulation to solve rescheduling problems. *Decision Support Systems*, 17(2), 141–157.
- Bierwirth, C., & Mattfeld, D. C. (1999). Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1), 1–17.
- Bongaerts, L., Monostori, L., McFarlane, D., & Kadar, B. (2000). Hierarchy in distributed shop floor control. *Computers in Industry*, 43(2), 123–137.
- Brandimarte, P., & Villa, A. (1999). *Modelling manufacturing systems: from aggregate planning to real-time control*. Berlin: Springer.
- Brennan, R. W., & Norrie, D. H. (2001). Evaluating the performance of reactive control architectures for manufacturing production control. *Computers in Industry*, 46(3), 235–245.
- Cavaliere, S., Garetti, M., Macchi, M., & Taisch, M. (2000). An experimental benchmarking of two multi-agent architectures for production scheduling and control. *Computers in Industry*, 43(2), 139–152.
- Chryssolouris, G., & Subramaniam, V. (2001). Dynamic scheduling of manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing*, 12(3), 281–293.
- Church, L. K., & Uzsoy, R. (1992). Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, 5(3), 153–163.
- Cowling, P. I., & Johansson, M. (2002). Using real-time information for effective dynamic scheduling. *European Journal of Operational Research*, 139(2), 230–244.
- Cowling, P. I., Ouelhadj, D., & Petrovic, S. (2000). Multi-agent systems for dynamic scheduling. In M. Garagnani, (Ed.), *Proceedings of the nineteenth workshop of planning and scheduling of the UK* (pp. 45–54). The Open University, UK.
- Cowling, P. I., Ouelhadj, D., & Petrovic, S. (2001). A multi-agent architecture for dynamic scheduling of steel hot rolling. In *Proceedings of the third international ICSC world manufacturing congress* (pp. 104–111). Rochester, NY, USA.
- Cowling, P. I., Ouelhadj, D., & Petrovic, S. (2003). A multi-agent architecture for dynamic scheduling of steel hot rolling. *Journal of Intelligent Manufacturing*, 14, 457–470.
- Cowling, P. I., Ouelhadj, D., & Petrovic, S. (2004). Dynamic scheduling of steel casting and milling using multi-agents. *Journal of Production Planning and Control*, 15, 1–11.
- Daniels, R. L., & Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2), 363–377.
- Dorn, J. (1995). Case-based reactive scheduling. In R. M. Kerr & E. Szelke (Eds.), *Artificial intelligence in reactive scheduling* (pp. 32–50). Dordrecht: Kluwer Academic.
- Dorn, J., Kerr, R. M., & Thalhammer, G. (1994). Reactive scheduling in a fuzzy temporal framework. In E. Szelke & R. M. Kerr (Eds.), *Knowledge-based reactive scheduling* (pp. 39–55). Amsterdam: North-Holland.
- Dorn, J., Kerr, R. M., & Thalhammer, G. (1995). Reactive scheduling: improving the robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning. *International Journal of Human Computer Studies*, 42, 687–704.
- Duffie, N. A., & Piper, R. S. (1986). Non-hierarchical control of manufacturing systems. *Journal of Manufacturing Systems*, 5(2), 137–139.
- Dutta, A. (1990). Reacting to scheduling exceptions in FMS environments. *IIE Transactions*, 22(4), 33–314.
- Fox, M. S. (1994). ISIS: A retrospective. Intelligent scheduling. In M. Zweben & M. S. Fox (Eds.), *Intelligent scheduling* (pp. 1–28). San Mateo: Morgan Kaufmann.
- Garetti, M., & Taisch, M. (1995). Using neuronal networks for reactive scheduling. In R. M. Kerr & E. Szelke (Eds.), *Artificial intelligence in reactive scheduling* (pp. 146–147). Dordrecht: Kluwer Academic.
- Garner, B. J., & Ridley, G. J. (1994). Application of neuronal network process in reactive scheduling. In E. Szelke & R. M. Kerr (Eds.), *Knowledge-based reactive scheduling* (pp. 19–28). Amsterdam: North-Holland.
- Glover, F., & Laguna, M. (1997). *Tabu search*. Boston: Kluwer Academic.
- Glover, F., Kelly, J. P., & Laguna, M. (1995). Genetic algorithms and tabu search: hybrids for optimisation. *Computers of Operation Research*, 22(1), 111–134.
- Goldsmith, S. Y., & Interrante, L. D. (1998). An autonomous manufacturing collective for job shop scheduling. In *The proceedings of AI & manufacturing research planning workshop* (pp. 69–74). Albuquerque. Menlo Park: AAAI Press.
- Gou, L., Luh, P. B., & Kyoya, Y. (1998). Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation. *Computers in Industry*, 37(3), 213–231.
- Henning, G. P., & Cerda, J. (2000). Knowledge-based predictive and reactive scheduling in industrial environments. *Computers and Chemical Engineering*, 24(9), 2315–2338.
- Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2), 289–306.
- Jahangirian, M., & Conroy, G. V. (2000). Intelligent dynamic scheduling system: the application of genetic algorithms. *Integrated Manufacturing Systems*, 11(4), 247–257.
- Jain, A. K., & Elmaraghy, H. A. (1997). Production scheduling/rescheduling in flexible manufacturing. *International Journal of Production Research*, 35(1), 81–309.
- Jensen, M. T. (2001). Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, 1(1), 35–52.
- Jozefowska, J., Mika, M., Roycki, R., Waligora, G., & Wglarz, J. W. (1998). Local search meta-heuristics for discrete-continuous scheduling problems. *European Journal of Operational Research*, 107(2), 354–370.
- Kerr, R. M., & Szelke, E. (1995). *Artificial intelligence in reactive scheduling*. Dordrecht: Kluwer Academic.
- Kutanoglu, E., & Sabuncuoglu, I. (2001). Routing-based reactive scheduling policies for machine failures in dynamic job shops. *International Journal of Production Research*, 39(14), 3141–3158.

- Le Pape, C. (1994). Scheduling as intelligent control of decision-making and constraint propagation. In M. Zweben & M. S. Fox (Eds.), *Intelligent scheduling* (pp. 67–98). San Mateo: Morgan Kaufmann.
- Lee, C. Y., & Uzsoy, R. (1999). Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, 37(1), 219–236.
- Leon, V. J., Wu, S. D., & Storer, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26(5), 32–41.
- Leus, R., & Herroelen, W. (2005). The complexity of machine scheduling for stability with a single disrupted job. *Operations Research Letters*, 33(2), 151–156.
- Li, H., Li, Z., Li, L. X., & Hu, B. (2000). A production rescheduling expert simulation system. *European Journal of Operational Research*, 124(2), 283–293.
- Lin, G. Y., & Solberg, J. J. (1992). Integrated shop floor control using autonomous agents. *IIE Transactions*, 24(3), 57–71.
- Lin, G. Y., & Solberg, J. J. (1994). An agent based flexible routing manufacturing control simulation system. In *Proceedings of the 1994 Winter simulation conference* (pp. 970–977).
- MacCarthy, B. L., & Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1), 59–79.
- Maturana, F., Shen, W., & Norrie, D. H. (1999). MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research*, 37(10), 2159–2173.
- Mehta, S. V., & Uzsoy, R. (1999). Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing*, 12(1), 15–38.
- Meziane, F., Vadera, S., Kobbacy, K., & Proudlove, N. (2000). Intelligent systems in manufacturing: current developments and future prospects. *Integrated Manufacturing Systems*, 11(4), 218–238.
- Miyashita, K., & Sycara, K. (1995). CABINS: a framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial Intelligence*, 76(1), 377–426.
- Muhlemann, A. P., Lockett, G., & Farn, C. K. (1982). Job shop scheduling heuristics and frequency of scheduling. *International Journal of Production Research*, 20(2), 227–241.
- Nof, S. Y., & Grant, F. H. (1991). Adaptive/predictive scheduling: review and a general framework. *Production Planning & Control*, 2(4), 298–312.
- O'Donovan, R., Uzsoy, R., & McKay, K. N. (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 37(18), 4217–4233.
- O'Hare, G., & Jennings, N. (1996). *Foundations of distributed artificial intelligence*. New York: Wiley.
- O'Kane, J. F. (2000). A knowledge-based system for reactive scheduling decision-making in FMS. *Journal of Intelligent Manufacturing*, 11(5), 461–474.
- Ouelhadj, D., Hanachi, C., & Bouzouia, B. (1998). Multi-agent system for dynamic scheduling and control in manufacturing cells. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1256–1262). Belgium.
- Ouelhadj, D., Hanachi, C., Bouzouia, B., Farhi, A., & Moualek, A. (1999). A multi-contract net protocol for dynamic scheduling in flexible manufacturing systems. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1114–1120). USA.
- Ouelhadj, D., Hanachi, C., & Bouzouia, B. (2000). Multi-agent architecture for distributed monitoring in flexible manufacturing systems (FMS). In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1120–1126). San Francisco, USA.
- Ouelhadj, D., Cowling, P. I., & Petrovic, S. (2003a). Contract net protocol for cooperative optimisation and dynamic scheduling of steel production. In A. Ibrahim, K. Franke, & M. Koppen (Eds.), *Intelligent systems design and applications* (pp. 457–470). Berlin: Springer.
- Ouelhadj, D., Cowling, P. I., & Petrovic, S. (2003b). Utility and stability measures for agent-based dynamic scheduling of steel continuous casting. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 175–180). Taipei, Taiwan. Selected in the finalist best student award.
- Ovacik, I. M., & Uzsoy, R. (1994). Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent set-up times. *International Journal of Production Research*, 32(6), 1243–1263.
- Park, J., Kang, M., & Lee, K. (1996). Intelligent operations scheduling system in a job shop. *International Journal of Advanced Manufacturing Technology*, 11, 111–119.
- Parunak, H. V. (1987). Manufacturing experience with the contract net. In M. Huhns (Ed.), *Distributed artificial intelligence* (pp. 285–310). London: Pitman.
- Parunak, H. V. (1996). Applications of distributed artificial intelligence in industry. In G. M. P. O'Hare & N. R. Jennings (Eds.), *Foundation of distributed artificial intelligence*. New York: Wiley-Interscience. Chap. 4.
- Parunak, H. V. (2000). Agents in overalls: experiences and issues in the development and deployment of industrial agent-based systems. *International Journal of Cooperative Information Systems*, 9(3), 209–227.
- Parunak, H. V., Baker, A. D., & Clark, S. J. (1997). The AARIA agent architecture: an example of requirements-driven agent based system design. In *Proceedings of the 1st international conference on autonomous agents* (pp. 482–483). California, USA.
- Pendharkar, P. C. (1999). A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling environments. *Expert Systems with Applications*, 16(2), 121–133.
- Petrovic, D., & Duenas, A. (2006). A fuzzy logic based production scheduling/rescheduling in the presence of uncertain disruptions. *Fuzzy Sets and Systems*, 157(16), 2273–2285.
- Pham, D. T., & Karaboga, D. (2000). *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. London: Springer.
- Rajendran, C., & Holthaus, O. (1999). A comparative study of dispatching rules in dynamic flow shops and job shops. *European Journal of Operational Research*, 116(1), 156–170.
- Ramasesh, R. (1990). Dynamic job shop scheduling: a survey of simulation research. *OMEGA International Journal of Management Science*, 18(1), 43–57.
- Ramos, C. (1994). An architecture and a negotiation protocol for the dynamic scheduling of manufacturing systems. In *Proceedings of IEEE international conference on robotics and automation* (pp. 8–13).
- Rossi, A., & Dini, G. (2000). Dynamic scheduling of FMS using a real-time genetic algorithm. *International Journal of Production Research*, 38(1), 1–20.
- Ruiz, D., Canton, J., Mara, N. J., Espuna, A., & Puigjaner, L. (2001). On-line fault diagnosis system support for reactive scheduling in multipurpose batch chemical plants. *Computers and Chemical Engineering*, 25(4), 829–837.
- Sabuncuoglu, I. (1998). A study of scheduling rules of flexible manufacturing systems: a simulation approach. *International Journal of Operational Research*, 36(2), 527–546.
- Sabuncuoglu, I., & Bayiz, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126(3), 567–586.

- Sabuncuoğlu, I., & Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, 18(4), 268–283.
- Sandholm, T. W. (2000). Automated contracting in distributed manufacturing among independent companies. *Journal of Intelligent Manufacturing*, 11(3), 271–283.
- Sarin, S. C., & Salgame, R. R. (1990). Development of a knowledge-based system for dynamic scheduling. *International Journal of Production Research*, 28(8), 1499–1513.
- Schmidt, G. (1994). How to apply fuzzy logic to reactive scheduling. In E. Szelke & R. M. Kerr (Eds.), *Knowledge-based reactive scheduling* (pp. 57–67). Amsterdam: North-Holland.
- Shafaei, R., & Brunn, P. (1999). Workshop scheduling using practical (inaccurate) data, Part 1: The performance of heuristic scheduling rules in a dynamic job shop environment using a rolling time horizon approach. *International Journal of Production Research*, 37(17), 3913–3925.
- Shaw, J. M. (1988). Dynamic scheduling in cellular manufacturing systems: a framework for Network decision making. *Journal of Manufacturing Systems*, 7(2), 83–94.
- Shen, W., & Norrie, D. H. (1999). Agent based systems for intelligent manufacturing: a state of the art survey. *International Journal of Knowledge and Information Systems*, 1(2), 129–156.
- Shen, W., Maturana, F., & Norrie, D. H. (2000). MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing. *Journal of Intelligent Manufacturing*, 11(3), 237–251.
- Shen, W., Norrie, D. H., & Barthes, J. P. A. (2001). *Multi-agent systems for concurrent intelligent design and manufacturing*. London: Taylor & Francis.
- Shukla, C. S., & Chen, F. F. (1996). The state of the art in intelligent real-time FMS control: a comprehensive survey. *Journal of Intelligent Manufacturing*, 7, 441–455.
- Smith, R. (1980). The contract net protocol: high level communication and control in distributed problem solver. *IEEE Transactions on Computers*, 29(12), 1104–1113.
- Smith, F. S. (1994). OPIS: A methodology and architecture for reactive scheduling. In M. Zweben & M. S. Fox (Eds.), *Intelligent scheduling* (pp. 29–66). San Mateo: Morgan Kaufmann.
- Smith, F. S. (1995). Reactive scheduling systems. In D. Brown & W. T. Scherer (Eds.), *Intelligent scheduling systems* (pp. 155–192). Dordrecht: Kluwer Academic.
- Sousa, P., & Ramos, C. (1999). A distributed architecture and negotiation protocol for scheduling in manufacturing systems. *Computers in Industry*, 38(2), 103–113.
- Stoop, P. P. M., & Weirs, V. C. S. (1996). The complexity of scheduling in practice. *International Journal of Operations and Production management*, 16(10), 37–53.
- Sun, J., & Xue, D. (2001). A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. *Computers in Industry*, 46(2), 189–207.
- Suresh, V., & Chaudhuri, D. (1993). Dynamic scheduling a survey of research. *International Journal of Production Economics*, 32(1), 53–63.
- Szelke, E., & Kerr, R. M. (1994). *Knowledge-based reactive scheduling*. Amsterdam: North-Holland.
- Tharumarajah, A., & Bemelman, R. (1997). Approaches and issues in scheduling a distributed shop-floor environment. *Computers in Industry*, 34(1), 95–109.
- Tharumarajah, A. (2001). Survey of resource allocation methods for distributed manufacturing systems. *Production Planning & Control*, 12(1), 58–68.
- Vieira, G. E., Herrmann, J. W., & Lin, E. (2000a). Analytical models to predict the performance of a single machine system under periodic and event-driven rescheduling strategies. *International Journal of Production Research*, 38(8), 1899–1915.
- Vieira, G. E., Hermann, J. W., & Lin, E. (2000b). Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of Manufacturing Systems*, 19(4), 256–266.
- Vieira, G. E., Hermann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling*, 6(1), 36–92.
- Wu, S. D., Storer, R. H., & Chang, P. C. (1991). A rescheduling procedure for manufacturing systems under random disruptions. In *Proceedings joint USA/German conference on new directions for operations research in manufacturing* (pp. 292–306).
- Wu, S. D., Storer, R. H., & Chang, P. C. (1993). One machine rescheduling heuristics with efficiency and stability as criteria. *Computers Operations Research*, 20(1), 1–14.
- Yamamoto, M., & Nof, S. Y. (1985). Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, 23(4), 705–722.
- Youssef, H., Sait, S. M., & Adiche, H. (2001). Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*, 14(2), 167–181.
- Zhou, H., Feng, Y., & Han, L. (2001). The hybrid heuristic genetic algorithm for job shop scheduling. *Computers and Industrial Engineering*, 40(3), 191–200.
- Zweben, M., & Fox, M. S. (1994). *Intelligent scheduling*. San Mateo: Morgan Kaufmann.
- Zweben, M., Daun, B., & Deale, M. (1994). Scheduling and rescheduling with iterative repair. In M. Zweben & M. S. Fox (Eds.), *Intelligent scheduling* (pp. 241–254). San Mateo: Morgan Kaufmann.