6

Adaptive Production Scheduling and Control in One-Of-A-Kind Production

Wei Li and Yiliu Tu
The University of Calgary, Canada

1. Introduction

Mass customization is one of competitive strategies in modern manufacturing (Blecker & Friedrich, 2006), the objective of which is to maximize customer satisfaction by producing highly customized products with high production efficiency. There are two starting points moving towards mass customization, mass production and one-of-a-kind production (OKP). The production volume of mass production is normally large, whereas that of OKP is usually small or extremely even just one. Mass production can achieve high production efficiency but relatively low customization, because products are designed in terms of standard product families, and produced repetitively in large volume. Comparatively, OKP can achieve high customization but relatively low production efficiency, because product design in OKP is highly customer involved, and each customer has different requirements. Therefore, the variation of customer requirements causes differences on each product. To improve production efficiency, OKP companies use mixed-product production on a flow line (Dean et al., 2008, 2009). Moreover, the production scheduling and control on OKP shop floors is severely challenged by the variation of customer requirements, whereas that in mass production is comparatively simple. Therefore, we focus on the adaptive production scheduling and control for OKP.

1.1 Characteristics of one-of-a-kind production

OKP is product-oriented, not capacity-oriented (Tu, 1996a). Customers can only choose a product within one of product families provided by an OKP company. Although customer choice is confined by product families, OKP is so customer involved that every product is highly customized based on specific customer requirements, and products differ on matters of colors, shapes, dimensions, functionalities, materials, processing times, and so on. Consequently, production of a product is rarely repeated in OKP (Wortmann et al., 1997). Moreover, OKP companies usually adopt a market strategy of make-to-order or engineering-to-order. Therefore, it is very important to meet the promised due dates in OKP. This market strategy challenges production scheduling and control differently from that of make-to-stock.

Typically, there are five types of problems challenging production scheduling and control in an OKP company. (1) Job insertion or cancellation frequently happens in OKP due to

high customer involvement. (2) Operator absence or machine breakdown needs to be carefully controlled to fulfill the critical due dates. (3) Variation in processing times usually happens to an operation, because a highly customized product is rarely repeated. (4) The overflow of work-in-process (WIP) inventories occurs. (5) Production delay on the previous day will affect the production on the current day; so will production earliness. When these problems dynamically happen to an OKP company, the daily production has to be adjusted online, i.e. adaptive production control. Therefore, OKP companies are continuously seeking new methods for adaptive production scheduling and control on shop floors.

1.2 Former research of flow shop production scheduling and control

Flow shop production scheduling has been researched for more than five decades since 1954 (Gupta & Stafford, 2006). Early research of flow shop production scheduling was highly theoretical, using optimization techniques to seek optimal solutions for *n*-job *m*-machine flow shop scheduling problems. However, the emergence of NP-completeness theory in 1976 (Garey et al., 1976) profoundly influenced the direction of research in flow shop production scheduling. NP-completeness implies that it is highly unlikely to get an optimal solution in a polynomially bounded duration of time, for a given complex problem in general. That is why heuristics are required to solve large problems.

Adaptive production control acutely challenges the research of flow shop production scheduling, because the relationship has not been completely revealed, among the number of jobs, the number of machines, job processing times and scheduling objectives. Moreover, the research of flow shop production scheduling is often based on strong assumptions, such as no machine breakdown or operator absence, processing times and some constraints are deterministic and known in advance (MacCarthy & Liu, 1993). During real production, disturbances are manifested in such occurrences as machine breakdown, operator absence, longer than expected processing times, new emergent orders, and so on (McKay et al., 2002), all of which may fail the original offline schedule and then require online re-scheduling for adaptive production control. Consequently, heuristics based on strong assumptions are not robust, making production scheduling systems inflexible (Kouvelis et al., 2005), and a large gap exists between theoretical research and industrial applications (Gupta & Stafford, 2006; MacCarthy & Liu, 1993).

1.3 Status of production scheduling and control in OKP

Currently, OKP companies primarily use priority dispatching rules (PDRs) to deal with disturbances. It is fast and simple to use PDRs to control production online, but PDRs depend heavily on the configuration of shop floors, characteristics of jobs, and scheduling objectives (Goyal et al., 1995), and no single specific PDR clearly dominates the others (Park et al., 1997). Moreover, the performance of PDRs is poor on some scheduling objectives (Ruiz & Maroto, 2005), and inconsistent when a processing constraint changes (King & Spachis, 1980). Consequently, there is a considerable difference between the scheduled and actual production progress (Ovacik & Uzsoy, 1997), and production may run into an "ad hoc fire fighting" manner (Tu, 1996a, 1996b).

Here is a real situation in Gienow Windows and Doors, Canada. Without a computer-aided system for adaptive production scheduling and control, an experienced human scheduler in Gienow carries out scheduling three days before the real production. It is an offline scheduling. Processing times of operations are quoted by Gienow's standards, which are the average processing times of similar operations in the past. On the production day, the production is initially carried out according to the offline schedule. However, real processing times of highly customized products might not be exactly the same as the quoted ones. Therefore, customer orders may be finished earlier or later than they are scheduled offline. This will cause problems such as the overflow of WIP inventories, the delay of customer orders, and so on. The production delay of customer orders is not allowed in Gienow, because the delivery schedule has a high priority. In addition, unexpected supply delays, machine breakdown and operator absence could even cause more problems. To cope with these dynamic disturbances, the shop floor managers and production scheduler in Gienow carry out the following activities based on their experience:

1. Re-allocate operators among work stages in a production line or lines.
2. Change the job sequence.
3. Postpone the production of other orders purely for a rush order
4. Cancel or insert orders into the current production.
5. Alter the production routine to divert orders from one production line to another.
6. Add more work shifts or overtime working.

Carrying out these activities by experience may avoid the overflow of WIP inventories in one stage or line, but cause it in other stages or lines, smoothing the production progress in one stage but slowing down the whole progress in Gienow. Due to the lack of an efficient computer system, Gienow does the adaptive production scheduling and control manually and inefficiently. Obviously, OKP shop floors have to be adaptively scheduled and controlled by a computer aided system (Wortmann et al., 1997; Tu, 1996b).

The rest of this chapter is organized as follows. Section 2 gives a brief literature review on flow shop production scheduling. Section 3 introduces a computer-aided production scheduling system for adaptive production scheduling and control in OKP, consisted of a feedback control scheme and a state space (SS) heuristic. Section 4 gives the results of various case studies. Finally, section 5 draws conclusions and proposes future work.

2. Literature review

In this section, we briefly review research of flow shop production scheduling from two perspectives first, seeking optimal solutions and seeking near-optimal solutions, and then discuss the requirements of heuristics for adaptive production scheduling and control.

2.1 Flow shop scheduling

2.1.1 Definition of flow shop scheduling

Scheduling is a decision making process of allocating resources to jobs over time to optimize one or more objectives. According to Pinedo (2002), one type of flow shop

consists of m machines in series, and each job has the same flow pattern on m machines. This is typically called a traditional flow shop (TFS). Another type of flow shop is called a flexible flow shop or hybrid flow shop (HFS), where there are a number of machines/operators in parallel in each of S stages. In addition to the difference of flow shop configurations, processing constraints are also different for TFS and HFS. For TFS, if the first in first out (*FIFO*) rule is applied to jobs in WIP inventories, it becomes a no pre-emption flow shop problem. It is also called a permutation (*prmu*) flow shop problem, because the processing sequence of jobs on each machine is the same. For HFS, because there are multiple machines/operators in a stage, the first job coming into a stage might not be the first job coming out of the stage. Therefore, the first come first serve (*FCFS*) rule is applied (Pinedo, 2002). Consequently, it is still a problem of no pre-emption flow shop. Another processing constraint could be no waiting (*nwt*), that is, there is no intermediate storage or WIP inventories between two machines or stages. The most common objective of flow shop scheduling is to minimize the maximum completion time or makespan, i.e. min(C_{max}). By the three parameter notation, $a/\beta/\gamma$ (Graham et al., 1979), the above problems can be notated as $Fm/prmu/C_{max}$ for m machine TFS problems with no pre-emption to minimize makespan, $Fm/nwt/C_{max}$ for m machine TFS problems with no waiting, $FFs/FCFS/C_{max}$, for S-stage HFS problems with *FCFS*, and $FFs/nwt/C_{max}$ for S-stage HFS problems with no waiting.

2.1.2 Research of flow shop scheduling for optimal solutions

2.1.2.1 Johnson's algorithm

Johnson proposed his seminal algorithm to get optimal solutions for n-job 2-machine flow shop problems in 1954 (Johnson, 1954), the objective of which is to min(C_{max}). The mathematical proof of his algorithm by using combinatorial analysis is as follows.

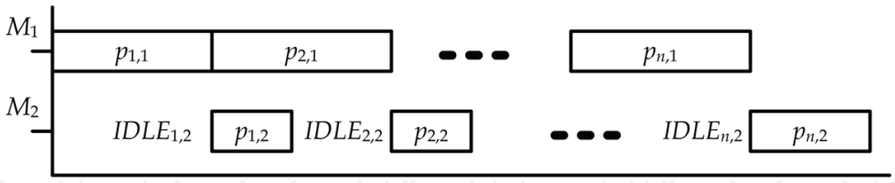

Fig. 2.1 n-job 2-machine flow shop problems, to min(C_{max})

The makespan or C_{max} consists of the sum of processing times and the sum of idle times caused by n jobs on the last machine (Fig. 2.1). For n-job 2-machine flow shop problems, $C_{max} = \sum_{i=1}^{n} p_{i,2} + \sum_{i=1}^{n} IDLE_{i,2}$. The sum of processing times of n jobs on the last machine is a constant. Thus, the objective to min(C_{max}) is converted to minimize the sum of idle times on the last machine. Johnson models the sum of idle times caused on machine 2 as $\sum_{i=1}^{n} IDLE_{i,2} = \max_{1 \le u \le n} \{K_u\}$, where $K_u = \sum_{i=1}^{u} p_{i,1} - \sum_{i=1}^{u-1} p_{i,2}$, in which $p_{i,1}$ and $p_{i,2}$ are the processing times of job i on machine 1 and machine 2 respectively.

To illustrate how to sequence n jobs, Johnson uses a combinatorial analysis approach, which is to compare two sequences, $\{\rho, i, i+1, \pi\}$ and $\{\rho, i+1, i, \pi\}$. The main difference of the two

sequences is that two jobs exchange the positions, and ρ is a subset for selected jobs, π for unselected jobs, $\rho \cap i \cap i{+}1 \cap \pi = \emptyset$, and $\rho \cup i \cup i{+}1 \cup \pi = \{n\}$. An optimal ordering of jobs is given by the following scheme. Job i proceeds job i+1, if max $\{K_{1u}, K_{1u+1}\} \leq$ max $\{K_{2u}, K_{2u+1}\}$. By subtracting $\sum_{i=1}^{u+1} p_{i,1} - \sum_{i=1}^{u-1} p_{i,2}$ from every term of equation in the above scheme, we can get min $\{p_{i,1}, p_{i+1,2}\} \leq$ min $\{p_{i+1,1}, p_{i,2}\}$, and Johnson's algorithm (JA) is developed accordingly.

2.1.2.2 Extension of combinatorial approach

Dudek and Teuton extend Johnson's combinatorial approach to n-job m-machine flow shop problems to min(C_{max}) (Dudek & Teuton, 1964), comparing the same two sequences as in Johnson's proof, and then develop their dominance conditions. Dudek and Teuton began the analytical framework for the development of dominance conditions for flow shop scheduling, although their initial method is shown to be incorrect later (Karush, 1965).

Smith and Dudek correct Dudek and Teuton's combinatorial approach, by introducing partial enumeration into dominance conditions (Smith & Dudek 1967). They propos two checks of dominance conditions. One is job dominance check and the other is sequence dominance check. The job dominance checks two different sequences, $\{\rho, i, i{+}1, \pi', \pi''\}$ and $\{\rho, i{+}1, \pi', i, \pi''\}$, in which π' and π'' are all possible combinations of exclusive subsets of π. The sequence dominance checks another two sequences, $\{\rho, \pi\}$ and $\{\rho', \pi\}$, in which ρ and ρ' are different permutations of the same selected jobs. The two dominance checks theoretically guarantee the optimal solution, but practically are still time consuming.

Based on D-T's framework, Szwarc proposes an elimination rule different from S-D's dominance checks (Szwarc, 1971a, 1971b). Let t (ρa, k) be the completion time of all jobs of sequence ρa on machine M_k. Then t (ρa, k) = max $\{t$ (ρa, k-1), t (ρ, k)$\}$ + $p_{a,k}$ with t ($\emptyset$, k) = t (ρ, 0) = 0, where k = 1,…,m. Define the difference of completion times of two sequences as $\Delta_k = t$ (ρab, k) – t (ρb, k), for k = 2,…,m. The elimination rule is to eliminate all sequences of the form ρb if $\Delta_{k-1} \leq \Delta_k \leq p_{a,k}$. However, Szwarc clearly stated that "*if there is no job c such that for all k:* $c_1 \leq c_k$ *or* $c_m \leq c_k$*, then no single sequence could be eliminated. In this case, the elimination method offers no advantage since we could have to consider all n! sequences*".

2.1.2.3 Branch and bound methods

Besides the combinatorial approach, a branch and bound (BB) method is also a general framework for NP-hard problems. It can be used to get optimal solutions to flow shop scheduling problems (Ignall & Schrage, 1965; Lageweg et al., 1978).

Usually, there are mainly three components in a BB method, a search tree, a search strategy, and a lower bound. A search tree represents the solution space of a problem (Fig. 2.2), the nodes on the tree represent subsets of solutions, and the descendants or child-nodes are given by a branching scheme. For an n-job m-machine flow shop problem, the search tree begins with a virtual node 0. For the first position in a sequence, there are n candidates or nodes, i.e. each of n jobs can be a candidate for position 1. If one job is selected for position 1, it will have n-1 descendants or child-nodes. Consequently, there are n×(n-1) nodes for position 2, n×(n-1)×(n-2) nodes for position 3, and finally, n! nodes for the last position n.

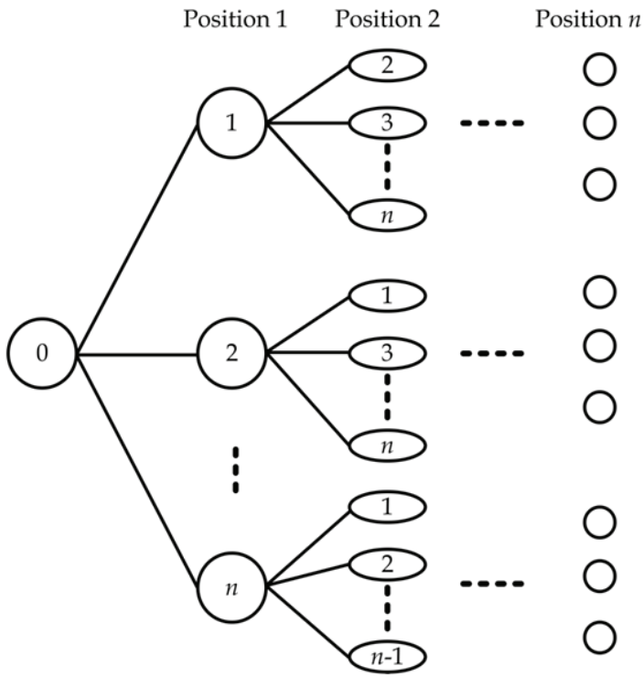

Fig. 2.2 A solution space of a BB method

At each node, a lower bound is calculated in terms of makespan for all permutations that descend it. For each position, all nodes are examined and a node with the least lower bound is chosen for branching. When a node represents an allocation of all jobs and has a makespan less than or equal to the lower bound, it is an optimal solution.

2.1.3 Heuristics for near-optimal solutions

Framinan et al. propose a general framework for the development of heuristics (Framinan et al., 2004). It has three phases: index development, solution construction and solution improvement. Phase 1, index development, means a heuristic arranges jobs according to a certain property of processing times. For example, Campbell et al. propose a CDS heuristic for an n-job m-machine TFS problem to min(C_{max}) (Campbell et al., 19770). CDS arranges jobs as follows. If there is a counter (Ctr) pointing to a machine j, then for each job i (i = 1,…,n) the sum of processing times on the first Ctr machines is regarded as its processing time on virtual machine 1, and that on the rest m–Ctr machines as on virtual machine 2. Then apply JA to this virtual 2-machine flow shop problem to get a sequence. As Ctr changes from machine 1 to machine m–1, m–1 sequences are generated by CDS, and the one with the minimum makespan is the final solution. In phase 2, solution construction, a heuristic constructs a job sequence by a recursive procedure, trying to insert an unscheduled job into a partial sequence until all jobs are inserted. NEH heuristic (Nawaz et al., 1983) is a typical heuristic in phase 2, for an n-job m-machine TFS problem to min(C_{max}). NEH constructs a job sequence as follows. Step 1, NEH heuristic calculates the sums of processing times on all of m machines for each of n jobs, and then arranges these sums in a non-ascending order. Step 2, NEH heuristic schedules the first two jobs in the order to get a partial sequence. Step 3, NEH heuristic inserts the third job into three possible positions to get another partial sequence, and so on. Finally, NEH heuristic inserts the last job into n possible positions, and then determines the final sequence. In phase 3,

solution improvement, heuristics have two main characteristics, an initial sequence generated by other heuristics and artificial intelligence to improve the initial sequence. One typical heuristic in phase 3 is an iterated greedy (IG) heuristic (Ruiz & Stützle, 2007), denoted as IG_RS heuristic. IG method consists of two central procedures, destruction and construction. The initial sequence of IG_RS heuristic is generated by NEH heuristic. For destruction, IG_RS heuristic randomly removes a number of d jobs from the initial sequence resulting a partial sequence π_D; and for construction, IG_RS heuristic follows step 3 of NEH heuristic to insert each of d jobs back in to π_D. Heuristic development in phase 1 is beneficial for future heuristic development in the other two phases (Framinan et al., 2004).

Ruiz and Maroto (2005) compare 19 heuristics for $Fm/prmu/C_{max}$ problems, and concluded that NEH heuristic is the best, CDS heuristic the eighth, and two PDRs (LPT and SPT rules) the worst. However, CDS heuristic has the second simplest computational complexity among the first 8 heuristics, $O(m^2n+mnlogn)$. Moreover, King and Spachis (1980) compare 5 PDRs and CDS heuristic for two different TFS problems, $Fm/prmu/C_{max}$ and $Fm/nwt/C_{max}$. They conclude that CDS heuristic and LWBJD (least weighted between jobs delay) rule are the best for $Fm/prmu/C_{max}$ problems and MLSS (maximum left shift savings) rule is the best for $Fm/nwt/C_{max}$ problems, but no single method is consistently the best for both $Fm/prmu/C_{max}$ and $Fm/nwt/C_{max}$ problems.

The literature on HFS is still scarce (Linn & Zhang, 1999; Wang, 2005). According to Botta-Genoulaz (2000), CDS heuristic is the best of 6 heuristics for HFS problems, including NEH heuristic. The problem in Botta-Genoulaz (2000) is an n-job S-stage HFS problem to minimize the maximum lateness. It is converted to an n-job S+1-stage HFS problem to min(C_{max}). The processing time of job i in stage S+1 is calculated by $p_{i,S+1} = D_{max} - d_i$, i = 1,…,n, where D_{max} = max(d_k), and d_k is the due date of job k, k = 1,…,n. When applying CDS heuristic to HFS problems, Botta-Genoulaz converts the processing times, $p'_{i,j} = p_{i,j}/OPTR_j$, j = 1,…,S+1, where $p_{i,j}$ is the original processing time of job i in stage j, and $OPTR_j$ is the number of operators/machines assigned to stage j.

For $FFs/nwt/C_{max}$ problems, Thornton and Hunsucker (2004) propose an NIS heuristic, the best among CDS heuristic, LPT and SPT rules, and a heuristic of random sequence generation. Different from CDS heuristic, NIS heuristic uses a filter concept to convert a $FFs/nwt/C_{max}$ problem to a virtual 2-machine problem, and then applies JA to get a job sequence. The stages before the filter are regarded as virtual machine 1, after the filter as virtual machine 2, and the stages that are covered by the filter are ignored. The filter goes from stage 2 to stage S–1, and the width of filter changes from 1 to S–2. In total, there are 1+(S–1)×(S–2)/2 sequences generated by NIS heuristic and the one with the minimum makespan is the final schedule.

2.2 Requirements for adaptive production control

2.2.1 Three criteria

Three main criteria are used to evaluate a heuristic for adaptive production scheduling and control (Li et al., 2011a): optimality, computational complexity, and flexibility. Usually optimality is used to evaluate a heuristic for offline production scheduling. However, when adaptive production control is taken into consideration, the computational complexity becomes critical. That is why some heuristics based on artificial intelligence are not suitable for adaptive production control, although they can get better solutions. Another criterion is

the flexibility, that is, whether a heuristic can deal with a disturbance. Of course, different situations have different requirements for optimality, computational complexity, and flexibility of a heuristic. There is inevitably a trade-off among these criteria, and the selection of heuristics for production scheduling and control depends on specifics of different situations, such as the value of optimality as compared to near optimal scheduling, as well as the type and volume of disturbances that underlies the requirements of response time.

2.2.2 Summary of existing heuristics for adaptive production control

For optimality, heuristics in phase 3 can get better solutions than heuristics in phases 1 and 2. However, for computational complexity, they take much longer time. For example, an adaptive learning approach (ALA) heuristic is in phase 3 for $Fm/prmu/C_{max}$ problems (Agarwal et al., 2006). The deviation of ALA heuristic is only 1.74% for Taillard's benchmarks (Taillard, 1993), much better than 3.56% of NEH heuristic. However, for the largest instance in Taillard's benchmarks, i.e. 500 jobs and 20 machines, it takes more than 19 hours for ALA heuristic to get a solution, more than 20 hours for Simulated Annealing, and more than 30 hours for Tabu search (Agarwal et al., 2006). Even for the recent IG_RS heuristic, it takes 300 seconds to get a solution to a 500-job 20-machine instance. For flexibility, we need to see if a heuristic can deal with a disturbance. According to Pinedo (2002), there are three types of disturbances in general for flow shop production, job insertion or cancellation, operator absence or machine breakdown, and variation in processing times. The perfect production information in OKP is available only after the production (Wortmann, 1992). Therefore, if a heuristic operates the known processing time only, it cannot deal with variation in processing times.

The performance of first eight of 19 heuristics is summarized in Table 2.1, and the optimality of each heuristic is quoted from Ruiz and Maroto (2005). However, there is a discrepancy of optimality of heuristics in the literature, because the optimality is evaluated by the deviation from the best known upper bounds that are under continuous improvement. For example, the deviation of 3.33% is for NEH and 9.96% for CDS in Ruiz and Maroto (2005), but 3.56% for NEH and 10.22% for CDS in Agarwal et al. (2006), and 3.59% for NEH and 11.28% for CDS in our case study. In the table, the column "Opt." means the optimality on Taillard's benchmarks for $Fm/prmu/C_{max}$ problems, "I/C" means the job insertion or cancellation, "OA/MB" means the operator absence or machine breakdown, and "Var." means the variation in processing times. The mark of "Yes§" means a heuristic can deal with a disturbance only with a modification of processing times, e.g. in Botta-Genoulaz (2000).

		Computational Complexity		Flexibility		
	Opt.		Note	I/C	OA/MB	Var.
NEH	3.33%	$O(mn^2)$	$O(mn^3)$	Yes	Yes§	No
Suliman	6.21%	Intractable	CDS first, then swap job pairs	Yes	Yes§	No
RAES	7.43%	Intractable	RA first, then swap jobs	Yes	Yes§	No
HoCha	8.06%	Intractable	CDS first, then swap job pairs	Yes	Yes§	No
RACS	9.17%	Intractable	RA first, then swap jobs	Yes	Yes§	No
Koula	9.22%	$O(m^2n^2)$	JA first, then job passing	Yes	Yes§	No
HunRa	9.69%	$O(mn+n\log n)$	3 × Palmer's slope index	Yes	Yes§	No
CDS	9.96%	$O(m^2n+mn\log n)$	JA	Yes	Yes§	No

Table 2.1 Summary of 8 heuristics for adaptive production scheduling and control

It is self-illustrative for optimality and flexibility of each heuristic in the above table. We only discuss the computational complexity in the following. NEH heuristic, in its original version, has a computational complexity of $O(mn^3)$, but, by calculating the performance of all partial sequences in a single step, its complexity is reduced to $O(mn^2)$ (Taillard, 1990). Both Suliman (Suliman, 2000) and HoCha (Ho & Chang, 1991) heuristics use CDS heuristic to generate an initial sequence, and then exchange job pairs to improve the performance, but they use different mechanisms for job pair swaps. Because the number of job pair swaps depends on the calculation of performance of each job pair, the computational complexities of Suliman and HoCha heuristics are intractable. Job swaps are also involved in RACS and RAES heuristics (Dannenbring, 1977), and their computational complexities are intractable too. These two heuristics are based on a rapid access (RA) heuristic (Dannenbring, 1977), which is a mixture of JA and Palmer's slope index (Plamer, 1965). Koula heuristic (Koulamas, 1998) is not purely for permutation flow shop problems. The job passing is allowed in Koula heuristic, because Potts et al. (1991) point out that a permutation schedule is not necessarily optimal for all n-job m-machine flow shop problems. Koula heuristic extensively uses JA to generate initial sequences, and then job passing is allowed to make further improvement. The overall computational complexity of Koula heuristic is $O(m^2n^2)$. HunRa heuristic (Hundal & Rajgopal, 1988) is a simple extension of Palmer's slope index. HunRa heuristic generates three sequences, one by Palmer's slope index, the other two by calculating indices differently. Therefore, the HunRa heuristic has the same computational complexity as Palmer's slope index, $O(mn+n\log n)$. Usually, the number of jobs n is much larger than the number of machines m, thus, the computational complexity of $O(m^2n+mn\log n)$ for CDS heuristic is comparable with that of $O(mn+n\log n)$ for HunRa heuristic.

For an industrial instance in Gienow with 1396 jobs and 5 machines, it takes NEH heuristic more than 70 seconds to generate a sequence, which is too slow to keep up with the production pace in Gienow. Therefore, NEH and other five heuristics, with computational complexity higher than $O(mn^2)$, are not suitable for adaptive production scheduling and control in Gienow. It takes less than one second for CDS or HunRa heuristics to generate a sequence for the same industrial instance. However, their performance is not good from the optimality perspective, with more than 9% deviation on Taillard's benchmarks.

3. Adaptive production scheduling and control system

For adaptive production scheduling and control, it is necessary not only to monitor the production on the shop floor, but also to give a solution in time when a disturbance happens. Our computer-aided system for adaptive production scheduling and control in OKP consists of a close-loop structure and a state space (SS) heuristic.

3.1 The feedback control scheme

For adaptive production scheduling and control, a computer-aided scheduling and control system has been proposed as illustrated in Fig. 3.1, which consists of SS heuristic and a simulation model called temporized hierarchical object-oriented coloured Petri nets with changeable structure (THOCPN-CS) (Li, 2006). High customization and dynamic disturbances in OKP demand for a great effort on a simulation model. Simultaneously, adaptive production control demands for solutions in a short time. Therefore, the unique feature of the THOCPN-CS simulation model makes it easy and flexible to simulate frequent changes in OKP for adaptive production control. Steps to achieve adaptive production scheduling and control in OKP are summarized as follows.

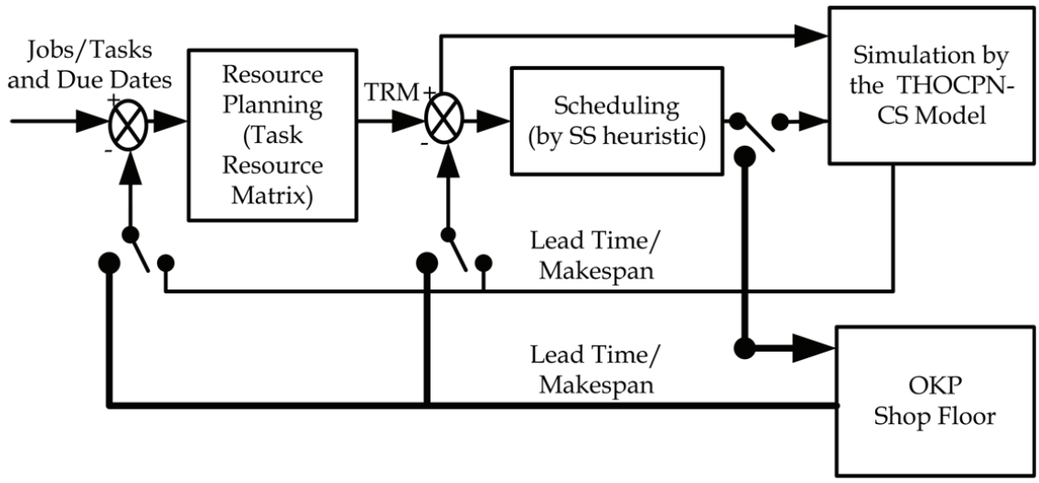

Fig. 3.1 A computer-aided production scheduling and control system

Step 1. Assign possible manufacturing resources (e.g. operators/machines) to each stage, and hence form a task-resource matrix (TRM) with jobs.

Step 2. Schedule the jobs by SS heuristic for offline scheduling, generating a sequence with the good performance for the next step.

Step 3. Simulate the production by the THOCPN-CS model, and identify the bottleneck stage(s) and overflow of WIP inventories. Human schedulers may carry out some adjustment to smooth the production flow, such as re-allocate operators/machines in stage(s), take some jobs away and then re-schedule the rest jobs, and so on.

Step 4. Re-schedule the jobs by both SS heuristic and human schedulers for offline scheduling. For online re-scheduling, re-schedule the jobs by either or both of the heuristic and scheduler, which depends on the time allowance for online re-scheduling.

Step 5. Repeat Steps 3 and 4 in the offline production scheduling phase until a satisfactory production schedule is obtained. This production schedule contains a job sequence and a number of operators/machines in each stage. In the adaptive production control phase, this step may be omitted, depending on specific requirements.

Step 6. Deliver the production schedule to the shop floor and switch the control loop from the simulation model to the shop floor.

Step 7. If any disturbance occurs on a shop floor, switch the control loop back to the simulation model, and go back to Step 3 if operators/machines re-allocation is necessary, or go back to Step 4.

Through repeating the above-mentioned steps iteratively, the production on OKP shop floors can be adaptively scheduled and controlled.

3.2 The state space heuristic

SS heuristic is mainly for HFS problems. Because there are multiple operators in each stage and the capacity of WIP inventories is limited, SS heuristic is not only to min(C_{max}), but also to maximize the utilization, max(*Util*). There are two concepts used in SS heuristic, a state space concept and a lever concept.

3.2.1 The state space concept

Consider a hybrid flow line with 3 work stages and 2 operators in each stage (see Fig. 3.2).

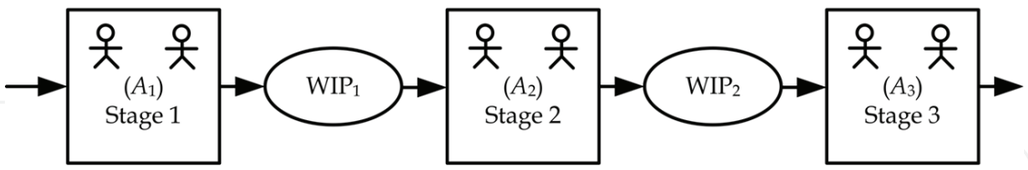

Fig. 3.2 A 3-stage flow line with 2 operators in each stage

The operators in each stage follow a *FCFS* rule. Then there is a next available time of each stage, A_s, where $A_s = \min(a_{s,k})$, for $k = 1,\ldots,OPTR_s$, in which $a_{s,k}$ is the next available time of operator k in stage s, and $OPTR_s$ is the number of operators in stage s. There are S–1 time differences between S-stage available times. In the example above, there are two differences of the next available times, A_2–A_1, and A_3–A_2. If we regard such a difference as a space, $SPACE_s = A_{s+1} - A_s$ for $s = 1,\ldots,S$-1, then $SPACE_s$ is a time period available for stage s to finish a job without causing idle to an operator in stage s+1. If the completion time of job i in stage s is larger than the next available time of stage s+1, then such a job causes idle to stage s+1, $IDLE_{i,s} = c_{i,s} - A_{s+1}$ where $c_{i,s}$ is the completion time of job i in stage s, $c_{i,s} = \max(A_s, c_{i,s-1}) + p_{i,s}$. If the completion time of job i in stage s is smaller than the next available time of stage s+1, then there are two possibilities depending on whether WIP is full. If the WIP inventory, WIP_s, is full, then a delay happens to operator k who processed job i in stage s, $DELAY_{i,s} = A_{s+1} - c_{is}$. Such a delay means that, after finishing job i, operator k in stage s has to hold it in hand for $DELAY_{i,s}$ time units until there is a vacancy in WIP_s. Therefore, the next available time of operator k in stage s is delayed. Alternatively, if WIP_s is not full, job i goes into the inventory, and there is no *IDLE* or *DELAY*.

The main idea of SS is to find a job that fits S–1 spaces, without causing *IDLE* or *DELAY*. After a job i is processed on a line, the next available times are changed, and the space is changed accordingly. Greater *IDLE* and *DELAY* are not good for production if objectives are to min(C_{max}) and max(*Util*), while greater *SPACE* is good to some extent.

From the foregoing description of SS, we can see *IDLE* and *DELAY* are evaluated according to job i and stage s, but *SPACE* is only evaluated by stage s. To make *SPACE* both job and stage dependant, there are two ways to model *SPACE*. One model is $SPACE_{i,s} = c_{i,s+1} - A_s$, for $s = 1,\ldots,S$-1. The other model is $SPACE_{i,s} = p_{i,s+1}$, for $s = 1,\ldots,S$-1. In our current version of SS heuristic, we use the latter model of *SPACE*, reducing one calculation in iteration and increasing the computation speed for adaptive control. However, we illustrate the alternative model, $SPACE_{i,s} = c_{i,s+1} - A_s$, in section 4 to show the flexibility of SS concept.

3.2.2 The lever concept in SS

From our previous research on TFS problems, we find that the lever concept is suitable for flow shop production (Li et al., 2011b), which means *IDLE* (or *DELAY*) in an earlier stage is worse for min(C_{max}) objective than in a later stage. Consider a lever where force F takes effect and causes a torque of $F \times L$, where F is the unit of force and L is the length of force arm. An S-stage flow line is modelled as a lever, and $IDLE_{i,s}$ or $DELAY_{i,s}$ has a torque effect manifested as $IDLE_{i,s} \times LVR\_IDLE_s$ or $DELAY_{i,s} \times LVR\_DELAY_s$.

The lever concept for *IDLE* in SS is shown in Fig. 3.3. For an *S*-stage flow line, a job could cause at most *S*–1 times of *IDLE*. No *IDLE* is caused in stage 1 and an *IDLE* takes effect in the next stage. Therefore, the fulcrum of a lever for *IDLE* is set between stages *S*–1 and *S*, and the length of arm for an *IDLE* caused by stage *s* in stage *s*+1 is $LVR\_IDLE_s = S–s$.

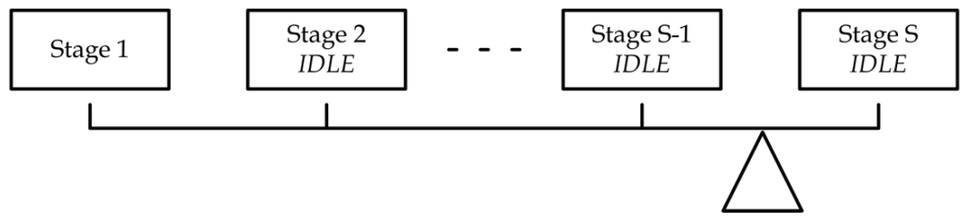

Fig. 3.3 A lever concept for *IDLE* in SS

The lever concept for *DELAY* in SS is shown in Fig. 3.4. Like the number of possible *IDLE*s, there could be *S*–1 *DELAY*s, and no *DELAY* in stage *S*. But a *DELAY* takes effect in current stage *s*, whereas *IDLE* is in the next stage. Therefore, one unit of *DELAY* in stage *s* should be worse than one unit of *IDLE* in stage *s*. Thus, the length of arm for a *DELAY* is $LVR\_DELAY_s = S–s+1$, for $s = 1,\ldots,S-1$. The fulcrum of a lever for *DELAY* is set in stage *S*.

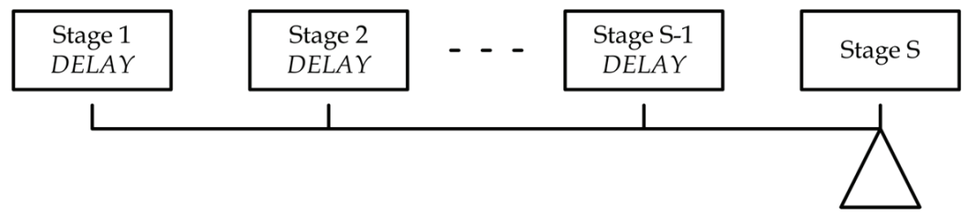

Fig. 3.4 A lever concept for *DELAY* in SS

There is also a lever concept for *SPACE* in SS, shown in Fig. 3.5. The length of force arm for a space is $LVR\_SPACE_s = s$, for $s = 1,\ldots,S-1$. The fulcrum of a lever for *SPACE* is set between stage 1 and stage 2, which means *SPACE* in a later stage is better than in an earlier stage.

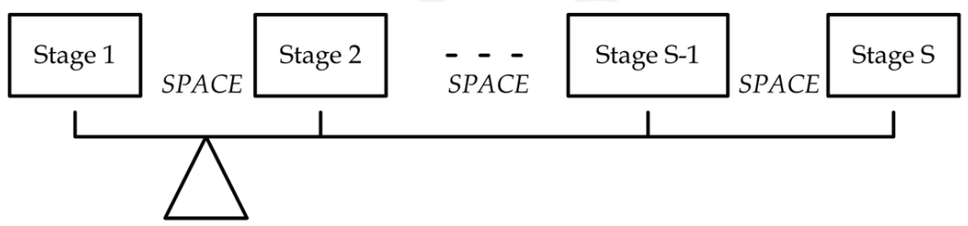

Fig. 3.5 A lever concept for *SPACE* in SS

Therefore, all *SPACE*s, *IDLE*s, and *DELAY*s are converted to torques, that is, $SPACE'_{i,s} = p_{i,s+1} \times LVR\_SPACE_s$, $IDLE'_{i,s} = IDLE_{i,s} \times LVR\_IDLE_s$, and $DELAY'_{i,s} = DELAY_{i,s} \times LVR\_DELAY_s$. The job selection scheme is $\max_{1 \le i \le n} \left[\sum_{s=1}^{S-1} SPACE'_{i,s} - \left(\sum_{s=1}^{S-1} IDLE'_{i,s} + \sum_{s=1}^{S-1} DELAY'_{i,s} \right) \right]$, that is, to select the job with the maximum torque difference between *SPACE*'s and *IDLE*'s + *DELAY*'s.

3.2.3 Steps to achieve the SS heuristic

Two items should be taken into consideration for initial job selection in SS. One is the number of initial jobs, and the other is the initial job selection scheme. The number of initial jobs is set as min($OPTR_s$, for s = 1,…,S). The reason is that if the number of initial jobs is smaller than min($OPTR_s$), then the first available time of a stage is zero since all operators are available at time zero; if the number is greater than min($OPTR_s$), then the number of (initial job number – min($OPTR_s$)) jobs are not selected by the state space concept.

For initial job selection scheme, five 1×S vectors are introduced as follows: $Vector\_1 = [0]_{1\times S}$; $Vector\_3 = [APT_s]_{1\times S}$, where $APT_s = \sum_{i=1}^{N} p_{i,s} / n$ is the average processing time of stage s; $Vector\_5 = [\max(p_{i,s}),\ i = 1,\ldots,n]_{1\times S}$ for s =1,…,S is the maximum processing time of stage s; $Vector\_2 = Vector\_3/2$; $Vector\_4 = Vector\_3+[Vector\_5-Vector\_3]/2$. The initial number of jobs are selected according to $\min(\sum_{s=1}^{S}\left|p_{i,s} - Vector\_v(s)\right|)$ for i = 1,…,n, which means the minimum absolute difference between one job's processing times and the vector.

Step 1. Determine the number of operators in each stage, i.e. $OPTR_s$. (a): Calculate n and S. (b): Set an expected throughput rate, r, which means a job is to be finished in every r time units. (c): $OPTR_s$ = Roundup (APT_s/r). (d): Set the start time of every operator to 0. (e): Put all of n jobs into a candidate pool. (f): Set an output sequence to be a 1×n zero vector, *Sequence_v*.

Step 2. Set the capacity of each of S–1 WIP inventories.

Step 3. Calculate five vectors for initial job selection.

Step 4. FOR v = 1:5, an iteration loop to select initial jobs according to one *Vector_v*.

Step 5. Select a number of min($OPTR_s$, for s = 1,…,S) jobs according to *Vector_v* by the equation $\min(\sum_{s=1}^{S}\left|p_{i,s} - Vector\_v(s)\right|)$. Then put selected jobs into a *Sequence_v* and eliminate them from the candidate pool. Calculate the next available time of each operator, the next available time of each stage, namely *STATE*, and WIP inventory status, namely *WIP_Status*, which is initially a 1×(S–1) zero vector.

Step 6. FOR i = min($OPTR_s$) + 1: n, an iteration loop to sequence rest n – min($OPTR_s$) jobs.

Step 7. According to *STATE* and *WIP_Status*, calculate $IDLE'_{is}$, $DELAY'_{is}$ and $SPACE'_{is}$.

Step 8. Select job i according to $\max_{1\le i\le n}\left[\sum_{s=1}^{S-1} SPACE'_{i,s} - \left(\sum_{s=1}^{S-1} IDLE'_{i,s} + \sum_{s=1}^{S-1} DELAY'_{i,s}\right)\right]$, and then put such job number into *Sequence_v* and eliminate it from the candidate pool.

Step 9. Calculate intermediate completion time of a partial schedule *Sequence_v*, update *WIP_Status*, and update *STATE*.

Step 10. END i. Calculate the utilization of a line. (a): Calculate utilization of each stage first, $Util_s = (\sum_{i=1}^{n} p_{i,s} / OPTR_s)/(c_{nks}-c_{1k's-1})$, c_{1k0} = 0, s = 1,…,S, in which c_{nks} is the completion time of the last job in stage s, and $c_{1k's-1}$ is the start time of the first job in stage s, i.e. the completion time of the first job in stage s-1. (b): Calculate the average utilization of each stage, i.e. $Util$ = average ($Util_s$), s = 1,…,S.

Step 11. ND v. Output each of five sequences and related makespan and utilization, and the minimum makespan and the maximum utilization are regarded as the final performance of SS.

3.2.4 The computational complexity of SS

The computational complexity of SS heuristic consists of two parts, job selection and makespan calculation.

For job selection, if the state of a flow line is known, then to select one out of n unscheduled jobs takes $S \times n$ operations, which means the computational complexity for adaptive control is $O(Sn)$. As n decreases from n to 1, the overall selection of n jobs takes $S \times n \times (n+1)/2$ operations. Although SS heuristic generates five sequences for an n by S HFS problem, the computational complexity of SS heuristic for job selection is $O(Sn^2)$, because only the highest order of operations is counted in computational complexity.

For makespan calculation, we can model an n-job S-stage HFS problem by a 2-dimension matrix, where the row dimension is for jobs, and the column dimension for stages. The makespan calculation could be carried out along the column dimension. It means that, if the input sequence of n jobs in stage 1 is known, then the output sequence of n jobs in stage 1 (or the input sequence in stage 2) can be calculated; the output sequence is in a non-descending order of completion times of n jobs; and then the output sequence in stage 2 can be calculated, and so on, finally the output sequence in stage S can be calculated. However, the capacities of WIP inventories are limited, which means the completion times of jobs in stage s are constrained by the next available times of operators in stage s+1. For example, when calculating the output sequence of stage s, if a job i's completion time in stage s causes an overflow of WIP_s, which means at that time the WIP_s is full and there is no operator available in stage s+1 to process a job in WIP_s, then a $DELAY$ happens to such job i. This $DELAY$ means the job i's completion time is delayed to a later time, and so is the next available time of operator k, who processes the job i in stage s. Consequently, the $DELAY$ affects the completion times of all jobs following job i in stage s, and the completion times in the previous stage need to be checked because of the limitation on WIP inventories. In an extreme situation, when a $DELAY$ happens in stage S-1, the job completion times in all previous stages have to be recalculated. Because of the recalculations, it is time consuming to calculate makespan along the column dimension.

For the makespan calculation along the row dimension, as n increases from 1 to n, the computational complexity is also $O(Sn^2)$, although makespan calculation is carried out five times. Therefore, the overall computational complexity of the SS heuristic is $O(Sn^2)$.

For an industrial instance with 1396 jobs and 5 machines in Gienow, the computation time of SS heuristic is 70.67 seconds, much longer than 782 milliseconds for CDS heuristic. However, the 782 milliseconds are only for CDS to generate sequences. Taking the makespan calculation into consideration, CDS will have the same computational complexity as SS. Moreover, from the adaptive control perspective, the computational complexity of SS heuristic is only $O(Sn)$, which means it takes only 10.12 milliseconds for SS heuristic to select the next job dealing with disturbances in Gienow for this instance.

4. Case studies

The computational complexities of some existing heuristics and SS heuristic are analyzed in sections 2 and 3 respectively. In this section, the comparison and evaluation of heuristics are mainly based on optimality and flexibility. Two kinds of case studies, with and without disturbances, are carried out on Taillard's benchmarks (Taillard, 1993) and on an industrial

case. Section 4.1 is for without disturbances, Section 4.2 is for with disturbances, and at last Section 4.3 gives a comparison between SS heuristic and Johnson's algorithm (JA).

4.1 Case studies without disturbances

4.1.1 *Fm*/*prmu*/C_{max} on Taillard's benchmarks

For traditional permutation flow shop scheduling problems, the deviation (*DEV*) from the best known upper bounds is used to evaluate the performance of a heuristic, where *DEV* = (C_{max} *of a heuristic* – *The upper bound*) ÷ (*The upper bound*) in percentage. The results of the deviation studies for CDS, NIS and SS, and a version of SS without the lever concept, SSnoLVR, heuristics are shown in Table 4.1.

In Table 4.1, the column "Scale" means the size of problems. For example, 20*5 means 20-job 5-machine problems. The column "Inst" means the number of instances in each scale. Columns 3 to 6 represent the average deviation of each of the CDS, NIS, SS, and SSnoLVR, heuristics respectively. We can see that SS heuristic has the smallest total average deviation for all 120 instances in Taillard's benchmarks, at 8.11%, NIS heuristic ranks the second at 9.01%, and CDS heuristic ranks the last at 11.28%. We can also see from Table 4.1 that the lever concept is suitable for flow shop production to minimize the makespan. The SS heuristic is better than the SSnoLVR heuristic, with a deviation of 8.11% versus 8.80%. To further compare the performance of the SS heuristic with the CDS heuristic's, a *t*-test is carried out using a function of TTEST (CDS results, SS results, 2, 1) in excel. The the SS heuristic's *p*-value is 3.20×10^{-5}, which means the improvement is extremely significant.

Scale	Inst	CDS	NIS	SS	SSnoLVR
20*5	10	9.05	7.41	9.14	7.80
20*10	10	13.48	9.46	10.18	13.13
20*20	10	11.07	7.30	10.64	14.02
50*5	10	7.15	4.96	3.60	3.38
50*10	10	14.46	11.57	9.67	9.24
50*20	10	18.13	14.50	16.15	16.12
100*5	10	5.25	4.70	1.60	1.75
100*10	10	9.51	8.27	6.71	6.05
100*20	10	16.45	13.50	11.83	15.71
200*10	10	7.55	6.61	3.09	2.48
200*20	10	13.75	11.33	9.10	11.31
500*20	10	9.56	8.44	5.63	4.60
Total Average		11.28	9.01	8.11	8.80
MAX		21.13	16.62	20.83	22.02
MIN		0.66	0.86	0.78	0.60

Table 4.1 Average deviations from Taillard's benchmarks for *Fm/prmu/*C_{max} problems (%).

4.1.2 *Fm*/*nwt*/C_{max} on Taillard's benchmarks

For traditional no wait flow shop problems, an improvement (*IMPR*) over NIS heuristic is used to evaluate the performance of CDS and SS heuristics based on Taillard's benchmarks. *IMPR* = (C_{max} of NIS – C_{max} of CDS or SS) ÷ (C_{max} of NIS) in percentage is shown in Table 4.2.

Scale	Inst	CDS	SS
20*5	10	-0.32	2.01
20*10	10	-2.59	-2.86
20*20	10	-3.50	-2.71
50*5	10	0.29	8.29
50*10	10	-1.29	0.49
50*20	10	-2.42	-1.67
100*5	10	-0.27	9.20
100*10	10	-0.61	3.78
100*20	10	-1.00	-0.02
200*10	10	-0.22	5.59
200*20	10	-0.41	1.69
500*20	10	-0.10	3.46
Total Average		-1.04	2.27

Table 4.2 Improvement over NIS heuristic for $Fm/nwt/C_{max}$ problems (%)

In Table 4.2, CDS heuristic performs 1.04% worse than NIS heuristic. In contrast, SS performs better than NIS on average, with an improvement of 2.27%. For the *t*-test based on 12 averages, SS heuristic's *p*-value is 0.0739. However, if the *t*-test is based on 120 individual cases, the SS' *p*-value is 2.07×10^{-11}, which means an extremely significant improvement. Moreover, we recognize that for HFS no wait problems the improvement of SS over NIS will shrink as the number of operators/machines in each stage increases. For example, if the number of operators in each stage is the same as the number of jobs, then C_{max} is fixed as $\max\left(\sum_{s=1}^{S} p_{i,s}\right)$ for $i = 1,\ldots,n$, no matter for no wait or no pre-emption flow shop problems.

4.1.3 *FFs/nwt/*C_{max} on Taillard's benchmarks

For hybrid no wait flow shop problems with identical parallel operators/machines in each stage, two operators/machines are assigned to each stage. The improvement of CDS and SS heuristics over NIS heuristic is shown in Table 4.3.

Scale	Inst	CDS	SS
20*5	10	-1.71	-2.66
20*10	10	-2.72	-2.02
20*20	10	-3.06	-2.88
50*5	10	-0.77	3.34
50*10	10	-1.50	-2.18
50*20	10	-3.48	-2.04
100*5	10	0.21	7.15
100*10	10	-0.55	0.60
100*20	10	-1.75	-1.13
200*10	10	-0.15	3.54
200*20	10	-0.50	0.97
500*20	10	0.01	2.00
Total Average		-1.33	0.39

Table 4.3 Improvement over NIS heuristic for $FFs/nwt/C_{max}$ problems (%)

For such hybrid no wait flow shop problems with two operators/machines in each stage, SS heuristic has a small improvement of 0.39% over NIS heuristic, and CDS heuristic still performs worse, -1.33%. For the *t*-test, SS heuristic's *p*-value is 0.6739, meaning that its improvement over NIS heuristic is not statistically significant.

4.1.4 *FFs/FCFS/*C_{max} on Taillard's benchmarks

Scale	Inst	min(C_{max})	max(*Util*)
20*5	10	-2.39	7.33
20*10	10	0.27	5.66
20*20	10	-2.65	-0.02
50*5	10	2.87	4.90
50*10	10	2.47	6.17
50*20	10	0.08	1.45
100*5	10	2.42	3.47
100*10	10	1.34	4.69
100*20	10	1.54	2.43
200*10	10	3.14	3.96
200*20	10	2.03	4.41
500*20	10	2.79	3.14
Total Average		1.16	3.96

Table 4.4 Improvement over CDS heuristic for *FFs/FCFS/*C_{max} problems (%)

For HFS problems with the *FCFS* rule applied to jobs in WIP inventories, two variables are set. One is a throughput rate r = 31, used to calculate the number of operators in each stage, where $OPTR_s$ = Roundup (APT_s/r). The average processing time of each stage ranges from 30.75 to 64.40 for all of 120 instances in Taillard's benchmarks, therefore, $OPTR_s$ varies from 1 to 3 for each stage. Another variable is the capacity of WIP inventories. Different configurations of WIP inventories have different impacts on production (Vergara & Kim, 2009). For the ease of case study, the capacity of each WIP inventory is set the same, WIP_s = 5, even though in theory each could be set to a different value. The calculation of processing times in CDS is $p'_{i,s} = p_{i,s}/OPTR_s$, s = 1,…,S (Botta-Genoulaz, 2000). For the objective of min(C_{max}), the improvement (*IMPR*) of SS heuristic over CDS heuristic is used to evaluate performance, where $IMPR_1$ = (C_{max} *of CDS* – C_{max} *of SS*) ÷ (C_{max} *of CDS*) in percentage. For the objective of max(*Util*), the improvement of SS heuristic over CDS heuristic is $IMPR_2$ = (*Util of SS – Util of CDS*) ÷ (*Util of CDS*). The results are shown in Table 4.4.

For the objective of max(*Util*), SS heuristic has an average 3.96% improvement over CDS heuristic on Taillard's benchmarks, and for the objective of min(C_{max}), SS heuristic has an average 1.16% improvement. For the *t*-test, SS heuristic's *p*-value is 0.0666 for min(C_{max}) meaning the improvement over CDS heuristic is not quite statistically significant. However, for max(*Util*), the *p*-value of SS heuristic is 3.34×10^{-5}, an extremely significant improvement.

4.1.5 An industrial case study

To validate the SS heuristic in a real setting, an industrial case study was carried out in Gienow Windows and Doors, Canada. This case consists of 1396 jobs on a flow line with 5

stages for one-day production. These jobs are delivered to customers at a predetermined time in 28 batches. Each batch of products is destined for customers in a given geographic area. Using data provided by Gienow, SS heuristic produces the results shown in Table 4.5. In the SS heuristic, the *SPACE* is modelled as $SPACE_{i,s} = c_{i,s+1} - A_s$.

	Gienow	SS	IMPR		Gienow	SS	IMPR
1	1,795	1,711	84	16	1,489	1,489	0
2	1,458	1,444	14	17	1,477	1,477	0
3	1,698	1,697	1	18	1,743	1,712	31
4	2,292	2,261	31	19	1,751	1,745	6
5	1,570	1,556	14	20	1,434	1,430	4
6	1,798	1,753	45	21	1,587	1,570	17
7	1,420	1,420	0	22	1,587	1,393	194
8	1,573	1,567	6	23	1,196	1,165	31
9	1,828	1,805	23	24	1,094	1,083	11
10	1,676	1,676	0	25	1,362	1,362	0
11	1,568	1,568	0	26	1,281	1,281	0
12	1,691	1,691	0	27	923	923	0
13	1,465	1,465	0	28	857	851	6
14	1,364	1,353	11	Total	42,300	41,771	529
15	1,323	1,323	0	Percent			1.25%

Table 4.5 An industrial case study

As shown in Table 4.5, Gienow used 42,300 time units to finish 1396 jobs. The production of 1396 jobs in 42,300 time units was achieved by Gienow's original schedule, which was generated by an experienced production scheduler in Gienow. SS heuristic can generate new schedules, respectively reducing 42,300 time units to 41,771, a 1.25% improvement in productivity. Such improvement translates into the production of 17 additional products daily, or more than $5000 revenue per day. For the *t*-test, SS heuristic's *p*-value is 0.0164, which means the improvement is very significant.

4.2 Case studies with disturbances

To test the suitability of SS heuristic to adaptive production control, a case study of operator absence is carried out on Taillard's benchmarks. Modeling operator absence is the same as modeling machine breakdown. We assume that, when a half of jobs are finished, one operator is absent in the middle stage of a flow line, specifically in stage 3, 6, or 11 according to the scale of instances in Taillard's benchmarks. For the remaining half of the jobs, if the production is carried on according to the original schedule when such disturbances happen to the shop floor, then the completion time is recorded as *Original*. If adaptive control is applied, that is, using SS heuristic to re-schedule the remaining jobs, then the completion time is recorded as *Adaptive*. The improvement of adaptive control over no adaptive control is used to evaluate the performance, i.e. (*Original* – *Adaptive*) ÷ (*Original*) in percentage.

To show the potential of the SS heuristic, case studies on operator absence are carried out under the two definitions of *SPACE*, $SPACE_{i,s} = p_{i,s+1}$ and $SPACE_{i,s} = c_{i,s+1} - A_s$. Moreover, a simple optimization method is also integrated with the SS heuristic.

4.2.1 $SPACE_{i,s} = p_{i,s+1}$

The results are given in Table 4.6. As we see, adaptive control is slightly better than no adaptive control with a 0.10% improvement for the SS heuristic if $SPACE_{i,s} = p_{i,s+1}$.

Scale	Inst	SS
20*5	10	2.46
20*10	10	1.81
20*20	10	3.01
50*5	10	0.88
50*10	10	2.17
50*20	10	-2.80
100*5	10	0.39
100*10	10	0.29
100*20	10	-4.18
200*10	10	-0.41
200*20	10	-1.25
500*20	10	-1.21
Total Average		0.10

Table 4.6 Adaptive control over no adaptive control, where $SPACE_{i,s} = p_{i,s+1}$

4.2.2 $SPACE_{i,s} = c_{i,s+1} - A_s$

The results are given in Table 4.7. As we see, for SS heuristic, if we model $SPACE_{i,s} = c_{i,s+1} - A_s$, the adaptive control has a 2.02% improvement over no adaptive control.

Scale	Inst	SS
20*5	10	7.75
20*10	10	6.62
20*20	10	-8.49
50*5	10	1.23
50*10	10	3.10
50*20	10	2.99
100*5	10	0.22
100*10	10	3.26
100*20	10	4.50
200*10	10	0.55
200*20	10	1.64
500*20	10	0.86
Total Average		2.02

Table 4.7 Adaptive control over no adaptive control, where $SPACE_{i,s} = c_{i,s+1} - A_s$

4.2.3 Integration with an optimization method

There are two effects in SS heuristic impacting the final production performance. *SPACE* is good for production but "*IDLE* & *DELAY*" is bad. We can introduce a weighting factor, α,

into SS heuristic, and then sequence jobs according to $\max[(1-\alpha)\times \sum_{s=1}^{S-1} SPACE'_{is} - \alpha\times(\sum_{s=1}^{S-1} IDLE'_{is} + \sum_{s=1}^{S-1} DELAY'_{is})]$. As α changes from 0 to 1 with increments of 0.1, the performance of SS heuristic, with $SPACE_{i,s} = p_{i,s+1}$, is shown in Table 4.8. The columns represent the performance of each α integrated with SS heuristic. A weight α = 0.0 means no *IDLE* or *DELAY* is taken into consideration to sequence jobs, and α = 1.0 means no *SPACE*. We can see that *SPACE* affects the production more than *IDLE* or *DELAY*, where α = 0.1 has the greatest improvement of 2.77%.

Scale	Inst	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
20*5	10	-0.59	0.75	2.17	3.12	1.97	2.46	-2.31	-7.08	-7.68	-9.25	-2.30
20*10	10	0.67	0.77	0.56	5.01	3.68	1.81	-2.86	-4.62	-2.11	-1.59	-4.27
20*20	10	7.50	10.0	8.69	6.74	6.15	3.01	0.00	-9.46	-9.12	-8.99	-9.79
50*5	10	0.22	1.11	1.17	0.97	1.55	0.88	0.15	-1.66	-1.33	-1.24	-0.64
50*10	10	5.50	4.97	4.06	2.80	2.75	2.17	-4.35	-7.10	-6.20	-8.76	-8.89
50*20	10	1.65	3.67	1.94	2.90	-5.41	-2.80	-5.53	-5.22	-7.50	-0.22	-1.07
100*5	10	0.43	0.88	0.36	0.49	0.58	0.39	0.17	-0.98	-1.09	-1.16	-2.24
100*10	10	3.05	3.02	2.20	2.42	1.05	0.29	-2.35	-1.97	-2.76	-1.42	-2.50
100*20	10	5.94	5.24	5.79	4.76	-0.39	-4.18	-5.26	-5.76	-6.61	-5.73	-7.49
200*10	10	0.47	0.47	0.05	0.27	0.15	-0.41	-0.75	-1.26	-1.10	-1.58	-1.15
200*20	10	1.53	2.24	2.39	0.82	-0.15	-1.25	-1.44	-1.48	-2.34	-2.36	-1.98
500*20	10	0.17	0.18	0.16	-0.02	-0.64	-1.21	-1.63	-1.69	-1.66	-1.59	-1.54
Total average		2.21	2.77	2.46	2.52	0.94	0.10	-2.18	-4.02	-4.96	-5.33	-6.15

Table 4.8 Adaptive control with α over no adaptive control, where $SPACE_{i,s} = p_{i,s+1}$

4.2.4 Case studies on variation in processing times

It is normal to have variation in processing times, especially for the production of highly customized products and with manual operations. Thus, it is necessary to test the suitability of SS heuristic to the disturbance of variation in processing times. In Gienow, processing times of products are quoted by the company standards.

For variation in processing times in the industrial case, we assume that, initially, we have a matrix of quoted processing times of n jobs in S stages, and we do not know the real processing time beforehand, because the perfect production information in OKP can be available only after the production (Wortmann, 1992). If we define this matrix as B, which means before production, we carry out the offline scheduling according to B to get a sequence SB. We might setup due dates based on the performance of SB, that is, the original performance, PO. After the actual production, we have a matrix of real processing times of n jobs in S stages, i.e. matrix A.

During the production, when variation in processing times happens and the production is carried out according to the sequence SB, the performance is PB. It means no adaptive control. It is difficult to use CDS heuristic for such disturbance, because we only know part of matrix A for finished jobs, but not the rest for unfinished jobs. However, we can adaptively re-schedule the rest jobs by SS heuristic, because the actual processing times of finished jobs affect the space, although we only know matrix B for the unfinished jobs. After one job has been produced, we use SS heuristic to select a job from remaining jobs according to processing times of unfinished jobs in matrix B and the actual space created by finished

jobs. Consequently, the performance of adaptive control by SS heuristic is *PA*. The *SPACE* of SS heuristic is modeled as $SPACE_{i,s} = c_{i,s+1} - A_s$. We compare the performance of no adaptive control *PB* or adaptive control *PA* with the original performance *PO*, by: *Diff_OB* = (*PB* – *PO*) ÷ *PO* and *Diff_OA* = (*PA* – *PO*) ÷ *PO*, both of which are in percentage. Four ranges of normal distribution are introduced into the processing times in the industrial case, [-5%, 5%], [0%, 10%], [0%, 25%] and [0%, 50%]. The results are summarized in Tables 4.9.

		Diff_OB	*Diff_OA*
[-5%, 5%]	Average	0.83	0.27
	MAX	3.76	1.17
	MIN	0.01	0.03
[0%, 10%]	Average	5.45	5.12
	MAX	7.89	5.88
	MIN	4.00	4.47
[0%, 25%]	Average	13.08	12.76
	MAX	15.72	15.69
	MIN	9.72	9.37
[0%, 50%]	Average	25.37%	24.98
	MAX	28.75%	28.19
	MIN	20.94%	20.38

Table 4.9 Adaptive control for variation in processing times

From Table 4.9, we can see that adaptive control performs better than no adaptive control for all four ranges of variation in processing times. Moreover, SS heuristic is stable to such disturbance, because its average difference of performance is close to the expected value of variation in four ranges respectively.

4.3 A case study on a 2-machine flow shop problem

To reveal the rationale and coherent logic of the state space concept, a scaled down version of SS heuristic is compared with JA for a 2-machine flow shop problem, $F2/prmu/C_{max}$. For $F2/prmu/C_{max}$ problems, the lever concept has no effect on the job selection in SS heuristic. This is because for this type of $F2/prmu/C_{max}$ problems, the WIP inventory between machines 1 and 2 is unlimited, thus no *DELAY* is taken into consideration. In addition, the length of force arm for *SPACE* or *IDLE* equals to one. However, the state space concept can yield different job sequences than JA. A numerical example is provided in Table 4.10.

	M_1	M_2
Job 1	5	20
Job 2	20	10
Job 3	10	15
Job 4	15	12

Table 4.10 A 2-machine flow shop example

JA sequences jobs according to the following scheme. If min $\{p_{i,1}, p_{i+1,2}\} \leq$ min $\{p_{i+1,1}, p_{i,2}\}$, then job *i* should be processed earlier than job *i*+1. Therefore, for the example in the above table,

JA generates a sequence of [Job 1, 3, 4, 2] with C_{max} = 62. According to the state space concept (but not exactly SS heuristic), and using JA for the initial job selection, two additional sequences can be obtained, [Job 1, 2, 3, 4] and [Job 1, 4, 3, 2], both of which have C_{max} = 62, and are different from the one generated by JA. Therefore, it is obvious that JA uses a sufficient condition for $F2/prmu/C_{max}$ problems, but not necessary in some cases. The state space concept can yield different sequences than JA with the same level of optimality, and hence can provide greater opportunities for improvement as the core of a more elaborate heuristic for adaptive production scheduling and control.

5. Conclusions and future work

One-of-a-kind production (OKP) challenges production scheduling differently from mass production, because of high customer involvement in OKP. Especially, it challenges production control severely, because of dynamic disturbances. Traditionally, offline production scheduling is separated from the online adaptive production control. Dynamic disturbances in OKP fail the production schedule, which are generated by heuristics that are developed based on strong assumptions for offline scheduling (MacCarty & Liu, 1993). Accordingly, adaptive production control is in need to deal with disturbances. Currently, the adaptive production control in OKP companies is carried out by shop floor managers using priority dispatching rules (PDRs) and their experience. However, the performance of PDRs is poor on most scheduling objectives (Ruiz & Maroto, 2005), and the experience might be good for local optimization but definitely lacks global optimization for the overall production. Therefore, the adaptive production scheduling and control is essential and indispensable to improve the production efficiency in OKP.

In regards to three criteria of optimality, computational complexity, and flexibility to evaluate a heuristic for adaptive production control (Li et al., 2011a), the state space (SS) heuristic is the better than most existing heuristics. For optimality, SS heuristic outperforms the most popular alternative heuristics (CDS, NIS) against Taillard's benchmarks no matter for $Fm/prmu/C_{max}$, $Fm/nwt/C_{max}$, $FFs/nwt/C_{max}$ and $FFs/FCFS/C_{max}$ problems. In addition, the production schedule generated by SS heuristic outperforms Gienow's original schedule, improving Gienow's daily productivity by 1.25%. For computational complexity, $O(m^2n+mnlogn)$ of CDS heuristic is simpler one than $O(mn^2)$ of SS heuristic for offline scheduling. However, if taking sequence evaluation into consideration, they have the same computational complexity of $O(mn^2)$. In addition, for online adaptive production control, the computational complexity of SS decreases to $O(mn)$, but that of CDS keeps the same. For flexibility, SS heuristic is more flexible than other heuristics. SS heuristic can deal with all three typical disturbances proposed by Pinedo (2002), job insertion or cancellation, operator absence or machine breakdown, and variation in processing times, whereas, CDS cannot deal with variation in processing times. Although NEH heuristic has the best performance for $Fm/prmu/C_{max}$ problems, its inflexible procedure to construct a job sequence renders it little flexibility to deal with disturbances. Moreover, SS heuristic is in the phase of index development, a phase that is beneficial for heuristic development in the other two phases (Framinan et al. 2004).

As discussed in this chapter, adaptive production scheduling and control in OKP challenges nearly all existing scheduling algorithms and heuristics, and almost all manufacturing companies are facing a certain level of disturbances, such as unreliable

supplies, unexpected operator absence, machine breakdowns, etc. There is still a gap between the theoretical research and industrial applications. Industrial applications require further understandings and studies of production scheduling and control. This draws the following future work. (1) Production planning on the company level should be integrated with production scheduling and control on the shop floor level. Production planning provides a company the production capacity that is a constraint for adaptive production scheduling and control. Meanwhile, the adaptive production scheduling and control requires frequent re-planning according to the production progress under unexpected disturbances. This is to meet due dates of customer orders or provide better estimated lead-times. The synergy and co-optimization between these two levels are necessary and should be further researched. (2) Consequently, adaptive production scheduling and control for non-deterministic problems is inevitable. Stochastic modeling or simulation for non-deterministic production problems is a valuable research topic and lucrative. (3) It is critical to integrate material flows on shop floors into a supply chain to successfully achieve adaptive production scheduling and control in OKP. This is in fact an urgent research topic to be studied.

6. References

Agarwal, A.; Colak, S. & Eryarsoy, E. (2006). Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach. *European Journal of Operational Research*, Vol.169, No.3, pp. 801-815

Blecker, T. & Friedrich, G. (Eds.) (2006). *Mass Customization: Challenges and Solutions*. Springer, ISBN 987-038-7322-22-3, New York, USA

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, Vol.64, No.1, pp. 101-111

Campbell, H.G.; Dudek, R.A. & Smith, M.L. (1970). A heuristic algorithm for the n-job, m-machine scheduling problem. *Management Science*, Vol.16, No.10, pp. 630-637

Dannenbring, D.G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, Vol.23, No.11, 1174–1182

Dean, P.R.; Tu, Y.L. & Xue, D. (2009). An Information System for One-of-a-Kind Production. *International Journal of Production Research*, Vol.47, No.4, pp. 1071-1087

Dean, P.R.; Tu, Y.L. & Xue, D. (2008). A Framework for Generating Product Production Information for Mass Customization. *International Journal of Advanced Manufacturing Technology*, Vol.38, No.11-12, pp. 1244-1259

Dudek, R.A. & Teuton Jr., O.F. (1964). Development of *M*-state decision rule for scheduling *n* jobs through *M* machines. *Operations Research*, Vol.12, No.3, pp. 471-497

Framinan, J.M.; Gupta, J.N.D. & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, Vol.55, No.12, pp. 1243-1255

Garey, M.R.; Johnson, D.S. & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, Vol.1, No.2, pp. 117-129

Goyal, S.K.; Mehta, K.; Kodali, R. & Deshmukh, S.G. (1995). Simulation for analysis of scheduling rules for a flexible manufacturing system. *Integrated Manufacturing Systems*, Vol.6, No.5, pp. 21-26

Graham, R.L.; Lawler, E.L.; Lenstra, J.K & Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, Vol.5, pp. 287-326

Gupta, J.N.D. & Stafford, E.F. (2006). Flowshop Research after Five Decades. *European Journal of Operational Research*, Vol.169, No.3, pp 699-711

Ho, J.C. & Chang, Y.L. (1991). A new heuristic for the n-job, m-machine flow-shop problem. *European Journal of Operational Research*, Vol.52, pp. 194–202

Hundal, T.S. & Rajgopal, J. (1988). An extension of Palmer's heuristic for the flow shop scheduling problem. *International Journal of Production Research*, Vol.26, No.6, pp. 1119-1124

Ignall, E. & Schrage, L. (1965). Application of branch-and-bound technique to some flow shop problems. *Operations Research*, Vol.13, No.3, pp. 400-412

Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, Vol.1, No.1, pp. 61-68

Karush, W. (1965). A counterexample of a proposed algorithm for optimal sequencing of jobs. *Operations Research*, Vol.13, No.2, pp. 323-325

King, J.R. & Spachis, A.S. (1980). Heuristics for flow-shop scheduling. I*nternational Journal of Production Research*, Vol.18, No.3, pp. 345-357

Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research*, Vol.105, No.1, pp. 66-71

Kouvelis, P.; Chambers, C. & Yu, D.Z. (2005). Manufacturing operations manuscripts published in the first 52 issues of POM: review, trends, and opportunities. *Production and Operations Management*, Vol.14, No.4, pp. 450-467

Lageweg, B.J.; Lenstra, J.K. & Rinnooy Kan, A.H.G. (1978). A general bounding scheme for the permutation flow-shop problem. *Operations Research*, Vol.26, No.1, pp. 53-67

Li, W. (2006). Adaptive *Production Scheduling and Control in One-of-a-Kind Production*, Thesis (M.Sc.), University of Calgary, Canada

Li, W.; Nault, B.R.; Xue, D. & Tu, Y.L. (2011a). An efficient heuristic for adaptive production scheduling and control in one-of-a-kind production. *Computers & Operations Research*, Vol.38, No.1, pp. 267-276

Li, W.; Luo, X.G.; Xue, D. & Tu, Y.L. (2011b). A heuristic for adaptive production scheduling and control in flow shop production. *International Journal of Production Research*, Vol.49, No.11, pp. 3151–3170

Linn, R. & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering*, Vol.37, No.1, pp. 57-61

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*, Vol.31, No.1, pp. 59-79

McKay, K.; Pinedo, M. & Webster, S. (2002). Practice-focused research issues for scheduling systems. *Production and Operations Management*, Vol.11, No.2, pp. 249-258

Nawaz, M.; Enscore, E.E. & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA The International Journal of Management Science*, Vol.11, No.1, pp. 91-95

Ovacik, I.M. & Uzsoy, R. (1997). D*ecomposition Methods for Complex Factory Scheduling Problems*. Kluwer Academic Publishers, ISBN 079-2398-351, Boston, USA

Palmer, D. (1965). Sequencing jobs through a multi-stage process in the minimum total time – a quick method of obtaining a near optimum. *Operational Research Quarterly*, Vol.16, No.1, pp. 101-107

Park, S.C.; Raman, N. & Shaw, M.J. (1997). Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach. *IEEE Transactions on Robotics and Automation*, Vol.13, No.4, pp. 486-502

Pinedo, M. (2002). *Scheduling Theory, Algorithms, and Systems*. Prentice Hall, ISBN 013-0281-387, New Jersey, USA

Potts, C.N.; Shmoys, D.B. & Williamson, D.P. (1991). Permutation vs. non-permutation flow shop schedules. *Operations Research Letters*, Vol.10, No.5, pp. 281-284

Ruiz, R. & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165, No.2, pp. 479-494

Ruiz, R. & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, Vol.177, No.3, pp. 2033-2049

Smith, R.D. & Dudek, R.A. (1967). A general algorithm for solution of the n-job M-machine sequencing problems of the flow shop. *Operations Research*, Vol.15, No.1, pp. 71-82 Also see their correction (1969). Errata. *Operations Research*, Vol.17, No.4, pp. 756

Suliman, S. (2000). A two-phase heuristic approach to the permutation flow-shop scheduling problem. *International Journal of Production Economics*, Vol.64, No.1-3, pp. 143-152

Szwarc, W. (1971a). Elimination methods in the *m*×*n* sequencing problem. *Naval Research Logistics Quarterly*, Vol.18, No.3, pp. 295-305

Szwarc, W. (1971b). Optimal elimination methods in the *m*×*n* flow-shop scheduling problem. *Operations Research*. Vol.16, No.3, pp. 250-1259

Taillard E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, Vol.64, No.2, pp. 278-285

Taillard, E. (1990). Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of Operational Research*, Vol.47, No.1, pp. 65-74

Thornton, H.W. & Hunsucker, J.L. (2004). A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage. *European Journal of Operational Research*, Vol.152, No.1, pp. 96-114

Tu, Y.L. (1996a). A Framework for Production Planning and Control in a Virtual OKP Company. *Trans. North American Manufacturing Research Institution of SME*, Vol.24, pp. 121-126

Tu, Y.L. (1996b). Automatic Scheduling and Control of a Ship Welding Assembly Line. *Computers in Industry*, Vol.29, No.3, pp. 169-177

Vergara H.A. & Kim, D.S. (2009). A new method for the placement of buffers in serial production lines. *International Journal of Production Research*, Vol.47, No.16, pp. 4437-4456

Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristic and artificial intelligence solutions. *Expert Systems*, Vol.22, No.2, pp. 78-85

Wortmann, J.C. (1992). Production management systems for one-of-a-kind products. *Computers in Industry*, Vol.19, No.1, pp. 79-88

Wortmann, J.C.; Muntslag, D.R. & Timmermans, P.J.M. (1997). *Customer-Driven Manufacturing*. Chapman & Hall, ISBN 041-2570-300, London, UK

Production Scheduling

Edited by Prof. Rodrigo Righi

ISBN 978-953-307-935-6
Hard cover, 242 pages
Publisher InTech
Published online 11, January, 2012
Published in print edition January, 2012

Generally speaking, scheduling is the procedure of mapping a set of tasks or jobs (studied objects) to a set of target resources efficiently. More specifically, as a part of a larger planning and scheduling process, production scheduling is essential for the proper functioning of a manufacturing enterprise. This book presents ten chapters divided into five sections. Section 1 discusses rescheduling strategies, policies, and methods for production scheduling. Section 2 presents two chapters about flow shop scheduling. Section 3 describes heuristic and metaheuristic methods for treating the scheduling problem in an efficient manner. In addition, two test cases are presented in Section 4. The first uses simulation, while the second shows a real implementation of a production scheduling system. Finally, Section 5 presents some modeling strategies for building production scheduling systems. This book will be of interest to those working in the decision-making branches of production, in various operational research areas, as well as computational methods design. People from a diverse background ranging from academia and research to those working in industry, can take advantage of this volume.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen